Intelligent Sensors and Sensor Networks

Week 3
Guerrilla Management

- A supervisor/agency architecture for scalable and cooperative management
- Uses mobile code techniques for nomadic and active management
- Uses a dynamic adaptive protocol for clustering and selecting nomadic managers
- Nodes range in functionality and capability
  - SNMP-capable, Probe-capable, and Full-featured

Two-tier infrastructure to facilitate adaptive, autonomous and robust management of ad hoc nets
Guerrilla Management Information Base GMIB

- **Nomadic Manager**
  - Collaborates autonomously to manage the entire ad hoc network with minimal help from the supervisor
    - Role change
    - Load sharing - cloning itself into another node
    - Spawning and merging

- **GMIB (Guerrilla MIB)**
  - A data structure equivalent to a SNMP MIB
    - An aggregation of management information collected from neighbor nodes via probes
    - Maintained inside NMM
  - Also includes
    - Management information (e.g., neighbor information) in the probe processing modules
  - Can be accessed by both the NMM and incoming probes
    - Modeled as a branch in a SNMP MIB
Specific Management Functions: Power Management

- Manages how a sensor node uses its power

- Example
  - Sensor node may turn off its receiver after receiving a message from one of its neighbors
    - avoid getting duplicated messages
  - When the power level of the sensor node is low
    - Broadcasts to its neighbor when it is low in power
    - Cannot participate in routing messages
    - Reserve the remaining power for sensing

- Requirements
  - Using battery
  - Limited Power
  - Expand the life time of sensor node
  - Reduce the overhead

Simple Routing ↔ Processing
Power Management in Protocol Layer

- **Physical layer**
  - Low Power Modulation Scheme
  - Transceiver, Sensor, Process: Small, Low Power, Low Cost

- **Data link layer**
  - Energy efficiency MAC protocol
    - Adaptive duty cycling – S-MAC, ASCENT, SPAN
    - Wake up on-demand – STEM, Wake-on-Wireless
  - Reduce the collision, signaling, frame overhead
  - Power saving mode (ex. On/Off mode)

- **Network Layer**
  - Energy-efficiency routing
  - Energy-efficiency data aggregation algorithms
  - Location based routing

- **Transport Layer**
  - Use UDP message protocol between Sink and Sensor node
  - Limited memory and processing power

- **Application Layer**
  - Energy-efficiency Applications
Topology Management

Goal
- is to coordinate the sleep transitions of all nodes, while ensuring adequate network connectivity, such that data can be forwarded efficiently to the data sink.

Requirements
- Heterogeneous node
- Data discovery & data dissemination
- Limited memory & power constraint
- Application requirements
- Node mobility

Ad-hoc Self-organization
- LCA (Linked Cluster Algorithm)
- LAA (Link Activation Algorithm)
- DEA (Distributed Evolution Algorithm)
- **SMACS** (Self-Organizing Medium Access Control for Sensor networks)
- **EAR** (Eavesdrop And Register)
  - BI (Broadcast Invite)
  - MI (Mobile Invite)
  - MR (Mobile Response)
  - MD (Mobile Disconnect)
- **SAR** (Sequential Assignment Routing)
- **SWE** (Single Winner Election)
- **MWE** (Multi Winner Election)
Security Management

- **Requirements**
  - Peanut CPU (slow computation rate)
  - Battery power: trade-off between security and battery life
  - Limited memory
  - High latency: conserve power, turn on periodically

- **Security Management in USN**
  - Applications need security (privacy)
  - Absence of security enables attacks such as spoofing & replay attacks, resulting in DoS or system compromise
  - Intrusion prevention: First line of defense
  - Intrusion detection: Second line of defense

- **Main Security Threats in USN**
  - Radio links are insecure
  - Sensor nodes are not temper resistant

