Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

Exploring Spontaneous Communication in a Seamless World

Vando Batista

CIn/UFPE

PhD Candidate

Centro de Informática U·F·P·E

Objectives

- Identify opportunities and challenges of mobile computing and networks
- Identify how to explore ad hoc connectivity
- Learn how to build distributed systems on top of Spontaneousware

- **Target audience**
  - Java™ developers interested on **mobile applications** and **spontaneous interactions** between users

Speaker
Vando Batista

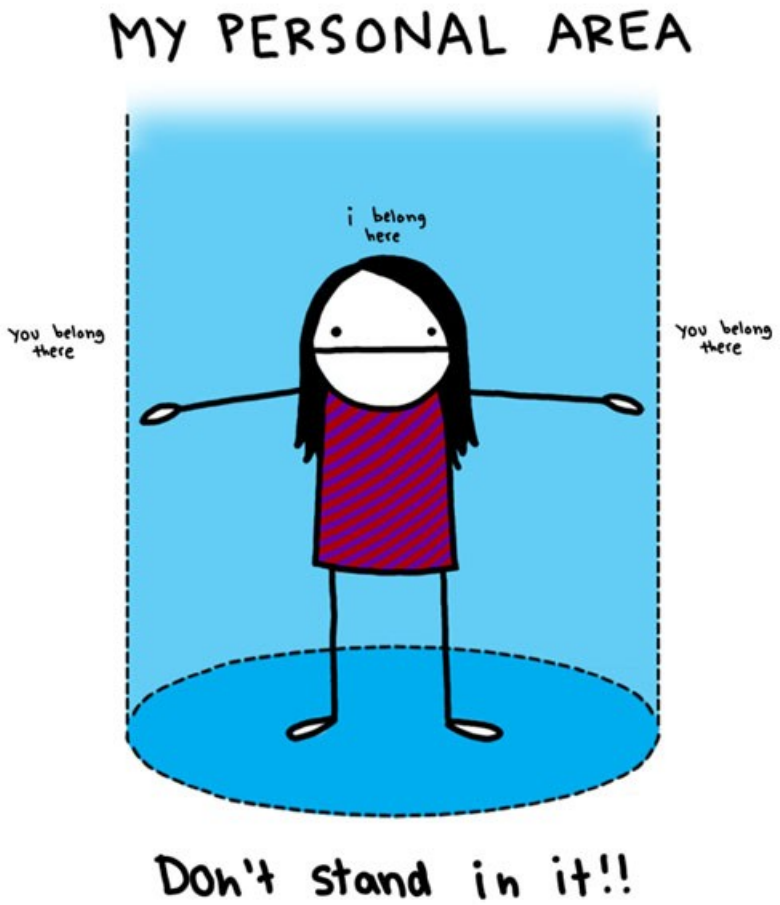
- PhD Candidate at CIn/UFPE
- Systems Analyst at Informe Air
  - RFID and Container Tracking
- Mobile Java Platform Consulting/Instructor
- Worked with Java ME, SE for Motorola and SonyEricsson projects at C.E.S.A.R
- MSc degree in Computer Science at CIn/UFPE
  - Spontaneousware: a Middleware Framework for Mobile Ad Hoc Networks
- Brazilian Computer Society (SBC)'s student delegate at CIn/UFPE
- Certifications: SCMAD, SCPJ, SCJA

Contextualization

- Evolution, popularization
  - Device resources
  - Connectivity

- New opportunities, new services
- Impacts on user interaction
- Mobile transparency

