Intergiciels pour réseaux ad hoc et spontanés

Guy Bernard
Institut National des Télécommunications, Evry, France
Guy.Bernard@int-evry.fr

Agenda

- top challenges for future middleware research
- state of the art
  - EU-IST workshops
  - middlewares from research teams
  - FP6 european projects
  - national initiatives
  - standardization activities
  - research teams in France and abroad
- conclusions
**Taxonomy**

- mobile systems
  - nomadic systems: wired core network and mobile nodes
  - ad hoc systems: no wired network, all nodes are mobile
  - (relatively) large resources, wireless network links
    - example: HP iPAQ hw6900 Mobile Messenger
    - Quad band GSM/GPRS/EDGE, 802.11b, Bluetooth 1.2, infrared
    - integrated GPS
    - Intel PXA270 416 MHz
    - 45 MB (persistent user storage) + 64 MB SDRAM (running applications)

- embedded systems
  - processing components embedded in a "device" (car, plane, …)
  - not mobile, small scale, wired connection with a server

- sensor systems
  - very limited resources
  - not mobile, large scale, wireless connection with a server

**Middleware**

- unifying software layer between
  - operating system and network,
  - distributed applications.

- gathers basic services that are necessary for developing and executing distributed applications

**Contents and services**

... collaborative work, crisis management, e-commerce, ...

Middleware

Operating System

Hardware Platform

Network
**“Classical” middleware example: CORBA**

**CORBA (2)**

- **client-server model (synchronous request-reply)**

---

**ORB core (Object Request Broker)**
CORBA services

Middleware development evolution

1980’s: distributed system platforms
- Athena (MIT), Andrew (CMU), ANSAware

1990’s: standardization
- DCE (OSF), CORBA (OMG), RM-ODP (ISO)

2000’s: general-purpose middleware architectures for distributed object computing
- CORBA (OMG), DCOM/COM+ (Microsoft)
- component-based design

2000’s:
- Java-based middlewares
  - RMI, JINI, Javaspaces, EJB
- message-based middlewares
  - MQSeries (IBM), JMS (Sun)
Needs for revisiting middleware design

- changing environment:
  - programming models
  - architecture
  - dynamic configuration

Changing environment (1)

(from Unix workstations connected by a wired LAN)

1. Enterprise application integration
   - need for integrating many applications/data sources within/across enterprises
   - large scale configuration, diverse interaction models
     - a chain of consecutive RPCs is too rigid
     - requirement for loosely-coupled interaction (autonomy of the various parties, spatial and temporal decoupling)

2. Internet applications
   - number of users unpredictable, stateful session difficult to maintain, no QoS guarantees, Web-based and legacy applications must interoperate, ...
   - requirement for autonomy, decentralized authority, intermittent connectivity, continual evolution, scalability
3. Quality of service

- need for response time, availability, data accuracy, consistency, security for commercial applications (charging for service) over a best-effort communication environment
- research projects about extended CORBA-based platforms (real-time, replication, fault-tolerance)
- integrating QoS management into middleware architectures is essential
  ➔ ... but a procedure for doing so has yet to be agreed upon

4. Nomadic mobility

- need for accessing and processing information “anywhere and any time”
- available resources vary widely and unpredictably
  - error rate, bandwidth, battery
- location awareness
  ➔ middleware must support the applications to explicitly accommodate these changes

5. Ubiquitous environment

- future computing environments will comprise diverse computing devices
  - from large computers to microscopic processing units
  ➔ addressing and naming a multitude of devices cannot be done with current technology
  - cannot consume an IPv6 address for each item sold
Programming models (1)

- **client-server**
  - distributed processing traditional paradigm
  - sequential "pull" model with a single logical control thread -- synchronous

- **publish/subscribe**
  - provides "push" model too -- asynchronous
  - example: workflow systems

- **Web cookies**
  - scalability ➔ the state cannot be maintained on the server
  - dealing with state requires a particular programming style

- **Peer-to-peer interaction**
  - serverless file sharing

⇒ CL-SV model is inappropriate in many interaction scenarios

Programming models (2)

1. **asynchronous interaction**
   - RPC paradigm cannot accommodate scalability, flexibility, decoupling, Internet applications
   - CORBA advances (3.0)
     - one step towards asynchronism: deferred invocations, oneway operations
     - CORBA messaging: asynchronous style, selectable QoS guarantees (message priorities, request time limit, queuing strategies)
     - CORBA notification & event service
   - SOAP
     - design goals: lightweight, open, flexible
     - request-response pair = 2 one-way transmissions
   - **event-based middleware**
     - goal: support decoupled and asynchronous interaction in large scale systems
     - uses events as the primary means of interaction
     - allows P2P notification between objects, event filtering, forwarding
   - **message passing**
     - the old brave stuff
     - accommodates P2P interaction
     - error-prone, difficult to test and debug
     - a backward step in middleware evolution
Programming models (3)

2. shared memory
   - ancestor: Linda tuple space approach (1986)
   - Linda idea revival
     - JavaSpaces (1999)
     - PerDiS project (2000)
     - FT-Linda, Objective Linda, …
   - shared memory issues in presence of disconnections
     - conventional pessimistic locking and concurrency control mechanisms are too stringent
     - rather, optimistic concurrency control, logging and reconciliation

3. mobile code
   - inherently asynchronous
   - new programming model, new typing concepts, new security provisions
   - shortcoming: homogeneous programming language environment

→ multitude of different models, vs. unified middleware programming model that supports decoupled, flexible and scalable interaction ??