- **Attacker types**
  - Mote-class
  - Outside / inside
**Attacks**

- Physical attack
- Denial-of-service
- Battery exhaustion
- Clock synchronization
- Location discovery
- Attacks on routing
  - spoofed, altered, or replayed routing information
  - selective forwarding
  - sinkhole attack
  - sybil attack
  - wormholes
  - HELLO flood attacks
  - acknowledgment spoofing
**Countermeasures**

- Link layer encryption – selective forwarding
- Using a counter – Replay attacks
- Limiting the number of neighbors per node – Insider attacks
- Bi-directionality of the link – HELLO flood
- Geographically routing – Wormhole attacks
Simulation Tools

Existing Simulators

- **JavaSim:**
  - **Pros**
    - Very modular
    - Easy to use
  - **Cons**
    - Geared for wired inter-networks
    - No wireless support, not efficient due to overhead

- **GlomoSim:**
  - Specific for mobile wireless networks.
  - Built as a set of libraries. The libraries are built in Parsec (a C-based discrete event simulation language).
  - Layered architecture with easy plug-in capability.

- **SSFNet and Glomosim are not better than NS-2 in terms of design and extensibility.**
Simulators

- NS-2: De facto standard for network simulations
  - Does support wireless simulations
  - A primitive energy model is present.
  - Object oriented design and Lots of documentation.
  - Uses Tcl to specify the Components, and Otcl to glue them together.

- Cons:
  - Difficult to use and learn
  - Interdependency among modules pose difficult to implement new protocols.
  - Originally built for wired networks, later extended for wireless.
  - Supposedly, does not work well for large topologies.
Simulators: Sense

- Battery Model:
  - Linear Battery
  - Discharge Rate Dependent and/or Relaxation Battery

- Application Layer:
  - Random Neighbor
  - Constant Bit Rate

- Network Layer:
  - Simple Flooding
  - A simplified version of AODV/ADOV (Ad hoc On Demand Distance Vector) without route repairing
  - A simplified version of DSR (dynamic source routing) without route repairing
  - Self Selective Routing (SSR)
  - Self Healing Routing (SHR)
Sense

- MAC Layer:
  - NullMAC
  - IEEE 802.11 with DCF
- Physical Layer: Duplex Transceiver
- Wireless Channel:
  - Free Space
  - Adjacency Matrix
- Simulation Engine: CostSimEng (sequential)
- Publications
SensorSim

- Extension to NS - 2.
  - Provides battery models, radio propagation models and sensor channel models.
  - Provides a lightweight protocol stack.
  - Has support for hybrid simulation.
  - Must be integrated with NS - 2.

SensorSim: A Simulation Framework for Networks
Sensor
Sung Park, Andreas Savvides, and Mani B. Srivastava
Electrical Engineering Department, University of California in Los Angeles
7702-B, Boelter Hall, Box 951594, Los Angeles, CA 90095-1594
Email: ~spark, asavvide, mbs}@ee.ucla.edu

http://compilers.cs.ucla.edu/emsoft05/ParkSavvidesSrivastava00.pdf
SensorSim Architecture

- **Monitor and Control**
  - Hybrid network (local or remote)

- **Real Sensor Apps**
  - On virtual sensor nodes

**Diagram Overview**

- **GUI**
- **app**
- **app**

**Components**

- **GUI Interface**
- **HS Interface**
- **ns**
- **Modified Event Scheduler**

**Network Connectivity**

- **Socket Comm**
- **Serial Comm**
- **Gateway**

**Protocols**

- **Ethernet**
- **RS232**

**Machine Types**

- **Simulation Machine**
- **Gateway Machine**

**Proxies**

- **Proxies for real sensor nodes**
SensorSim Architecture Overview

- Sensor NW has three types of nodes:
  - **Sensor nodes**: monitor immediate environment, with many transducers
  - **Target nodes**: generate various stimuli for sensor nodes
  - **User nodes**: client and administration of sensor network

- Separate channels:
  - **Sensor channels**: communication among sensor nodes and target
  - **Network channels**: to user node or gateways and onward