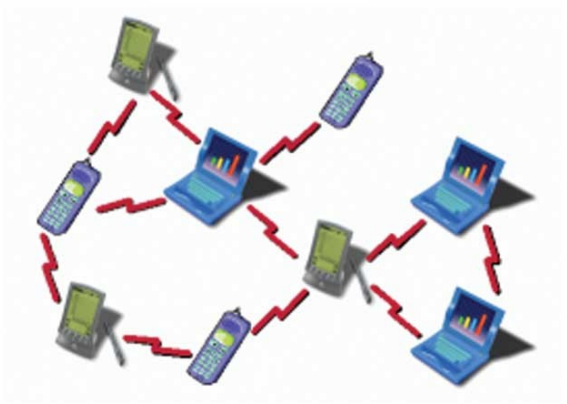
Mobile Communication



Bluetooth™

ZigBee™

uwb
embedded technology

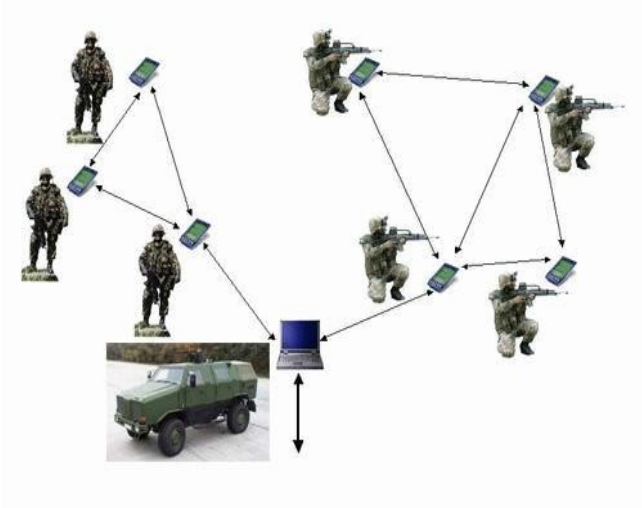


Ad hoc

MANETS

Figures: ARC Communications Research Network; www.nataliedee.com

Applications



- Military
- Emergency rescue
- Vehicular
- Interests/Goals sharing
- Marketing

- **On the move apps**

Improve user experience

Figures: www.fgan.de; car2car consortium

In a more complex environment…

- Devices
  - heterogeneity, memory, process
- Network
  - decentralization, bandwidth, range, service discovery
- Resources
  - distribution, connection availability, battery

- **How to explore human interaction on a resource constrained and high-mobility environment?**

Technical Needs

- Leverage spontaneity on communication
- Follow the user displacement

- Requirements
  - Discover hosts dynamically
  - Effective use of connection
  - Handle disconnections
  - Support platform interoperability

Initiatives

- Decrease effort, cost
- Improve productivity, reliability
- Middleware
  - Fixed vs. Mobile
  - Provide transparent access to mobile service and resources with effective cost
- Reuse
- Framework

