

# one.world Programming for Pervasive Computing Environments

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

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

## Problem

- **Building pervasive applications is too hard**
  - Devices roam, users switch devices, network may provide only limited services
- **Existing systems are designed for less dynamic environments**
  - Extend single-node programming models and hide distribution

## Approach

- **Expose change to applications**
  - Developers decide how applications react to change
- **Provide systems support to make developers' task feasible**
  - Integrated architecture provides the right primitives for coping with constant change

## Basic Abstractions

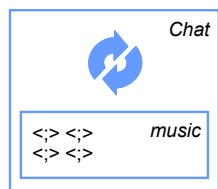
- **Tuples**
  - Provide common data format to simplify sharing and querying of data
  - Used for storage, networking, events
- **Components**
  - Exchange asynchronous events
- **Environments**
  - Group tuples, components, environments to simplify application management

## Developers' Toolbox

- **Manages asynchronous interactions**
  - Based on logic/operation pattern
    - Logic: Computations that do not fail
    - Operations: Interactions that may fail
  - Provides automatic timeouts and retries
  - Supports composition
- **Operation**
- **Checkpoints**
  - Captures and restores an application's execution state
  - Operates on all components in an environment tree
    - Quiesces threads executing event handlers
    - Serializes each environment's components
  - Stores checkpoint as a tuple in root of tree
- **Migration**
  - Moves or copies an application and its data
    - Operates on an environment tree
      - Stored tuples
      - Components
      - Nested environments
    - But nothing outside the tree
      - Breaks bindings to outside
      - Must be restored by application
- **Discovery**
  - Sends events to services with unknown location
    - Includes support for early and late binding, multicast
    - Relies on discovery server elected from local nodes
    - Integrated with remote event passing
- **Remote event passing**
  - Sends events to remote services
    - Services export event handlers under symbolic descriptors (tuples)
      - Resulting bindings are leased
    - Clients send events by specifying symbolic receiver

## Programming for Change

- **"No application is an island"**
  - An application's runtime environment
    - May change frequently
    - May be changed by others
- **Example application: Emcee**
  - Manages users and their applications
  - Structures environment hierarchy
    - / User / <user-name> / <application>
  - Exploits environment nesting for
    - Checkpointing a user's applications
    - Migrating applications between users
    - Migrating users between nodes



Chat with music stored in a nested environment

- **Example application: Chat**
  - Provides text and audio messaging
  - Uses discovery for automatic message routing
    - Locates participants in face of migration
  - Explicitly handles change
    - After activation, restoration, migration
      - Verifies user is the same
      - Exports handlers for subscribed channels
  - Silences audio if hardware/music is unavailable
  - Checks for concurrent termination before handling events (including chat messages)



Chat's main window