- Transmission to other network.
  - Concurrent transmission possible
  - Easier to model complex behavior of sensor nodes, reaction to
- Multiple sensor signals.
Typical Scenario

- Nodes coordinator: local processing among neighbors to combine their results
  - lower network traffic, higher-level sensory tasks
- Application: large scale, dynamically changing, robust sensor colonies
Sensor Node Model in SensorSim

- Node Function Model
  - Applications
  - Middleware
    - Network Protocol Stack
      - Network Layer
      - MAC Layer
      - Physical Layer
    - Sensor Protocol Stack
      - Sensor Layer
      - Physical Layer

- Wireless Channel
- Sensor Channel

- Power Model
  - Radio Model
  - CPU Model
  - Sensor #1 Model
  - Sensor #2 Model
  - Battery Model

- State Change
- Status Check
Battery Model

- Ideally battery capacity is decided by the amount of active material stored in a cell.
- In reality, this depends on how the battery is discharged:
  - discharge rate
  - discharge profile
  - operating voltage and power drained
Radio Model

Example Values:

\[ E_{\text{elec}} = 50\text{nJ/bit} \]
\[ \varepsilon_{\text{amp}} = 100\text{pJ/bit/m}^2 \]
Battery Zero Reached (BZR) Event:
The BZR event is used to indicate when the energy stored in the battery is completely drained.
The BZR event is created by the Battery Model every time the battery model handles the BCP event.

Battery Threshold Reached (BTR) Event: The BTR event is similar to BZR event. The difference comes from the fact that BTR event is inserted when the energy level of the battery reaches a certain threshold.

Battery Change Power (BCP) Event: The BCP Event is created by the energy consumers to change their rate of power draw.
Power Management Model

Without Power Management

With Power Management
Power Model Simulation Results

Effect of Power Management for Light Traffic Load

Effect of Power Management for Heavy Traffic Load

Energy in Joules

sec
Hybrid Simulation

- **Real application support**
  - The same applications that run on the real nodes can also run on the simulated nodes

- **Interaction with real nodes**
  - Real sensor nodes can participate in the simulation

- **Advantages**
  - Enables the use of real traffic from the sensor channel that is currently not well understood and the models are not yet mature
  - Validate protocols and applications running on the real nodes by being able to test these applications in large networks
  - Study the behavior of sensor network protocols and applications at scale
Hybrid Simulation Challenges

- Virtual Time Synchronization between ns and external entities
  - done between ns and external processes
- Real and Virtual Time Synchronization
- Hybrid Modeling of Wireless Channel
  - collision between packets from real and virtual nodes
  - Open Problem!
Interaction with Real Nodes

- Real nodes are represented by proxies in the simulation
- Real nodes can be placed anywhere in the simulation topology

Sensor Reports are transmitted to the applications running on the simulated nodes.
Middleware for WSN

- Often there is a gap between the applications and network protocols in wireless networks
- Adaptation functions are needed to satisfy the requirements of special features of WSNs and diversity of potential applications

- What is it?
  - A software infrastructure that glues together the network hardware, operating systems, network stacks, and applications.

- Role
  - Provide standardized system services to diverse applications.
  - Provide a runtime environment that can support and coordinate multiple applications.
  - Provide mechanisms to achieve adaptive and efficient utilization of system resources.
In other words

**MW is:**

- “The glue which connects objects which are distributed across multiple heterogeneous computer systems”
- “An extension of the operating system which provides a transparent communication layer to the applications”
- “A software layer that serves to shield the application of the heterogeneity of the underlying computer platforms and networks”
- Intended to hide low level details of hardware, networks, and distribution
Challenges in designing middleware.

- **Limited power and resources.**
  - Advance microelectronics technology allows tiny devices but limited in energy and resources, i.e. CPU and memory.
  - Middleware should provide mechanisms for efficient processor and memory use while enabling lower-power communication and prolong sensor node lifetime. i.e. sleep mode, minimize number of transmission.