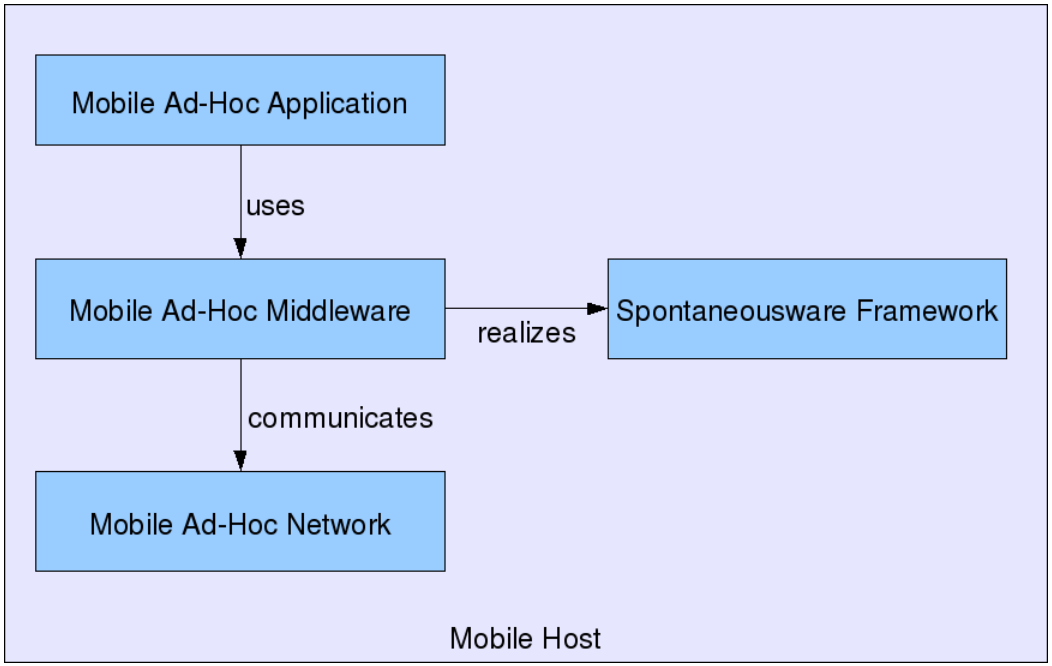
Solutions

ACE
MIDAS
ReMMoC
PolyORB
Peer2Me
MobCon
Arcademis
Quarterware
JXTA JavaME
LIME
UbiquitOS
Marge
SELMA
Limone
XMIDDLE
MESHMdl
STEAM
Mobile Gaia
JMS
EMMA
Expeerience

Spontaneousware
Proposal

"Generalization of variety efforts towards the middleware development for application support over MANETs"

