one.world
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
Vision

• Shift in focus to users and their tasks
  – Enabled by ubiquitous smart devices
• 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
  - Processor, storage, networking
- Applications are missing
  - Too hard to design, build, and deploy
- Fundamentally, a systems problem
Problem: Immense Variability

- How to build and deploy applications that
  - Run across range of devices
  - Continue to provide service when connectivity is limited or intermittent
  - Preserve the security and privacy of all participants
  - All on a global scale
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
Approach (continued)

• Support mobile code and data management in a single architecture
• Yet, keep them separate
  – Not a distributed object system
  – Rather, a loosely coupled system with easy access to data from code
    • Data and code can evolve separately
    • Exchange of passive data easier than exchange of active objects
Engineering Challenge

• How to integrate the right technologies in the right way?
• We need support for
  – Data management
  – Controllability
  – Reliability and availability
  – Scalability
  – Composability

one.world
one.world

- Structured I/O
- Encapsulation
- Storage integrated with mobile code
- Asynchronous events
- Dynamic composition
- Discovery, leasing, transactions
Structured I/O

• Preserve structure of application data
  – Communication: application-level framing
  – Storage: effective searching and sharing
• Provide easily predictable operations
  – Generally atomic
  – Optionally transactional
Structured I/O

• Tuples
  – Strongly typed records
• Basic operations
  – Communications and storage
  – Put, read, listen
• Extended operations
  – Storage
  – Put, read, listen, query, delete
Structured I/O

• Data represented and accessed as tuples
• Slides
  – Fields: title, body, previous, next
• Audio
  – Broken into chunks
    • Description
    • Binary chunk

one.world
one.world

• Structured I/O
• Encapsulation
• Storage integrated with mobile code
• Asynchronous events
• Dynamic composition
Encapsuluation

• “Foreign” applications
  – Hosting
  – Software agents
• Robustness
  – Isolate active entities
• Controllability
  – Track and limit resource usage
Encapsulation

- Hierarchical environments
  - Containers
    - Stored tuples
    - Active computations
    - Other environments
  - Isolation for active computations
  - Resource controls
Encapsulation

• Outer environment
  – Application
• Nested environments
  – Slides
  – Audio
  – Video
• Limits on resources
  – Accessible resources
  – Consumed resources

one.world
one.world

- Structured I/O
- Encapsulation
- Storage integrated with mobile code
- Asynchronous events
- Dynamic composition
Storage & Mobile Code

- Node failures
- Disconnected operation
- Reliability and availability
  - Capture execution state
  - Make data and code available locally
    - Replicate it
    - Move it
Storage & Mobile Code

• Active computations
  – Checkpoint
• Environments
  – Move
• Code
  – Stored in environments
Storage & Mobile Code

• Nested environments
  – Include application code
  – Moved as one unit

one.world
one.world

- Structured I/O
- Encapsulation
- Storage integrated with mobile code
- Asynchronous events
- Dynamic composition
Asynchronous Events

• Make execution state explicit
• Provide control over scheduling
  – At a coarse grain: appointments
  – At a fine grain: multimedia
• Scale across devices
• Use asynchronous events instead of threads

TinyOS  Palm OS  Chinook  Windows  Mac OS  Ninja

one.world
Asynchronous Events

- Functionality expressed through events and event handlers
  - Events passed by value
- “Animators” provided by system
  - Event queue
  - One or more threads to process queue
    - One thread: Conventional event loop
Asynchronous Events

• Events
  – User input ("next slide")
  – Audio chunk
  – Video frame

• Event handlers
  – Presentation application
  – A/V devices
one.world

- Structured I/O
- Encapsulation
- Storage integrated with mobile code
- Asynchronous events
- Dynamic composition
Dynamic Composition

• Flexible glue
  – Rapid prototyping
  – Scripting languages
  – Graceful evolution
  • From prototype to production system
  • Through application revisions
Dynamic Composition

• Dynamic typing
  – Dynamic tuples
    • Fields declared and typed dynamically
  – Uniform event handling interface
    • public interface EventHandler {
      void handle(Event e);
    }
  • Remote communication
• Components
  – Export and import event handlers
  – Dynamically linked

one.world
Dynamic Composition

• Annotations to slides
• Data exchange between applications
• Interaction with room components
  – Projection system
  – Audio/video recording system
one.world

- Structured I/O
- Encapsulation
- Storage integrated with mobile code
- Asynchronous events
- Dynamic composition
Features

• Economy of mechanism
  – Tuples, event handlers, environments
  – Single API for communication and storage
• Range of typing options
  – Statically typed
  – Dynamically typed
Current Status

- Working on implementation in Java
  - Defined core interfaces
  - Building
    - Core services
    - Default replication layer
- Exploring applications
  - *digime*
Open Issues

• Integrated policies
  – Access control
  – Resource control
• Impact of real time
  – Granularities of time
    • At a coarse grain: appointments
    • At a fine grain: multimedia
• Validation
Opportunities

• System-level
  – Forced termination
  – Resource accounting
  – Key-based authentication and authorization
  – Transaction manager
  – Discovery service
  – Conduits to Internet services and protocols

• Applications
Conclusions

• Building pervasive applications is hard
• Need common software architecture
  – Structured I/O
  – Encapsulation
  – Storage integrated with mobile code
  – Asynchronous events
  – Dynamic composition
http://one.cs.washington.edu