Systems Directions for Pervasive Computing

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

University of Washington

The Vision

- Make computers usable for (computer) illiterate
  - Enabled by ubiquitous smart devices
- Implies a shift in focus
  - Away from devices and technology
  - Towards users and their tasks

The Reality

- Hardware is almost there
  - Handhelds, tablets, cars, fridges, dogs
  - Wireless networking
  - Location sensing
- Applications are missing
  - Too hard to design, build, and deploy in a giant, ad-hoc distributed system
    - Stuck with email and WWW

The Challenge

- Average hacker needs to develop applications that
  - Adapt to a changing environment
  - Work even if
    - Devices are roaming
    - Users switch devices
    - Network provides only limited services, or none at all

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

What To Do?

- Smash up and replace
  - Invoke Dawson Engler, abstraction exterminator
  - Replace with simpler, more suitable abstractions
- Pile on
  - Provide abstractions/primitives that directly help programmers
    - “Checkpoint” and “restore”
    - “Move to remote node”
    - “Find matching resource”

Smash up

Distributed Objects

- Popular in distributed systems (Legion, Globe)
- Difficult to evolve
  - Data formats set by standard bodies
  - Vendors compete on functionality
    - Many interfaces and implementations that change faster than data formats
- Difficult to control
  - Easier to provide security and resource controls for passive data

Replace

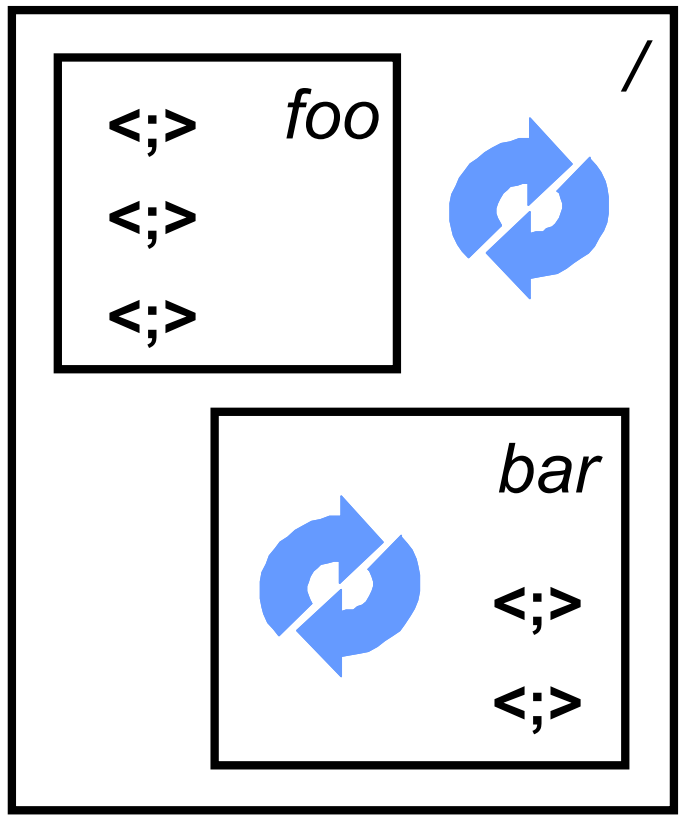
Separate Data and Functionality

- Tuples represent data
  - Are self-describing records
  - Define common data model, type system
  - Enable a common query language
- Components implement functionality
  - Export and import event handlers
  - Are distributed in a common binary format
  - Rely on a common core API
- But data and functionality depend on each other
  - Migrating an application and its data

Pile on

Environments

- Serve as containers for
  - Tuples
  - Components
  - Environments
- Preserve independent access
- Represent a combination of
  - File system directories
  - Nested processes



Smash up

Transparent Distribution Is Harmful

- Hides access to remote resources
  - Distributed file systems
  - Remote procedure calls
- Treats failures or unavailability as an extreme case
- But frequent changes are inherent to pervasive computing environments
  - Result: limited application availability

Replace

Expose Change

- Applications need to acquire all resources and be able to reacquire them at any time
  - Explicitly bind resources
  - Use leases to provide timeouts when accessing unavailable resources
- Programming for change shifts burden to application developer
  - Can we do better?

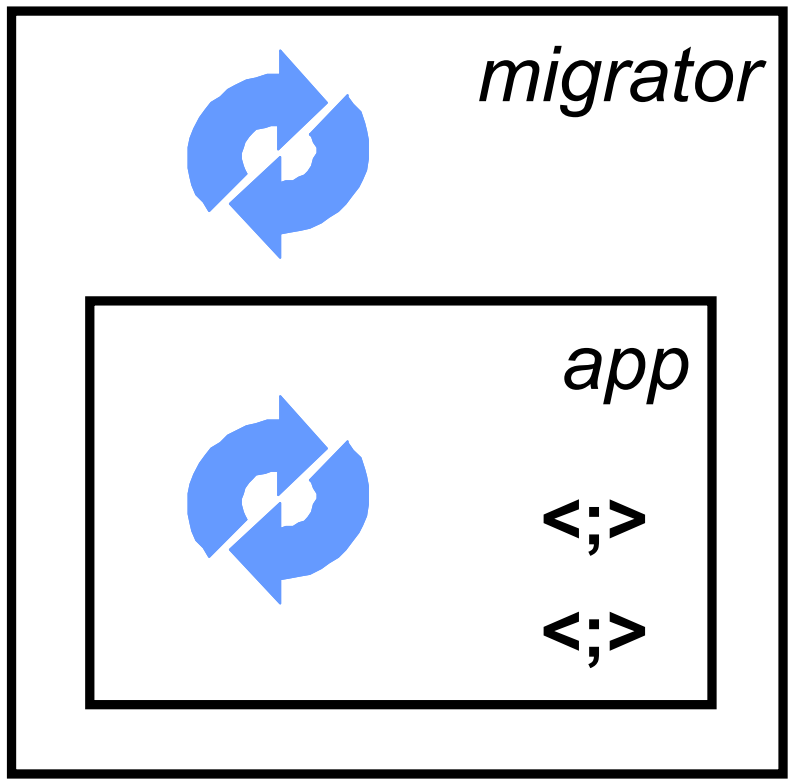
Pile on

Say Yes To Migration

- Moves/copies an application and its data
- Affects an entire environment tree
  - Tuples
  - Components
  - Environments
  - But nothing outside the tree
    - Breaks bindings to outside
- Makes migration in the wide area feasible

Putting It All Together

- Root of tree controls
  - When to migrate
  - Where to migrate
- Compose for migration
  - Isolate migration logic in separate environment
  - Embed application in that environment



Summary

- Challenge
  - Build applications that gracefully adapt to constant change
- Solution
  - Provide dedicated systems support
    - Separate data and functionality
    - Expose change to applications
    - Include primitives to cope with change

http://one.cs.washington.edu