- Framework
- Reuse
  - Requirements
  - Architecture
  - Design
  - Code

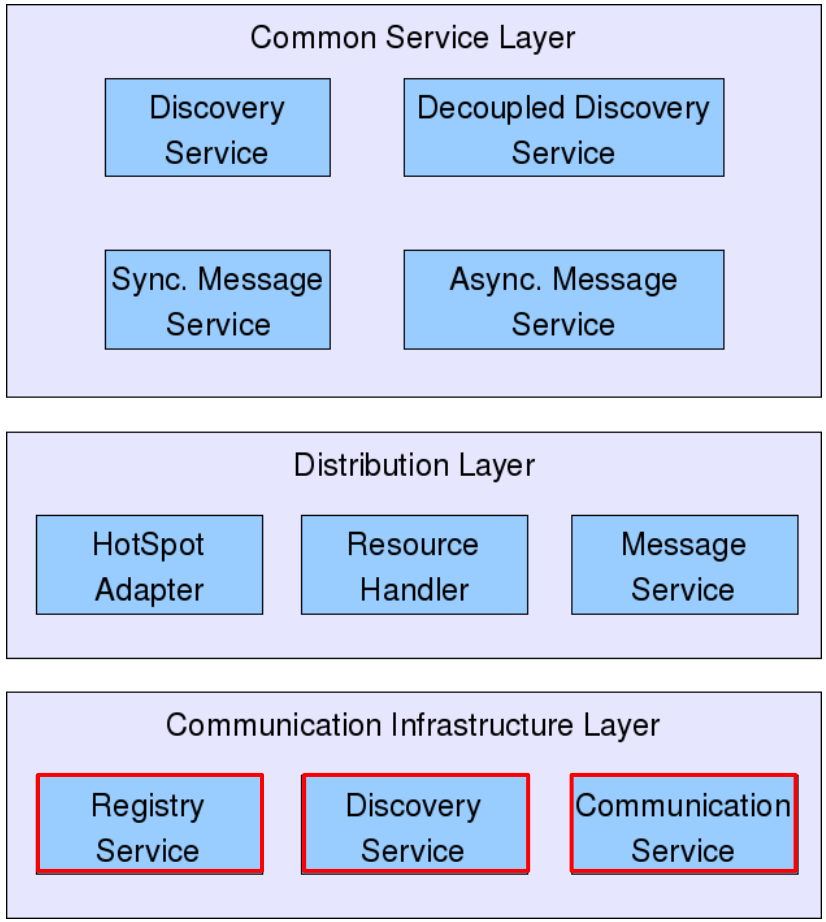


Spontaneousware
Characteristics

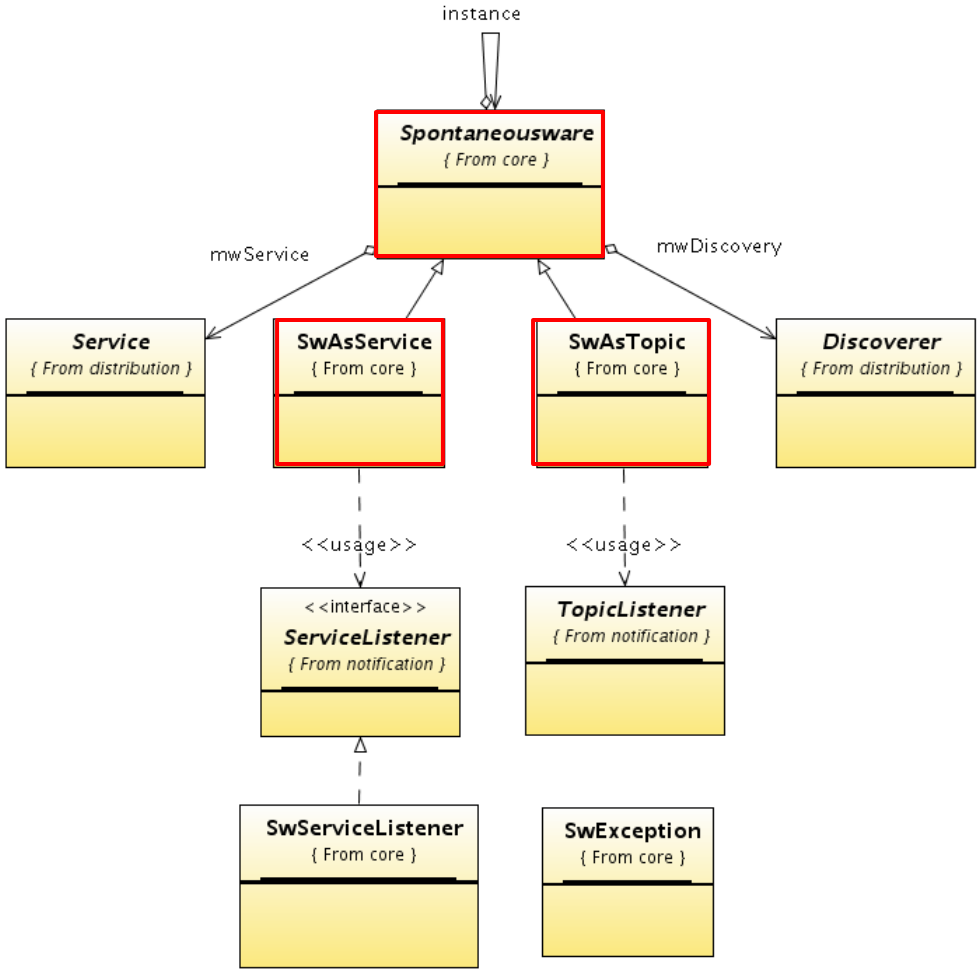
- Framework classification
  - Middleware integration
  - White-box
- Generated middleware
  - Message-Oriented (MOM)
  - Communication
    - Seamless connectivity
  - Services
    - Synchronous
    - Asynchronous

Spontaneousware
Features

- Common API
  - Main behavior defined
- Extension Points
  - Communication
  - Registry
  - Discovery
- Messages
  - Type abstraction
  - Persistent vs. Transient



