Will Pervasive Computing be Manageable?

An Architectural View

Gaetano Borriello

University of Washington/CS&E

gaetano@cs.washington.edu
www.cs.washington.edu/homes/gaetano

What is Pervasive Computing?

- Technology view
  - Mobile portable devices
  - "Scrap" devices
  - Embedded sensors
  - Voice/vision/motion interfaces
- User view
  - Distraction-free, "invisible"
  - Minimal configuration
  - Augmenting human abilities in context of tasks

The Vision

- Make computers usable by those who don't really care to (or can't ever) become system administrators
  - enabled by ubiquitousity of smart devices, sensors, services, and infrastructure
- Implies a shift in focus
  - away from devices and technology
  - towards users and their goals

The Reality

- Hardware is almost there
  - handhelds, tablets, cars, fridges, dogs, . . .
  - wireless networking
  - location sensing
- Applications are slow in coming
  - too hard to design, build, debug, and deploy – a giant, ad-hoc distributed system
  - new abstractions are needed

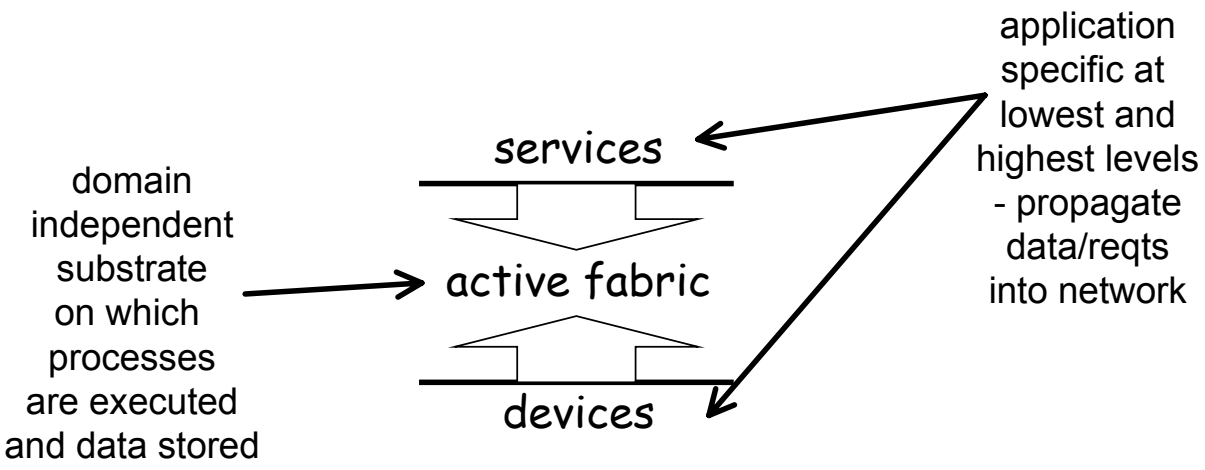
The Challenge

- Average programmer needs to develop applications that
  - adapt to a constantly changing environment
  - continue to work even if
    - devices are roaming
    - users switch devices
    - network provides only limited services
    - connectivity is intermittent

This Is A Systems Problem!

- Existing approaches to building distributed systems are not suitable
  - designed for smaller, closed, and less dynamic environments
- Need dedicated systems support to make programmers' task feasible
  - "checkpoint" and "restore"
  - "move to remote node"
  - "find matching resource"

In a nutshell



Events

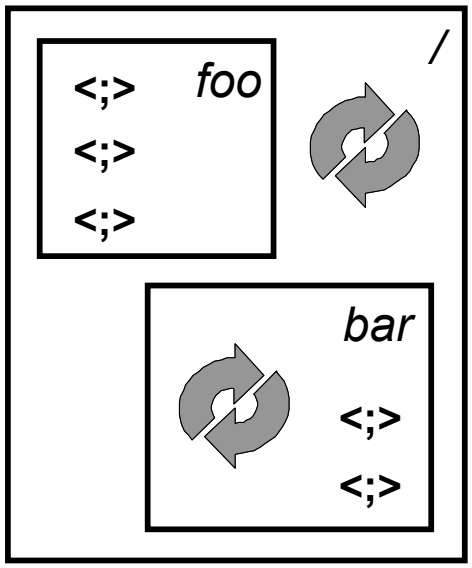
- Devices inject events into the network
  - they find their own way – active
  - discovery services
  - aggregation and correlation
- Database/tuplespace
  - common query language
  - publish/subscribe
  - passive data provides increased security
- Abstraction services
  - extraction of higher-order events
  - monitoring
  - redundancy

Separate Data and Functionality

- Tuplespace represents data
  - self-describing records
  - define common data model, type system
  - composability through queries
- Components implement functionality
  - export and import event handlers
  - rely on a common core API
  - bound to tuplespaces
- But data and functionality do depend on each other

Environments

- Containers for
  - tuples
  - components
  - other environments
- Represent a combination of
  - access privileges
  - file system directories
  - nested processes
- Fundamental unit of migration/replication



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
  - provide checkpoint/restore/migration primitives

Migration

- Moves/copies an application and its data
- Affects an entire environment tree
  - tuples
  - components
  - environments
  - but nothing outside the tree
- Make migration in the wide area feasible
  - environments negotiate to move – keep promises
  - moves entire operating environment
  - including security

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
- Web sites
  - portolano.cs.washington.edu
  - one.cs.washington.edu
  - labscape.cs.washington.edu

Questions to panel

- Will communication facilities be able to cope with the required bandwidth and give guaranteed quality of service for future mobile, multi-media traffic?
- Can we develop intelligent context-aware systems which can determine our activity and react appropriately eg to distinguish a heart problem from exertion due to running for a bus.
- How do mobile computers locate the required services from the local environment to form ad-hoc collaborative groups?
- How can we develop self-organising hardware and software architectures for pervasive computing?
- Is there a distinction between management and normal functionality for adaptive and self-organising systems?
- Will interaction techniques based on biological organisms be an appropriate solution for coherent behaviour from vast numbers of unreliable sensors, actuators and communication devices?
- How will personal-privacy be affected by constant monitoring and location tracking?
- Will current security and management mechanisms scale to cater for millions of mobile computers interacting with a pervasive computing environment.