

Distributed Event Dissemination for Ubiquitous Agents

Sasu Tarkoma

Helsinki Institute for Information Technology



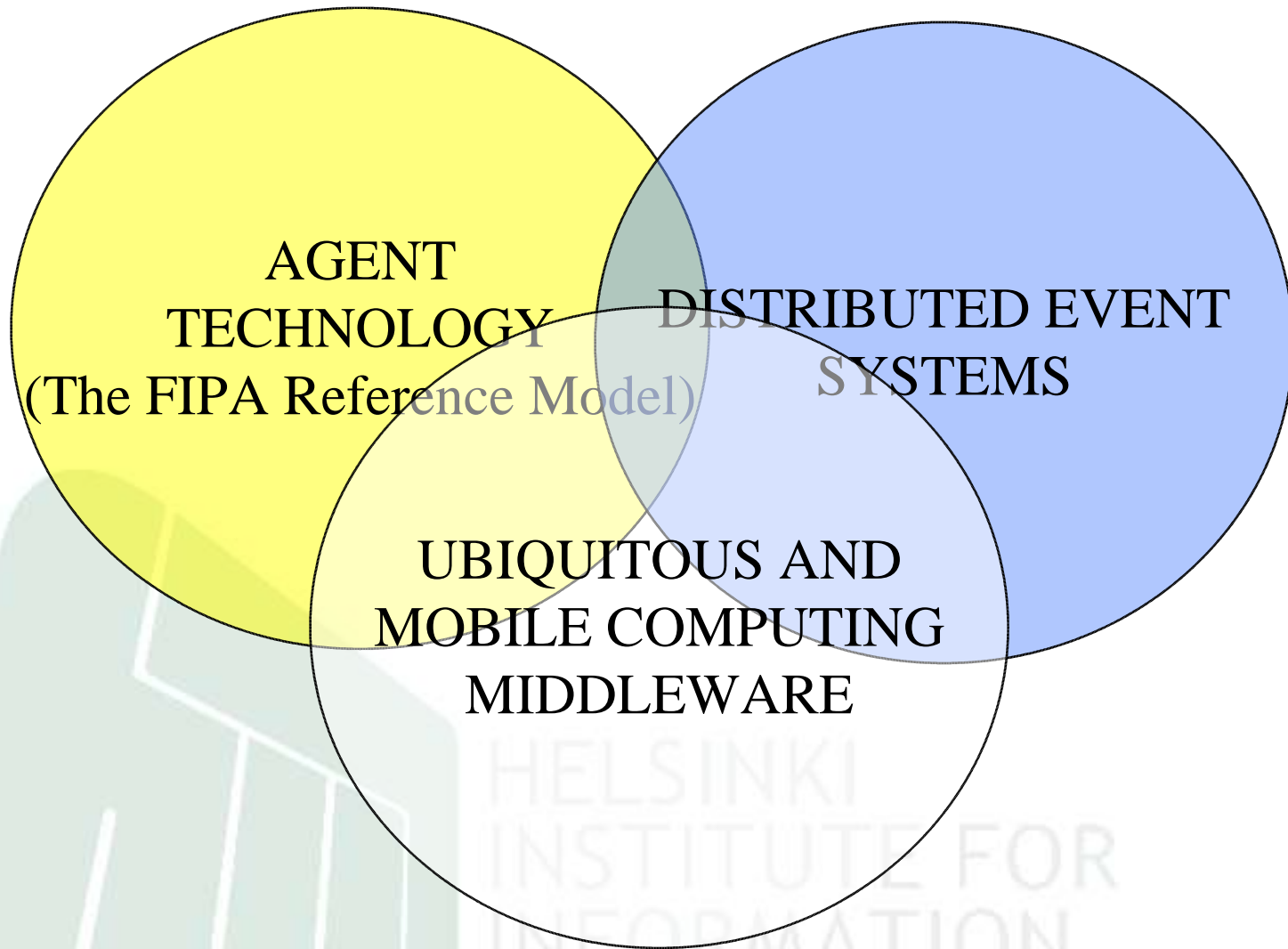
HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

Contents

- Introduction
- Background
 - Ubiquitous computing
 - Agent communication
 - Event-based systems
- Distributed Event Service for Agents
- Architectural and implementation issues
- Conclusions

