Bundle: A Group Based Programming Abstraction for Cyber Physical Systems ICCPS

Pascal A. Vicaire
Enamul Hoque
Zhiheng Xie
John A. Stankovic

Computer Science Department
University of Virginia

Presented By: Amara Naas
CSD at ODU
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
Introduction

Vision of CPSs:

- heterogeneous sensing and actuation devices
- Multiple applications can be executed simultaneously, and can be accessible and controllable via the Internet.
- presence of mobility
Outline

- Introduction
- **Motivation**
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
Across Network Concurrent Applications

Tracking

Temperature Regulation

Home Network

University Network
Dynamic Membership Update
Dynamic Membership Update
(Environmental Change)
Dynamic Membership Update (Mobility)

Home Network

University Network
Dynamic Membership Update (Mobility)

Home Network

University Network
Heterogeneity

Actuator is a key component in Cyber Physical systems

Home Network

University Network
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
Contributions

- The design, implementation and evaluation of a centralized group based abstraction named bundle that:
  - Allows concise programming of applications (in Java) using across network sensors and actuators
  - Dynamically updates group membership in response to intra and inter network mobility, and environmental changes
  - Relieves the programmer of CPU and memory management on resource constrained devices
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
## Related Work

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Hood</th>
<th>Abstract Region</th>
<th>Logical Neighborhood</th>
<th>Scope</th>
<th>Bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Across Network</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actuators</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heterogeneous Devices</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic Membership Update</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centralized</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
Underlying Architecture: Physicalnet – 4 Tier SOA

**Why Centralized Design:**
- Maintaining application logic in resource constrained service providers is expensive.
- The centralized gateway works as a communicator for heterogeneous service providers.
- To support inter-network mobility.

**Disadvantages:**
- In-network aggregation not possible
- Increased communication
The Bundle Programming Abstraction

- Bundle = dynamic, logical set of services
  - Definition
  - Specification of what the members should do

```java
public class MyApplication extends Application{
    public MyApplication(){
        this.add(new Negotiator(HOST,PORT,USER,PASSWORD);
        this.execute(1000/*milliseconds*/);

        new Bundle<Light>(Light.class, this){
            public boolean rule(Light l){
                return true;
            }
            public void foreach(Light l){
                l.on.set(true);
            }
        }
    }
}
```

Connect to a negotiator

Create a bundle of light actuators

Bundle of ALL the lights

All the lights must be turned on
Complex Bundle Examples

Create a bundle of temperature sensors that have adequate energy

```java
new Bundle<Temp>(Temp.class, this){
    public boolean rule(Temp t){
        Energy e = t.provider().getService(Energy.class);
        return !(e != null &&
            e.sense.getLastSample() != null &&
            e.sense.getLastSample() >= ENERGY_THRESHOLD);
    }
}
```

- Considers each temperature sensor as a possible member of the Bundle
- Gets the available energy of each temperature sensor
- Includes a temperature sensor in a Bundle only if it has sufficient energy
Complex Bundle Examples

A Surveillance Application

```java
final PhotoBundle temps=new PhotoBundle(this){
    public boolean rule(MyPhoto t){
        return room.contains(t);
    }
    public void forEach(final MyPhoto t){
        t.period.set(1000l);
        t.sense.set(true);
    }
};
```

- Creates a Bundle of light sensors located in a specific room.

- The members of the Bundle senses light intensity.

```java
final SounderBundle sounders=new SounderBundle(this){
    public boolean rule(Sounder s){
        return room.contains(s) &&
        temps.getAverage()!=-1 &&
        temps.getAverage() > LIGHT_THRESHOLD;
    }
    public void forEach(Sounder s){
        s.on.set(true);
    }
};
```

- Creates a Bundle of sounders in that specific room if the light intensity is more than a threshold.

- The sounders are turned on.
Create a Bundle of Temperature Sensors

Application connects to Negotiator and creates a Bundle

<table>
<thead>
<tr>
<th>id</th>
<th>Gateway</th>
<th>loc</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>a1</td>
<td>temp</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>a1</td>
<td>temp</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>a2</td>
<td>motion</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>a2</td>
<td>motion</td>
</tr>
</tbody>
</table>

Temperature sensor
Motion sensor

Gateway 1
Network 1

Gateway 2
Network 2
Application connects to Negotiator and creates a Bundle

| id | Gateway | loc | type |...
|----|---------|-----|------|...
| 1  | 1       | a1  | temp |...
| 2  | 2       | a2  | temp |...
| 3  | 2       | a2  | motion |...
| 4  | 2       | a2  | motion |...
Dynamic Membership Update

Rule: all the nodes that are in the same room as service X
Dynamic Membership Update

Rule: all the nodes that are in the same room as service X
Dynamic Membership Update

- The bundle membership is dynamically updated
- The latest values of the variables involved in the membership are considered
- New and leaving members are automatically reconfigured
Dynamic Membership Update

- The bundle membership is dynamically updated.
- The latest values of the variables involved in the membership are considered.
- New and leaving members are automatically reconfigured.
Dynamic Membership Update

- The bundle membership is dynamically updated

- The latest values of the variables involved in the membership are considered

- New and leaving members are automatically reconfigured
Actuators

- Actuator states can be complex:
  - Pan and zoom level of a camera
  - Light intensity level
  - Speed of a motor

---

**Application**
(Turn light 1 on)

**Negotiator**

<table>
<thead>
<tr>
<th>id</th>
<th>gateway</th>
<th>loc</th>
<th>type</th>
<th>req</th>
<th>state</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>a1</td>
<td>light</td>
<td>on</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

---

**Gateway 1**

- on
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- **Evaluation**
- Conclusion
Evaluation

- It is difficult to evaluate a programming abstraction
- They have implemented 32 applications
- All applications can handle: node mobility, robust actuator configuration, multiple networks, heterogeneity
- Applications tested and/or emulated
- They evaluate the conciseness of Bundles and energy efficiency
Tracker

- TinyOS service providers seamlessly interact with Java service providers

- User moves around with 2 MICAz motes (mediaTag and lightTag)

  - If mediaTag on, turn television on in the room
    - If no television, turn music player on

  - If lightTag on, turn all the lights on

- Television and music player emulated by a java service provider in laptop

- 30 lines
NeighborhoodWatch

- Multimodal sensing across networks
- Each of a set of neighbors wear a MICAz tag
- If a house is empty, all the accelerometers and light sensors are turned on
- The sounders of the MICAz tags of all neighbors are turned on, if there is an intruder in any house
- 56 lines
Evaluation: Conciseness

- Each of the 32 applications is programmed in less than 60 lines of code

- They implemented one application *ParkingSpaceFinder* in nesC which took 42 lines

- Using Bundles it took 20 lines
Evaluation: Energy Efficiency

- Comparison with VigilNet for a target tracking application by simulation

- XSM platform and its empirical power consumption model

- 10,000 nodes randomly placed in a square of edge 1000 meters

- One base station (gateway) for every 100 nodes
Evaluation: Energy Efficiency

Detection Probability: Percentage of the targets that are successfully detected
Outline

- Introduction
- Motivation
- Contributions
- Related work
- Bundle Design
- Evaluation
- Conclusion
Conclusion

 Contribution: the design, implementation, and evaluation of the bundle programming abstraction that eases programming by:

- Abstracting away networking protocols and cross-network application deployment

- Automatically and robustly reconfiguring nodes according to network dynamics

- Relieving the programmer of CPU and memory management on WNSA nodes
Questions?
Thank You