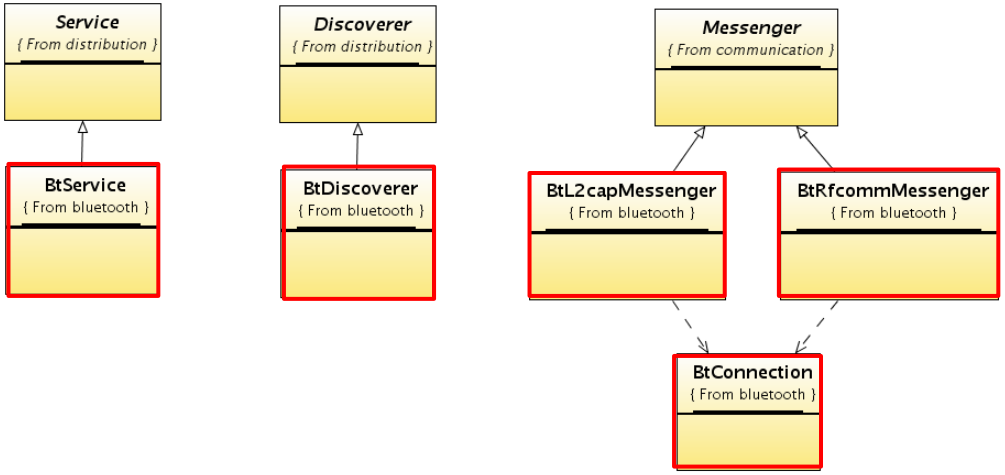
Spontaneousware
Classes



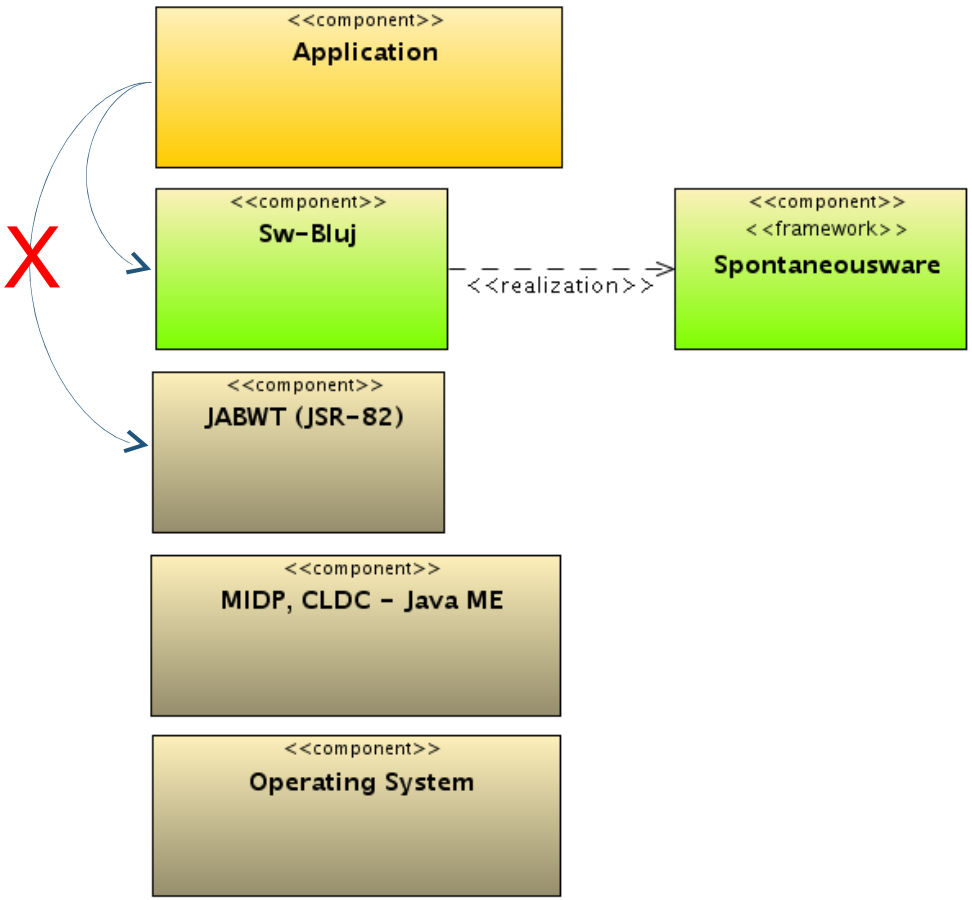
Sw-BluJ

Middleware implementation

- Spontaneousware for Bluetooth Java
  - **Java ME** platform
  - **Bluetooth** network
- Loads protocol dynamically (L2CAP, RFCOMM)



Sw-BluJ



- Application do not access network layer directly
- Middleware hides the network complexity
- Network-independent services