- **Scalability, mobility, and dynamic network topology.**
  - As the application grows, device failure, moving obstacles, mobility, and interference, the network will change frequently.
  - Middleware should maintain performance and robust operation while network changes dynamically. Also, It should support mechanisms for fault tolerance and sensor node self-configuration and self-maintenance.

- **Heterogeneity.**
  - CPU-power, networking, memory and storage, operating systems.
  - Middleware should able to interface various kinds of hardware, software and networks.

- **Dynamic network organization.**
  - In most situation, Client/Server mode is impossible, no infrastructure is not available.
  - Middleware should support Ad-hoc capability to discover resource and its location which affect the trade-offs among latency, reliability, and energy.
Challenges in designing middleware.

- **Real-world integration.**
  - Most of the applications are dealing with real-time phenomena.
  - Middleware should provide real-time services.

- **Application knowledge.**
  - Developer would like to inject application knowledge to the network so as to map application communication requirements to it,
  - Middleware design should balance between application specificity and middleware generality.

- **Data aggregation.**
  - Network generate lots of redundant data, communications cost is much higher than computational cost. Sending a single bit can consume the same energy as executing 1000 instructions
  - Middleware should able to aggregate data to eliminate redundancy and minimize the number of transmissions to the sink.

- **Security.**
  - Middleware efforts should concentrate on developing and integrating security in the initial phases of software design, hence achieving different security requirements such as authentication, integrity, freshness, and availability.
Central abstraction: Virtual sensors

- A virtual sensor can be **any kind of data producer**
  - a real sensor, a wireless camera, a desktop computer, etc.

- **Abstract from implementation details**
  - physical sensors
  - a combination of other virtual sensors
  - 1 virtual sensor = $n$ input data streams + processing + 1 output data stream

- **Specification**
  - **metadata** (identification, data, type, location)
  - **structure** and properties of input and output streams
  - **declarative** SQL-based specification of the data stream processing
  - **functional properties** related to stream quality management, persistency, error handling, life-cycle management, and physical deployment.
MW- Sensor Nodes

- ARM Microcontroller
- Temperature Sensor
- 68HC11 Microcontroller
- Pressure Sensor
- 68HC11 µC
- No Sensor
The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks

http://lsirpeople.epfl.ch/aberer/PAPERS/LSIR%202006-006.pdf

<table>
<thead>
<tr>
<th>GSN node architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrity service</strong></td>
</tr>
<tr>
<td><strong>Access control</strong></td>
</tr>
<tr>
<td><strong>GSN/Web/Web-Services Interfaces</strong></td>
</tr>
<tr>
<td><strong>Notification Manager</strong></td>
</tr>
<tr>
<td><strong>Query Processor</strong></td>
</tr>
<tr>
<td><strong>Query Repository</strong></td>
</tr>
<tr>
<td><strong>Storage</strong></td>
</tr>
<tr>
<td><strong>Virtual Sensor Manager</strong></td>
</tr>
<tr>
<td><strong>Life Cycle Manager</strong></td>
</tr>
<tr>
<td><strong>Input Stream Manager</strong></td>
</tr>
<tr>
<td><strong>Stream Quality Manager</strong></td>
</tr>
<tr>
<td><strong>Pool of Virtual Sensors</strong></td>
</tr>
</tbody>
</table>

- Each GSN node hosts a number of virtual sensors
Each GSN node hosts a number of virtual sensors

**Virtual sensor manager**
- provides access to the virtual sensors
- manages the delivery of sensor data

**Life-cycle manager**
- provides and manages the resources provided to a virtual sensor
- manages the interactions with a virtual sensor
- ensures stream quality
- manages the life-cycle of sensors
Each GSN node hosts a number of virtual sensors

**Virtual sensor manager**
- provides access to the virtual sensors
- manages the delivery of sensor data

**Life-cycle manager**
- provides and manages the resources provided to a virtual sensor
- manages the interactions with a virtual sensor
- ensures stream quality
- manages the life-cycle of sensors

