

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

- Focus on users and their tasks
- Example: giving a talk at IBM Watson
 - Latest slides are “simply” projected
 - Slides prefetched
 - Application installed
 - Discussion is captured
 - Notes on top of slides
 - Audio / video recording

Reality

- Hardware is almost “there”
 - Processor, storage, networking
- Applications are missing
 - Too hard to design, build, and deploy

Immense Variability

- Devices
 - From wearables to super-computers
- Network connectivity
 - From IR to gigabit ethernet
 - Often intermittent
- Administration
 - Across many different domains

Problem

- 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

Jini

- Sun's "connection technology" on top of Java
- Traditional client-server model
- Mobile code
- Plug & play

Jini

[Waldo, *Computer*, 6/00]

- Infrastructure
 - Discovery / lookup
- Programming model
 - RMI
 - Leasing
 - Transactions
 - Distributed events
 - Distributed security
- Services
 - JavaSpaces
 - Transaction manager

Is Jini enough?

Challenge

- How to integrate the right technologies in the right way?
 - Programmability
 - How to compose parts?
 - Controllability
 - How to schedule activities?
 - How to limit resources?

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
 - Storage: effective searching and sharing
 - Communication: application-level framing
- Provide atomic operations
 - Optionally transactional

Structured I/O

- Tuples
 - Records with named fields
- Basic operations
 - Communications and storage
 - Put, read, listen
- Extended operations
 - Storage
 - Put, read, listen, write, delete, query, take

Structured I/O

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

one.world

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

Encapsulation

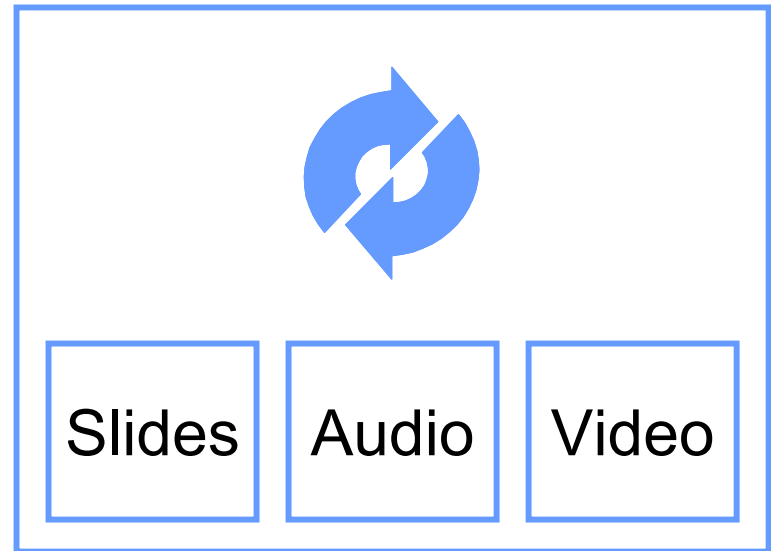
- “Foreign” applications
 - Hosting
 - Software agents
- Robustness
 - Isolate active entities
- Accountability and controllability
 - Track and limit resource usage

Encapsulation

- Tasks
 - Active entities
 - Isolated from each other
- Environments
 - Encapsulate tuples, tasks, environments
 - Hierarchical resource controls

Encapsulation

- Nested environments
 - Slides
 - Audio
 - Video
- Presentation application
 - Represented as a task
- Limits on resources
 - Accessible resources
 - Consumed resources



one.world

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

Storage & Mobile Code

- Node failures
- Disconnected operation
- Reliability
 - Capture execution state
 - Make data & code available locally
 - Replicate it
 - Move it

Storage & Mobile Code

- Tasks
 - Checkpoint
- Environments
 - Move
- Code
 - Stored in environments

Storage & Mobile Code

- Nested environments
 - Contain application code
 - Moved as one unit

one.world

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

Asynchronous Events

- Make time a first-class object
 - Scheduling of operations
 - At a fine grain: multimedia
 - At a coarse grain: appointments

Asynchronous Events

- Avoid threads
 - Concurrency control
 - Implicit state
 - Scalability
- Use asynchronous events



Asynchronous Events

- Uniform event handling interface

- ```
public interface EventHandler {
 void handle(Event e)
 throws EventDeliveryException;
}
```

- Event queues and thread-pools

- One thread: event loop
  - Multiple concurrent threads

# Asynchronous Events

---

- Events
  - User input (“next slide”)
  - Audio chunk
  - Video frame
- Scheduling through events

# *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

---

- Ad-hoc tuples
  - Simplified version of XML
- Uniform event handling interface
  - Remote communication
- Component model
  - Based on tuples and event handlers

# Dynamic Composition

---

- Annotations to slides
- Data exchange between applications
- Interaction with room components
  - Projection system
  - Audio / video recording system

# Features

---

- Economy of mechanism
  - Tuples, event handlers
  - Single API for communication and storage
- Range of typing options
  - Statically typed
  - Dynamically typed
- Separation of functionality from data

# Current Status

---

- Defined core interfaces
- Working on implementation
- Exploring applications
  - *digime*
  - Presentations / collaboration
  - Jukebox

# Open Issues

---

- Integrated policies
  - Access control
  - Resource control
- Impact of real time
  - Granularities of time

# 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>