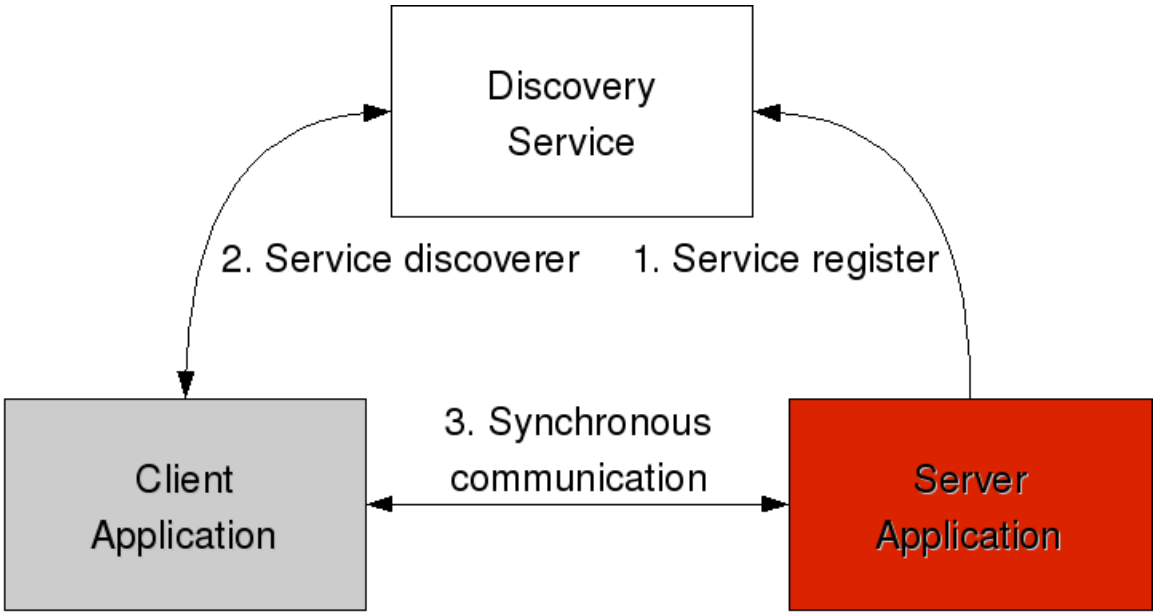
Synchronous Scenario

Synchronous Communication

- Basic communication infrastructure
- Application service based
  - A middleware identified by a specific service
  - A lot of services on each middleware/host
- Location transparency
  - Service name identification
- Applications
  - Register a service
  - Search a service
  - Communicate between **client** and **service**

Synchronous Communication

- Point-to-point model



Synchronous Communication
Service Registry

```
// middleware instantiation
SwAsService sw = Spontaneousware.getAsService();
// creates a service listener
ServiceListener asl = new AppServiceListener();
// registers the service javaoneconf
sw.registerService("javaoneconf", asl);
```

