Middleware for pervasive computing: A Survey

Vaskar Raychoudhury, Jiannong Cao, Mohan Kumar, Diqiang Zhang

Presented by: Yunus Basagalar
Pervasive computing
Pervasive computing

• Vast number of heterogeneous devices
• Huge amounts of data
• Highly dynamic environments
• No global view of resources
Middleware for pervasive computing

- Hide heterogeneity
- Hide complexity
- Ease resource management
- Ease programmer’s burden

Hardware and software resources: Operating systems, sensors etc.
Goal

• Design dimensions
• Common services
• Categorization
Design dimensions

- Programming Abstraction
- System Architecture
- System Services and Runtime Supports
Programming Abstractions

• Abstraction level
  – Node level
    • Environment as a distributed system
  – System level
    • Environment as a single virtual system

• Programming paradigm
  – Component-based
    • Model-View-Controller (MVC) model
  – Context-based
    • Context changes trigger events in the application
  – Decentralized
System architecture

• Mode of system control
  – Centralized
  – Decentralized

• Mode of interaction
  – Any communication primitive is fine
    • Message passing, Publish/Subscribe etc.
Design dimensions

Programming Abstraction

System Architecture

System Services and Runtime Supports
Reference Model

Pervasive Computing Middleware

Common System Services

Service Manager

Context Manager

Runtime Support

Processing
Communication
Storage

Cross-layer Support

Reliability Manager
Security Manager

User Applications

Programming Abstraction

Operating Systems

Network Protocols

Hardware Services

Senso

Senso
Context management service

- Gathering raw data
- Event-driven
  - Asynchronous
  - Good for constraints
- Query-based
  - SQL-like interface
  - Synchronous
Context management service

- Aggregate readings
  - Highly-correlated
  - Redundant
- Conserve resources
  - Energy
  - Bandwidth
Context management service

- Large heterogeneous data
  - Redundancy
  - Inconsistency
- Frequent updates
  - Updates for related objects
- Dynamic environments
  - Sources leave
  - Consumer interest change
Context management service

- Represents context
- Purpose
  - Deriving high-level context
  - Reasoning
  - Querying current and history
  - Extending existing context types
Context management service

- Key-value
  - <temperature, 25 °C>
    - Limited in expressiveness
- Logic-based
  - (Temperature, Room A, is, 25 °C)
    - Allow some reasoning
- Object-oriented
  - Encapsulation, inheritance, reusability
    - Nontrivial to extend
- Markup
  - Tags with attributes and content
    - Adapt content with respect to device
      - No reasoning, no relationships
- Ontology-based
  - Resource Description Framework
    - Good for inferring context
      - Limited for inexact reasoning
Context management service

- Infer implicit higher-level context from explicit lower-level ones
- Exact reasoning
  - Inputs must be present and precise
- Inexact reasoning
  - Allows imprecise contexts
  - High computation overhead
  - Not suitable for accuracy critical applications
Reference Model

User Applications

Programming Abstraction

Pervasive Computing Middleware

Common System Services

Service Manager

Context Manager

Runtime Support

Processing

Communication

Storage

Cross-layer Support

Reliability Manager

Security Manager

Operating Systems

Network Protocols

Hardware Services

Senso

Senso
Service management service

• Acquired context might require an action

• Discover services needed by the action

• Compose services to execute the action
Service management service: Discovery

- Resource rich
- Static
- Reliable

- Resource constrained
- Dynamic
- Unreliable
Service management service: Composition

• Describing services
  – Providers advertise atomic services

• Specifying composition plan
  – Low-level requests: specify workflows
  – High-level requests: specify a goal to be achieved

• Selecting service provider
  – Same service from multiple providers
  – Choose the one that suits better
Service management service: Composition

• Architecture
  – Centralized composition
    • Easy to implement
    • Single-point of failure
  – Distributed composition
    • Difficult to implement
    • Highly redundant

• What if provider dies in the ongoing workflow
  – Static technique
  – Dynamic technique

Hybrid composition
Reference Model

Pervasive Computing
Middleware

User Applications

Programming Abstraction

Common System Services

Service Manager

Context Manager

Runtime Support

Processing

Communication

Storage

Cross-layer Support

Reliability Manager

Security Manager

Operating Systems

Network Protocols

Hardware Services
Reliability in context management

- **Problem:** Noisy and incomplete data
  - Inconsistencies
  - Mistakes in reasoning
- Preprocess raw data in sensor layer
  - Deploy statistical and probabilistic techniques
- Context inconsistency detection
- Context inconsistency resolution
  - Keep history and drop outliers
  - Assign quality values to context data
Reliability in service management

- **Problem**: Dynamic environment
  - Hard to follow where services are
- Directory-less
  - Advertisement rate
  - Announcement diameter
- Directory-based
  - Pick $K$ relatively static and reliable nodes
  - Form the backbone with these nodes
  - Store and replicate directory over the backbone
Security in context management

• Sharing context data might breach privacy
  – Health situation passed on unwanted parties
• Access control
  – Determine who can access which data
• Pseudonymity/anonymity
Security in service management

• Prove yourself to access services
  – Bluetooth
  – Universal Plug and Play
• Mutual authentication
• Secure communication channel
Future Research

• Data is getting larger and larger
  – Efficient ways to store and query
• Detect user intention based on situation
• Privacy and security is not studied well
  – It is highly required in such open environments
• Predicting user’s individual and social behavior
Thank you!
Questions/Comments