Outline

• Motivation/Problem
  - Characteristics of sensor networks
  - Techniques for building sensor network applications
  - Why sensor network programming hard?
  - Need: High-level programming abstraction

• Our approach
  - Active dataspace (ADS)
  - Features of ADS
  - Tuples on demand

• Example scenario
  - Vehicle detection: Setup
  - Event sequence: Bootstrapping
  - Event sequence: Detection phase
Characteristics of sensor networks

- Resource constraints
  - Energy reserve
  - Computation power
  - Memory

- Unpredictable communication links
  - Intermittent connectivity
  - Asymmetric links and radio directionality

- Node failure
  - Exposed to harsh environment and attacks
  - Battery depletion

- Possibly large number of nodes
- Possibly difficult deployment environment
Techniques for building sensor network applications

• General resource conservation
  – In-network processing
  – Localized algorithms
  – Hibernation (e.g., sentry service)

• Optimizations for key communication patterns
  – Tree-based aggregation scheme (many-1)
  – Firecracker protocol (1-many)

• Adaptive protocols
  – Protocols that adaptively optimize communication based on local information and feedback (e.g., directed diffusion and PARC’s CB-LRTA*)

• Exploiting redundancy and broadcast medium
Why sensor network programming hard?

- Deploying new or additional sensors
- Limited CPU power and memory
- Scalability
- Locality
- Data aggregation
- Hibernation
- Applications
- Intermittent end-to-end connectivity
- Attacks

Sensor nodes
Need: High-level programming abstraction

Applications

- Optimizing CPU & memory use
- Aggregation
- Security
- Scalability
- Locality

- Naming
- Deploying new or additional sensors
- Intermittent end-to-end connectivity
- Hibernation

Sensor nodes

Focus of this project
Active dataspace (ADS)

- ADS is an active data repository that provides associative operations for data access
- Inspired by the tuple space model [Gelernter 85], developed for parallel computing
- Every data tuple (or record) contains a list of fields
- Basic TS operations:
  - *in* is used to remove tuples from TS
  - *rd* to read tuples
  - *out* to create data tuples
  - *eval* to create “active” tuples
Features of ADS

• Data-centric model
• Time-uncoupling: Data consumers and producers do not need to be active at the same time
• Identity-uncoupling: Endpoints do not need to know each other’s identities
• Stable network paths between endpoints need not exist
• Virtual tuples support data generation on demand
• Tuple set operator and cardinality constraint to facilitate in-network aggregation
• Search constraint for specifying the scope and preferences for tuple selection to exploit locality
Tuples on demand

- Motivation: To enable sensor nodes to conserve energy and other resources during time intervals in which their work is not needed.
- A virtual tuple represents the capability of a node to generate a certain type of tuple specified by the virtual tuple.
- When a tuple request matches a virtual tuple, the corresponding node will be contacted to produce the data on demand.
- Use of virtual tuple is transparent to data consumers.
Vehicle detection: Setup

- Sensors deployed in a region for vehicle detection
- 2 types of sensor nodes
- Type 1 (Sensors A, B, and C) :
  - Low-cost to operate
  - less accurate
  - has a shorter range
  - cannot classify vehicles
- Type 2 (Sensor X):
  - Expensive to use
  - more accurate
  - have a longer range
  - can distinguish different classes of vehicles
Event sequence: Bootstrapping

- Sensors put virtual tuples in ADS, which represent their detection capabilities.
- Sensor \( X \) hibernates.
- Sensors \( A, B, \) & \( C \) take turn being active (i.e., the sentry).

\[
\text{outv(type1,?\{A,B,C\})}
\]

\[
\text{outv(type2,?\{X\})}
\]
Event sequence: Detection phase (1)

1. A detects a vehicle
2. A requests other type1 sensor data
3. ADS matches the request with B’s and C’s virtual tuples
4. B and C produce sensor reading tuples
5. A confirms detection with B’s and C’s input
Event sequence: Detection phase (2)

1. \textit{A} requests type2 sensor data
2. \textit{ADS} matches the request with \(X\)'s virtual tuple
3. \(X\) is awaken for vehicle detection
Expected results

- High-level programming model and language to ease sensor network programming for a wide range of application domains
- Architecture and techniques to implement a resource-efficient, adaptive, and trustworthy ADS system