Towards a system architecture for pervasive computing

Robert Grimm, Janet Davis, Ben Hendrickson, Eric Lemar, Tom Anderson, Brian Bershad, Gaetano Borriello, David Wetherall
University of Washington

http://one.cs.washington.edu

Pervasive Computing

- **Vision**
  - Focus on users and their tasks
  - Enabled by ubiquitous smart devices
  - Simple example: giving a talk
    - Prefetch and install latest slides and presentation application
    - Discover A/V devices and connect to them
    - Capture discussion
- **Reality**
  - Hardware is almost there
  - Applications are missing

Problem

- **Too hard to build and deploy applications**
  - Applications need to
    - Run across the range of devices
    - Provide service when connectivity is limited or intermittent
    - Preserve the security and privacy of all participants
    - Do all this on a global scale
  - Fundamentally, a systems problem!
  - Rethink OS abstractions

Approach

- **Common system architecture**
  - Targeted at application developers and administrators
- **Support for mobile code**
  - Move functionality to where it is needed
- **Support for data management**
  - Make data self-describing
  - Make data available where it is needed
- **Keep code and data separate**
  - A loosely coupled system with easy access to data from code
  - Not a distributed object system

Features

- **Structured I/O**
  - Preserve structure of application data
  - Data represented as tuples
  - Strongly typed records
  - Common interface to storage and communications
    - Atomic read and write operations
    - Storage only
    - Transactions
    - Searches

- **Asynchronous events**
  - Scale better than threads
  - Make execution state explicit
  - Provide control over scheduling
  - Have been used successfully across a wide range of other systems:
    - Tiny OS
    - Palm OS
    - Chinook
    - Windows
    - Ninja

- **Encapsulation**
  - Control storage and computations
  - Hierarchical environments
    - Containers for stored tuples, active computations, and other environments
    - Provide isolation and resource controls

- **Integration of mobile code with storage**
  - Mobile code needs local storage
  - Code, like data, stored in environments
  - Computations can be check-pointed
  - Environments can be moved between nodes

- **Dynamic typing and components**
  - Easier to exchange data than to compose traditional interfaces
  - Dynamic tuples: fields declared and typed dynamically
  - Uniform event handler interface
  - Components
    - Export and import event handlers
    - Dynamically linked

- **Public interface EventHandler**

  ```java
  public interface EventHandler {
    void handle(Event e);
  }
  ```

  The event handler interface

- **Status**
  - Finished design
  - Working on implementation in Java
  - Defined core interfaces
  - Building core services and default replication layer
  - First public release in November

An example environment hierarchy: A presentation application running in the outer environment, code, slides, audio, & video stored in nested environments.