

SASSI: Sliverware Architecture for Sensor Systems Integration

Seth Holloway, Alexander Griffith, Angela Dalton, Drew Stovall, and Christine Julien
The University of Texas at Austin

Development challenges for Embedded Sensor Applications:

- Sensor network applications are typically written at low-level of abstraction
 - Difficult for developers to rapidly deploy new applications
 - Impossible to enable *domain programmers* who are experts in particular application domains
- Applications are largely independent of each other
 - Redundancy in communication, computation, and sensing is very expensive

The Sliverware approach:

- Provides a generic programming framework for collaborative applications
 - Simplifies application development
 - Focuses developers on high-level functions
- Entails lightweight, cross-cutting, vertical abstractions
 - Separates sensing and communicating components from applications
 - Promotes abstraction and reuse

A SASSI Instantiation: The Holistic Home

Slivers define collaboration components:

- A single sliver provides a single high-level collaboration function
 - e.g., presence information about a home's occupants
- The sliver encompasses collaboration (i.e., sensing) functions, grouping functions, and communication

Combining slivers into applications:

- A single application is often more complex than a single sliver
 - An application developer may require multiple collaborative activities (e.g., presence information and temperature events to open and close air conditioning vents)
- Combining slivers makes applications highly modular

Three sliver layers:

- Collaborative service:
 - e.g., location service
 - Hides complex, low-level coordination function in abstract *events*
- Group membership:
 - e.g., house presence group
 - Coordinates related groups of devices based on application-specified policies
- Network communication:
 - e.g., RFID
 - Provides communication capabilities that adhere to Sliverware-provided interface

Application integration:

- SASSI allows multiple applications to be integrated
 - In embedded sensor applications, this allows a unified interface to a wide set of functionality
 - e.g., applications to control a home

Sliverware programmability:

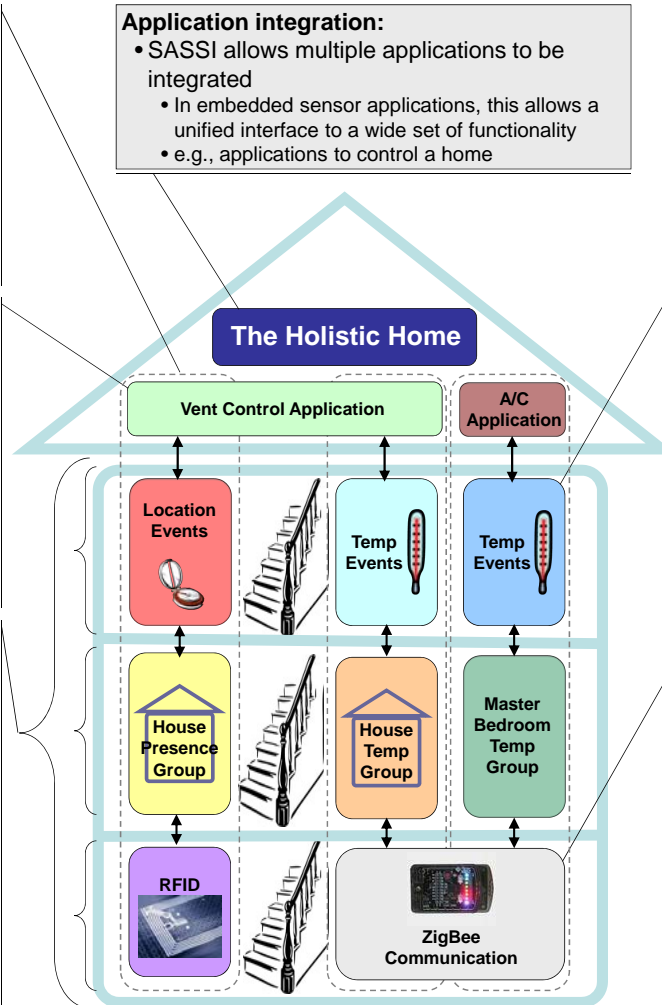
- Developers tailor slivers and SASSI applications by:
 - Creating new applications using existing slivers
 - Creating new slivers by combining existing sliver components
 - Introducing new low-level capabilities by defining new sliver components

Sliver component upgradability:

- Sensing and communication technologies are rapidly changing
 - New technologies often differ significantly from previous generations
- Components are separated by crisply defined interfaces
 - At any layer, a new implementation of a sliver component can be swapped in for an outdated component
 - With no impact on application functionality (but with potential benefits to non-functional characteristics)

Redundancy handled within Sliverware infrastructure:

- Embedded sensor applications typically operate independently
 - They incur redundancy in communication, sensing, etc.
- Sliverware infrastructure merges slivers bottom-up
 - Combines functionality that is redundant across seemingly independent applications
 - e.g., reuse already deployed communication protocols or previously defined group



Why SASSI?

- Simplify development of embedded sensor applications
 - Decrease level of complexity by increasing the degree of abstraction
 - Shelter programmers from the complexity of directly handling sensing and communication tasks
- Separate communication and sensing concerns from implementation of application behavior
- Provide container for deploying new application components
- Enable application integration
 - Combine previously deployed applications into amalgams of complex behavior

SASSI Benefits

- Provides component reusability on-the-fly
 - SASSI infrastructure detects and eliminates redundancy among simultaneously deployed slivers
- Enables easy upgradability and maintainability
 - New slivers can be inserted at any time
 - New sliver components can be created and injected into slivers without affecting existing application behavior
- Allows rapid application deployment at varying levels of expertise
 - *Domain programmers* can create applications out of existing slivers, while more experienced programmers can create new slivers and sliver components