

System-Level Programming Abstractions for Ubiquitous Computing

Eric Lemar,
Robert Grimm, Janet Davis, Adam MacBeth,
Steven Swanson, Steven Gribble, Tom
Anderson, Brian Bershad, Gaetano Borriello,
David Wetherall

University of Washington

Problem

- Building pervasive applications is too hard for the average hacker
 - Devices roam
 - Users switch devices
 - Networks provide limited services



This Is A Systems Problem!

- Need dedicated systems support to make programmers' task feasible
- But existing approaches to building distributed systems are not suitable
 - Extend single-node programming models
 - Designed for smaller, less dynamic environments

Programming for Change

- Pervasive computing requires a new way of thinking.

“No Application is an island”

An application's runtime environment

- May change frequently
- May be changed by others

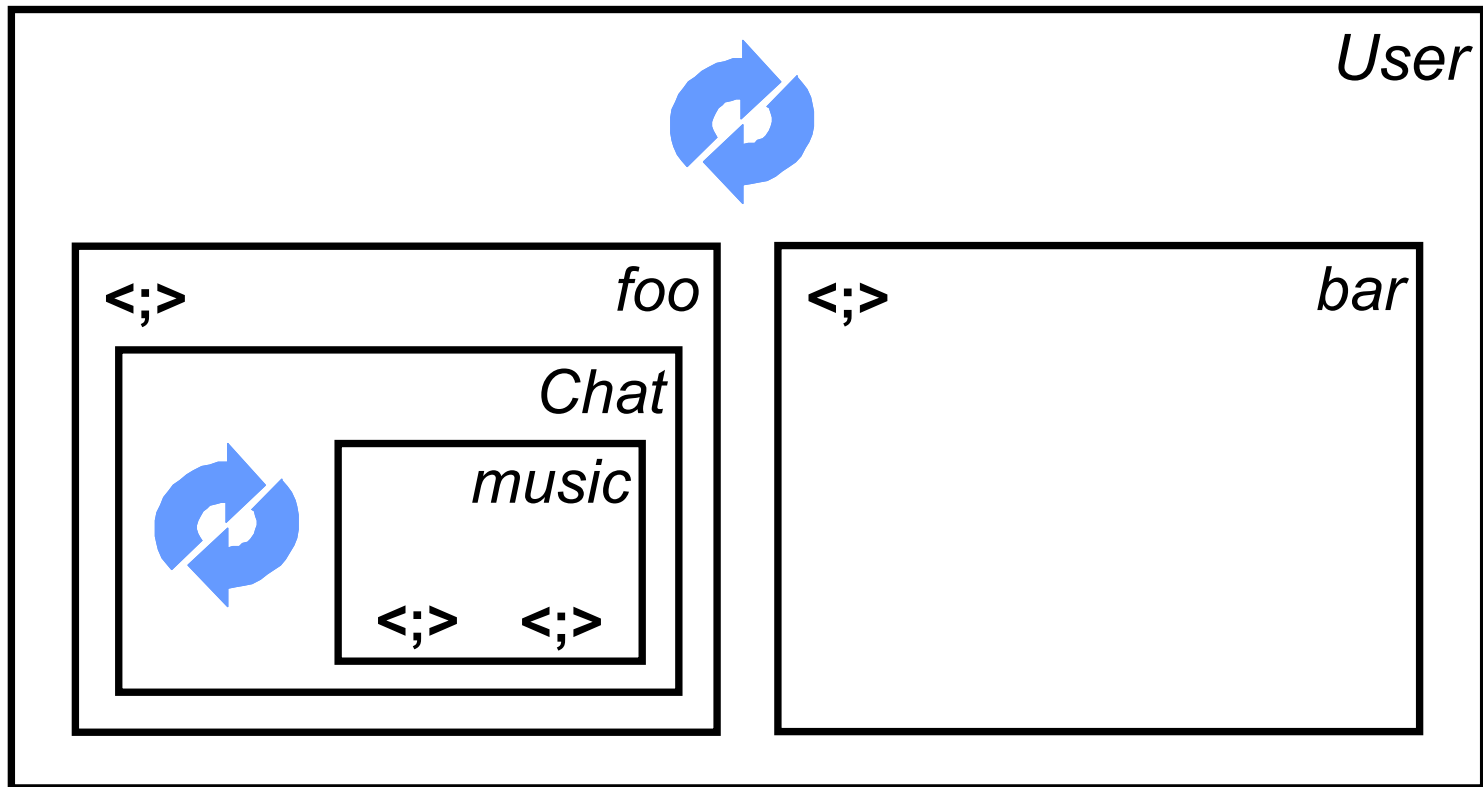
Approach

- Expose change to applications
 - Change is an expected and desirable property in a pervasive system
 - Application must be involved in reacting to change
- Provide systems support to make developers' task feasible
 - Need architecture that provides primitives for coping with constant change

Basic Abstractions

- Tuples
- Components
 - Exchange asynchronous events
- Environments
 - Group tuples, components, and nested environments to simplify application management