Introduction

- Multi-agent systems are based on asynchronous communication, which needs to be efficient and reliable
- Current agent communication standards and platform implementations support directed communication
- In ubiquitous environments changes in the environment need to be propagated to interested components that may not know each other beforehand
- Distributed events are a generic enabler for content-based information delivery and de-coupled communication
- Agents may benefit from platform support for distributed events



**AGENT
TECHNOLOGY**
(The FIPA Reference Model)

**DISTRIBUTED EVENT
SYSTEMS**

**UBIQUITOUS AND
MOBILE COMPUTING
MIDDLEWARE**

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

Ubiquitous Computing

- Ubiquitous Computing, Pervasive Computing, and Ambient Intelligence draw a picture of a heterogeneous world of entities interacting in order to provide intelligent and adaptive services for users irrespective of time and location
- In order to detect changes and important properties in the heterogeneous environment we need mechanisms for subscribing and producing information in a dynamic fashion
- The ubiquitous environment presents a number of challenges that need to be met
 - in the design of agents
 - in the underlying agent platforms, and
 - other middleware

Challenges

- Wireless communication
 - Reliable message transport protocols
- User mobility
 - User changes terminal or becomes disconnected
- Terminal mobility
 - Terminal roams to a new physical location, which may cause a change in the logical location/service access point.
- Terminals have limited memory, processing power, and battery life
 - Software agents need to be relatively simple in order to be deployed on PDAs and mobile phones
- Challenges of ad hoc operation

Agent Communication

- Currently, agent communication is directed to a priori set of receivers, which are determined using unique identifiers or names
 - FIPA specifications
 - Most agent platform implementations
- Information dissemination plays a large role in many agent applications
 - This motivates the support for content-based delivery of information
 - Message delivery not based on a priori addresses but on the content of the message and the interests of the receiver
 - Distributed events

Event-based Systems

- Event-based middleware
 - A good candidate for mobile and ubiquitous computing: asynchronous, anonymous, one-to-many communication
 - Information dissemination, content-based forwarding using filters
 - Standards such as the CORBA Notification Service, and the Java Messaging Service
 - Research projects such as JEDI, Siena, Elvin, Gryphon,...
 - Event service
 - logically centralized service that provides interfaces for interest registration and notification
 - Notifier pattern, event channel pattern

Rendezvous-Notify (R-N)

- An overlay event architecture for mobile computing that consists of mobile clients and servers
- Two server roles:
 - Access servers maintain event sessions for clients. Sessions store subscription information and buffer events
 - Resolution servers maintain event channels
 - Distributed event channel subscription table and two-phase filtering between servers, filter merging
- Access to event channels is done using a rendezvous mechanism.
 - Constant or near constant subscription management cost
 - Session handover procedure for mobile clients
- See DEBS'03 paper for more information