**Storage layer**
- persistent storage for data streams
### GSN node architecture

- Each GSN node hosts a number of virtual sensors
- **Virtual sensor manager**
  - provides access to the virtual sensors
  - manages the delivery of sensor data
- **Life-cycle manager**
  - provides and manages the resources provided to a virtual sensor
  - manages the interactions with a virtual sensor
  - ensures stream quality
  - manages the life-cycle of sensors
- **Storage layer**
  - persistent storage for data streams
- **Query manager**
  - manages active queries
  - query processing
  - delivery of events and query results to registered, local or remote consumers
### GSN node architecture

- **Each GSN node hosts a number of virtual sensors**
- **Virtual sensor manager**
  - provides access to the virtual sensors
  - manages the delivery of sensor data
- **Life-cycle manager**
  - provides and manages the resources provided to a virtual sensor
  - manages the interactions with a virtual sensor
  - ensures stream quality
  - manages the life-cycle of sensors
- **Storage layer**
  - persistent storage for data streams
- **Query manager**
  - manages active queries
  - query processing
  - delivery of events and query results to registered, local or remote consumers
- **Top layers**: access, access control, and integrity
Design principles

- Localized algorithms
- Distributed algorithms that achieve a global goal by local (neighbor) coordination
- Adaptive fidelity algorithms
  - Trade the quality of the result against resource usage (key element for resource efficiency)
- Data-centric communication
  - Addressing node by focusing on the data produced on the node
- Application knowledge
  - For data caching and aggregation
Middleware approaches for WSN.

- Virtual Machine (Cluster-Based)
  - Because of its similarity to the virtual machine concept in traditional distributed systems in terms of providing application semantic transparency from the physical infrastructure.
    
    Pros:
    - Common abstraction.
    - Sand-boxing.
    
    Cons:
    - High overhead
    - Difficult to exploit heterogeneity.

- Example 1 - Maté:
  - Power-centric abstraction.
  - Tied to TinyOS.
  - Broken up into 24 byte-long instruction capsules, easy for distribution.
  - Provide simple programming interface to sensor nodes. i.e. 6 instructions only for sense and send program.
  - Communication in synchronous, less complex in programming.
  - No support for message buffering / large storage.
Middleware approaches for WSN.

- Mobile Agents (Modular programming)
  - Pros:
    - Only parts of the program need to be updated, propagate efficiently.
  - Cons:
    - High overhead, Doesn’t allow hardware heterogeneity.

- Example:
  - IMPALA
    - As modular as possible, efficiency of updates and support dynamic applications.
    - The nature of its code instruction doesn't allow hardware heterogeneity.
    - Application Adaption with different profiles possible. (energy efficient)
    - Use in the ZebraNet project (wildlife monitoring).
Middleware approaches for WSN.

- **Database**
  - **Pros:**
    - Entire sensor network is abstracted as a virtual relational database. Ease to interoperate with existing systems.
  - **Cons:**
    - Doesn’t support real-time applications, and provides only approximate results.

- **Example:**
  - **Cougar**
    - Represents all sensors and sensor data in a relational database.
    - Control of sensors and extracting data occurs through special SQL-like queries.
    - Allows the scheduling of ongoing queries that provide incremental results.
    - Decentralized Implementation, message passing based on controlled flooding.
  - **SINA (System Information Networking Architecture)**
    - Based on a spreadsheet database, wherein network is a collection of data-sheets and cells are attributes.
    - Attribute-based naming, e.g. [type=temperature, location=N-E, temperature=50].
    - Queries again performed in an SQL-like language.
    - Decentralized Implementation based on clustering.
Middleware approaches for WSN.

- **Message Oriented**
  - **Pros:**
    - Use Publish-subscribe to support asynchronous communication, allowing a loose coupling between the sender and the receiver
  - **Cons:**
    - Overhead

- **Example:**
  - **MIRES**
    - Publish-subscribe
    - Multi-Hop Routing
    - Additional Service (e.g. data Aggregation)
    - Sense – advertise over P/S and route to Sink.