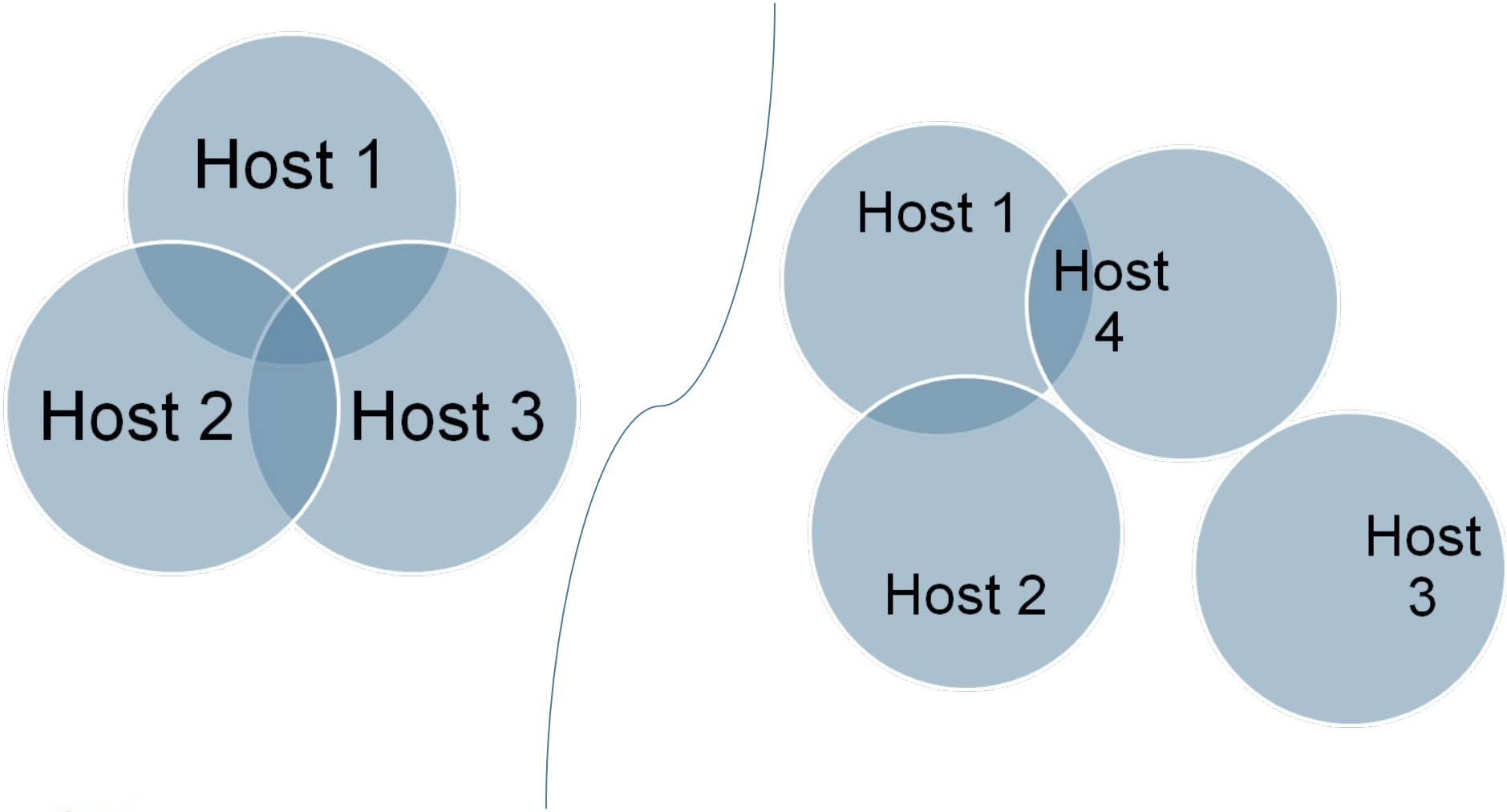
Synchronous Communication
Search Services & Message Exchange

```
// middleware instantiation
SwAsService sw = Spontaneousware.getAsService();
// finds the javaoneconf service
Messenger mSync = sw.findService("javaoneconf");
// sends 3 messages
mSync.send("JavaOne"); // String
mSync.send(2009); // int
mSync.send(true); // boolean
// receives service response
mSync.receiveBoolean(); // true
```

Synchronous Communication
Message Exchange

```
// service notification method
ServiceListener.handleRequest(Messenger mSync) {
  // receives 3 message
  mSync.receiveString(); // JavaOne
  mSync.receiveInt(); // 2009
  mSync.receiveBoolean(); // true
  // sends confirmation response
  mSync.send(true);
}
```