Environment Hierarchy



 Environment  Tuple  Components

Toolbox for Change

- Checkpointing
- Migration
- Remote event passing
- Discovery
- Operation

Checkpointing

- Captures and restores application state
 - Operates on all components in an environment tree
 - Aids in recovery from device failure
 - Stores the checkpoint as a tuple in the environment

Migration

- Moves or copies an application and its data
 - Operates on an environment, moving
 - Stored tuples
 - Components
 - Nested Environments
 - Resources outside the tree are not migrated
 - Application is responsible for reacquiring external resources

Remote Event Passing

- Services export event handlers under symbolic descriptors (tuples)
 - Resulting bindings are leased
 - Clients send events by specifying symbolic receiver

Discovery

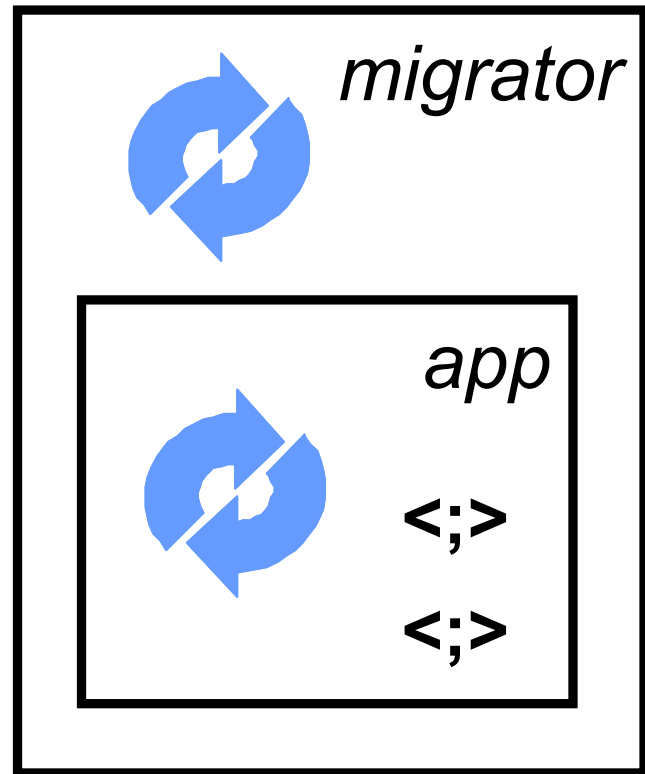
- Sends events to unknown physical destinations
 - Includes support for early and late binding, broadcast
 - Relies on discovery server elected from local nodes
 - Integrated with remote event passing

Logic/Operation Pattern

- Asynchronous, unreliable events
 - Are natural in a pervasive environment
 - But dealing with all cases is too complex
- Easing the burden: the Logic/Operation pattern
 - Logic: computations that do not fail
 - Operations: interactions that may fail
 - Implementation provides automatic timeouts and retries