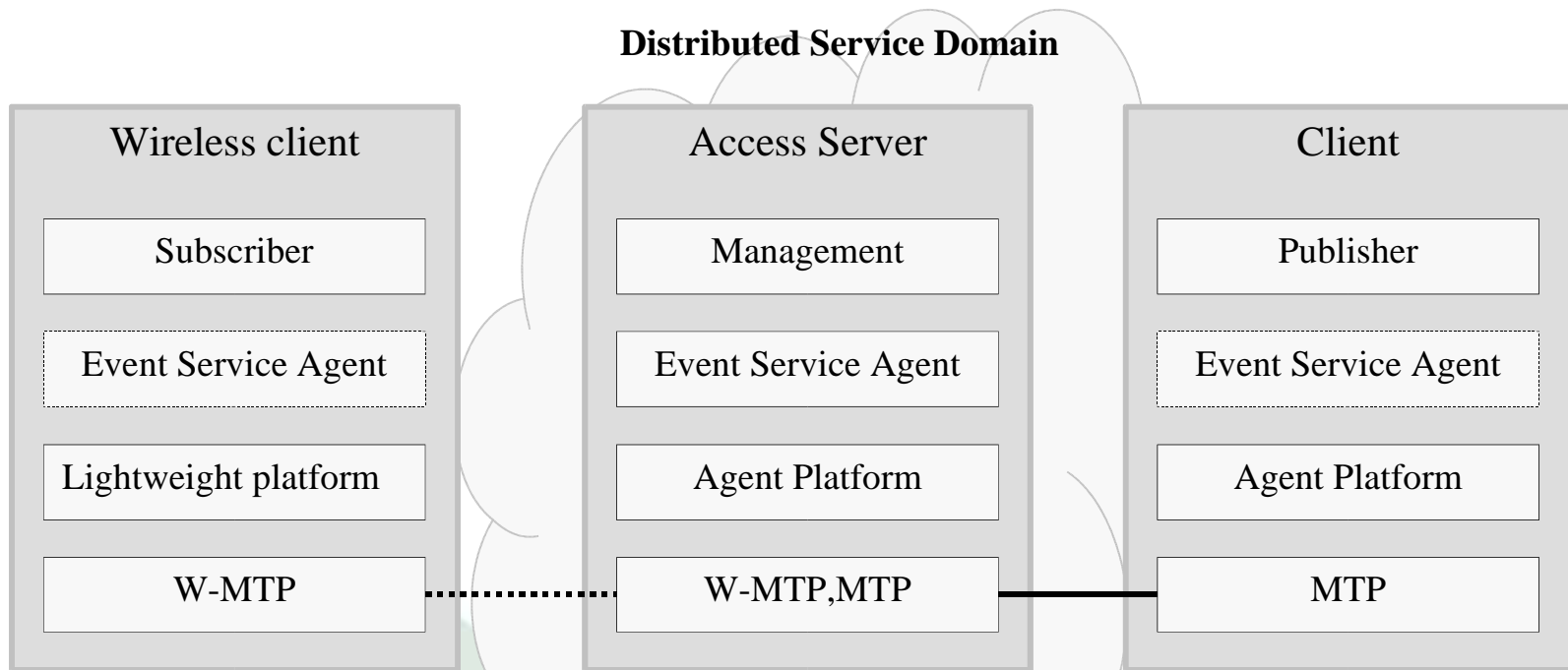
Events for Agents

- Currently, using FIPA platforms, agents need to manage subscriptions explicitly using the Observer pattern. This requires the following phases:
 - Subscriber agent locates a producer agent
 - Subscriber contacts the producer and subscribes the event using a filter
 - Producer accepts the filter and adds it to the list of subscribers
 - In time, the producer agent generates the event and notifies subscribers whose filter matches the event
- A discovery service (for example DF) is used to locate every producer agent. The producer agent needs to have the processing power for subscription management
- In this model subscribers and producers are coupled, which causes problems during disconnections / relocations

Events for Agents II

- Subscription management may be delegated to the agent platform or a designated component using the Notifier pattern
 - Support local and distributed events using the same interfaces
 - QoS and mobility support
 - Less complex agents. For example: the monitoring tasks of a reactive agent may be defined using filters and distributed over the event network
- Different implementation approaches
 - as methods available to the agent, SOAP/WSDL interface, ACL-interface
 - For portability we need standards compliant interfaces, for example: Event Service Agents with ACL interfaces and different roles for clients and servers

Agent-based Event Service



Discussion I

- Agents would benefit from a distributed event system:
 - Delegation of monitoring tasks and thus simpler agents (less concurrent tasks)
 - Optimization of filters and event delivery. The service may optimize the communication by using filter merging, covering relations, and status queries
 - The goal is to filter as close to the source as possible, and to minimize the number of false positives
 - Anonymous group (one-to-many) communication
 - Resource discovery and synchronization through events (sync. depends on the ordering / timestamping model)
 - Support for mobility and disconnected operation through the underlying event service

Discussion II

- For example: Rendezvous-Notify supports
 - user mobility (buffering notifications in sessions),
 - terminal mobility (handover procedure for sessions, and fast updates to subscription topology), and
 - agent mobility (same as terminal mobility)
 - Agents know the next location beforehand so handover may be optimized
- Client-side filtering is needed for flexible communication and peer-to-peer operation
- Standards-based interfaces are required for interoperability
- ACL-based interfaces allow the system to take an advantage of platform services such as the DF, and the support for ontologies. Ontologies allow agents to define the concepts used in the events.

Conclusions

- Agents are based on asynchronous messaging and events that make reactive computing possible
- Ubiquitous environments require that signals and information from heterogeneous sources are delivered efficiently to interested parties
- Agents may gain benefits from a distributed event framework
- The event service can filter, optimize and distribute filters, route, and store events on behalf of the agents and provide mobility support
- The implementation of information producing/consuming agents may be simplified by delegating the subscription processing and management to the event service