Asynchronous Scenario

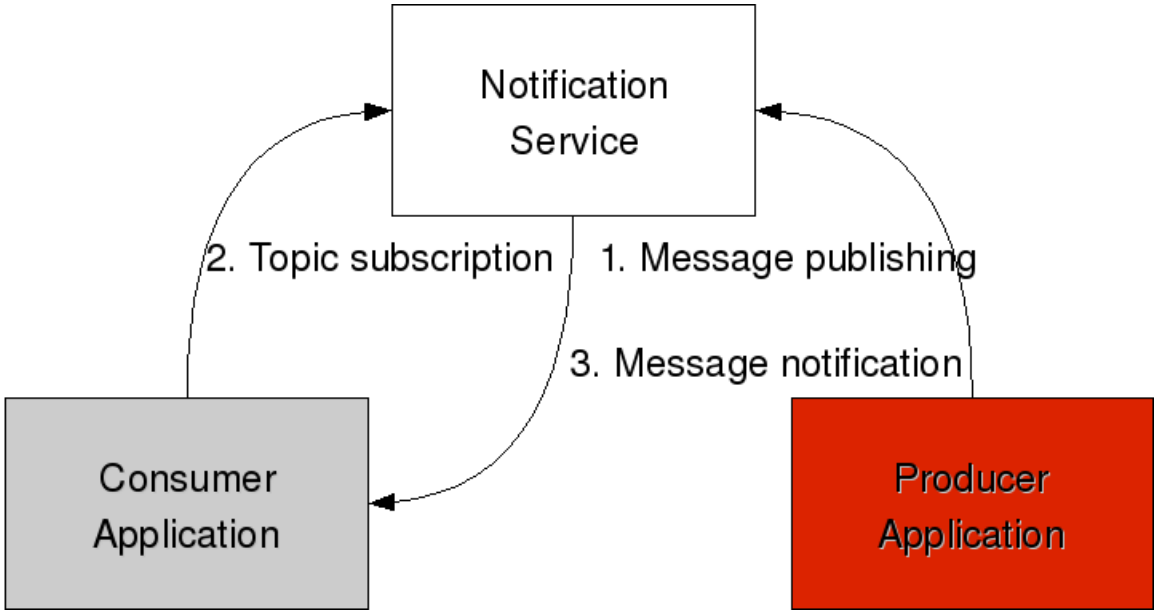


Asynchronous Communication

- Message topic based
  - Publish/Subscribe mechanism
  - Anonym communication
  - *Message Queue* and *Result Callback*
- Mobility transparency
  - Topic name identification
- Applications
  - Subscribe in a topic
  - Publish a message in a topic
- **Producer** and **Consumer** process

Asynchronous Communication

- 1:M model



Asynchronous Communication
Topic Subscription

```
// middleware instantiation
SwAsTopic sw = Spontaneousware.getAsTopic();
// creates a topic listener
AppTopicListener atl = new AppTopicListener();
// subscribes in mobility topic
sw.registerToTopic("mobility", atl);
```

Asynchronous Communication
Message Publishing

```
// middleware instantiation
SwAsTopic mw = Spontaneousware.getAsTopic();
// finds the mobility topic
MessengerAsync mAsync = mw.findTopic("mobility");
// creates a message
Message msg = new Message();
// sets message content
msg.setContent(myContent);
// publishes the message in topic
mAsync.send(msg);
```

Asynchronous Communication
Message Notification

```
// topic notification method
AppTopicListener.receiveMessage(MessageReceived recMsg) {
  // gets the message type
  String type = recMsg.getMessageType().getName();
  try {
    String rs = "";
    // checks message type
    if ("java.lang.Boolean".equals(type)) {
    // gets the message content
    rs = new
  Boolean(recMsg.getContentAsBoolean()).toString();
    ...
  }...
```

Demo

- Applications will be shown demonstrating how new opportunities could be explored leveraging the spontaneous interactions between mobile devices/ users.

More Information

- Release package
  - 30 Kbytes
- Experimental evaluation, Reuse percent
  - Middleware: 72%
  - Application: 40% LOC, 43% CC
- The project is hosted at: http://spontaneousware.dev.java.net/
  - including the framework, middleware, and applications

Conclusions

- Discussion about spontaneous communication
- A initiative for high mobility environment
- A way to decrease the application complexity

Thank You

Vando Batista
vandofb@gmail.com
www.cin.ufpe.br/~vfb

PhD Candidate :: CIn/UFPE