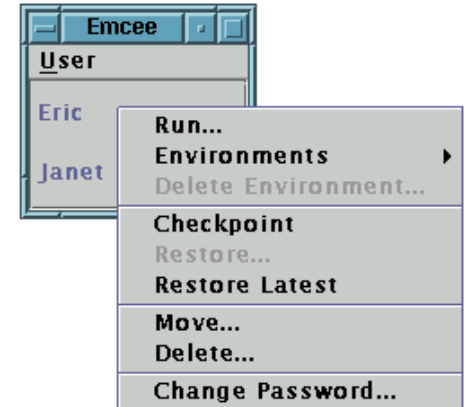
Example of Composition

- Composing for mobility
 - Root of tree controls
 - When to migrate
 - Where to migrate to
 - Isolates migration logic from application logic

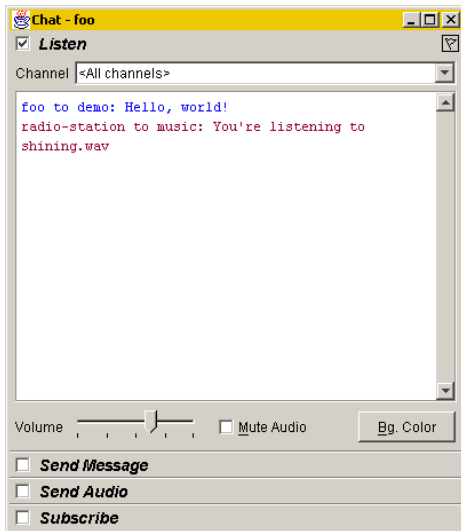


Application Example: *Emcee*

- An agent of change
- Manages users and their applications using nesting:
 - /User/<user-name>/<application>
- Exploits nesting to
 - Checkpoint user state
 - Migrate applications between users
 - Migrate users between nodes
- Uses discovery to locate users in the local area



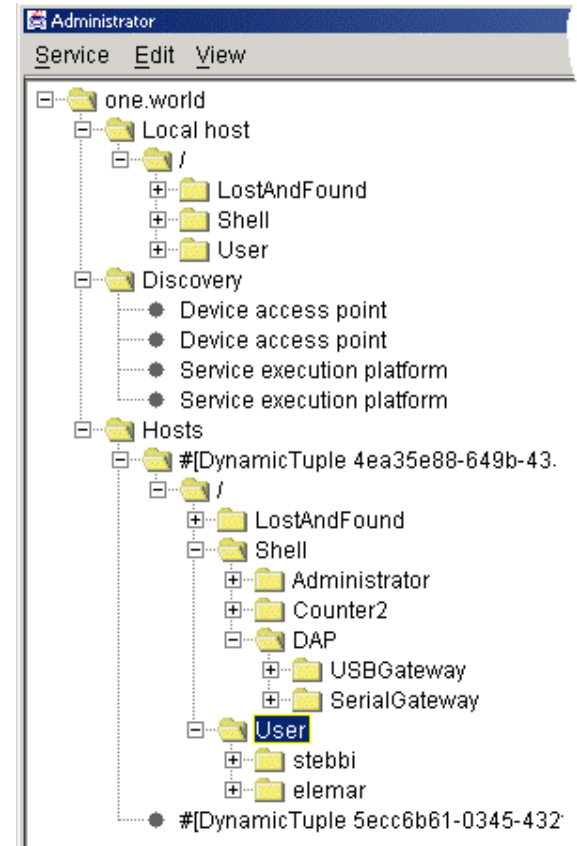
Application Example: *one.radio*



- Provides text and audio messaging
 - Messages and audio delivered to “channels”
 - Routes messages through discovery
 - By programming to one.world, supports migration
- Explicitly handles change
 - Disables audio features if hardware is unavailable
 - When moved or restored, checks that the user is still the same
 - Exports its handlers to discovery

Labscape

- Project of Larry Arnstein and others at the University of Washington
- Data collection in a biology lab
 - Collect and display data as scientist moves around and beyond the lab
 - Ease the installation of new hardware
 - Lower management overhead



Summary

- Pervasive applications require dedicated systems support that
 - Exposes change
 - Provides services which help react to and exploit change
- *one.world* provides this support in the form of
 - Unified treatment of data and applications
 - Application mobility
 - Flexible communications primitives
 - Encouraging a programming style which expects change

More information and a source release are
available at:

<http://one.cs.washington.edu>