Architecture (1)

1. distribution transparency
   - beneficiary and necessary for programming distributed applications – was the ultimate goal of “distributed operating systems”
   - cannot be the foremost goal in nomadic computing and context-aware applications
     - need selective transparency features
     - open research questions
       - how to expose network imperfections at the right level of granularity and abstraction?
       - how applications on top of the middleware deal with a selectable degree of transparency?
       - complete transparency is inappropriate when applications must adapt to fluctuations in resource supply
         - example: consumers should be aware of battery shortage or bandwidth variation

→ currently no middleware facilities to control the degree of transparency
Architecture (2)

- example: AMPROS architecture (INT 2003)

Architecture (3)

2. layering
   - new application requirements make necessary to support direct interaction between non-adjacent layers
     - because of selective distribution transparency need
     - examples
       - context-aware applications may need IP address / geographical location
       - the application may need authentication information
         - e.g., encryption key in SSL transport layer – the middleware itself is not interested in this information
       - an application may require a customized transport protocol to perform a multimedia transfer
         - this influences a nonadjacent layer’s configuration
     ➔ middleware could offer appropriate programming interface elements to the applications and pass information to and from the lower layers
     ➔ middleware may need runtime information via a callback to the applications to request handling decisions (e.g., caller’s security credentials)
Architecture (4)

3. **Monolithic architectures**
   - current middleware platforms (CORBA, COM+) are too bloated to be loaded into a resource-scarce mobile device
     - PalmPilot PDA: 2 MB memory
     - middleware should be customizable
   - same for QoS management
     - diverse application requirements \(\Rightarrow\) impossible to construct a ready-made platform for all QoS needs
     - example: QuO (BBN, 1998)

   ➤ find the **specific patterns that middleware frameworks require**

Dynamic configuration (1)

1. **Disconnected operation**
   - PDAs switch themselves off to save battery power (or other reasons)
   - server side: keeping state, buffering replies (how long ?), reauthentication
   - client side: reply identification, restoring the state of earlier server associations
   - this should not be considered as a fault and handled by conventional fault-tolerance mechanisms
     - Web applications: encode state information in URL, store it in cookies
     - CORBA cannot handle this operation in a systematic, architected way

   ➤ **message-based, store-and-forward communication**
Dynamic configuration (2)

2. **adaptive applications**
   - mobile applications must cope with dynamically varying resource supply
   - the underlying middleware cannot completely mask these fluctuations
     - same problem as for QoS management
     - example: bandwidth becomes scarce ➔ display text and low-resolution pictures instead of video clips
   ➔ the middleware must
     - monitor the resource supply and demand
     - compute adaptation **decisions**
     - notify applications if they require adaptation
   ➔ **strategies** for making adaptation still requires exploration

Dynamic configuration (3)

3. **ad hoc organization**
   - lower-layer discovery protocols ➔ devices can automatically detect others and spontaneously form ad hoc agglomerations
   ➔ existing middleware platforms **do not scale** to
     - device diversity
     - population size
     - runtime dynamics
   ➔ self-organization, extensive information caching, delegation of activities
   - Jini
     - Java objects can discover, join, and interact with communities of objects
   ➔ **scalability** ??
4. intermediaries
   - a low-end device may be incapable of hosting the complete middleware software
     - requires support from an intermediary on a more powerful device
     - the intermediary
       - translates and forwards external communication requests to the low-end device
       - manages agglomerations of low-end devices
   - connecting heterogeneous middleware domains (e.g. COM+ - message queuing system)
   - manipulation of information streams
     - pragmatic solutions, but lack of comprehensive general principles
     - integration of security, transactions, conversion overhead, reliability

Synthesis: top challenges for future middleware research

- what is the most appropriate programming model for the diverse application scenarios?
- does a single distributed programming model fit all applications?
- can we build customizable, configurable, and flexible middleware for inherently heterogeneous environments?
- what middleware features and infrastructure services will the dynamics and ad hoc nature of mobile-ubiquitous computing require?
State of the art

- EU-IST workshops
- middlewares from research teams
- FP6 european projects
- national initiatives
- standardization activities
- research teams in France and abroad

Expected functional properties

**mobile systems:**
- event notification
  - nomadic systems: through wired network – ad hoc systems: decentralized
- mobility & location awareness
  - capacity to identify a new environment and to adapt to it
- addressing
  - identification of the diverse nodes
- service discovery
  - identification of new services/protocols
- code updater
  - dynamic changing of protocol or code (application/middleware) to adapt to a new context

**embedded systems:**
- + real time

**sensor systems:**
- + wakeup coordination, cross layer communication, multi-hop routing, clustering, distributed task scheduler, …
Expected non-functional properties

- heterogeneity
  - language, OS, hardware
- openness
  - capacity to extend/modify the system
- scalability
- disconnection tolerance
- fault tolerance
  - capacity to function in presence of failures of other components
- security
- performance
  - real time, minimization of memory footprint, energy, ...
- adaptability
  - capacity to adapt to changes application/user/network

Analysis of EU-IST workshops (1)

- Co-operating Objects Workshop, June 23-24, 2005 (Unit G3 - Embedded Systems)
  - 6 sessions organized according the contributions received:
    - Wireless Sensor Networks (17 presentations)
    - Middleware for Co-operating Objects (7)
    - Intelligent Infrastructures (5)
    - Agriculture, Farming and Environment (6)
    - Domestic/Professional Environments and Appliances (3)
    - Security, Trust and Ethical Aspects (5)
  - the interest of middlewares for [mobile] cooperating object systems is not commonly perceived
    - among 43 presentations, only 7 mention the word “middleware”
Analysis of EU-IST workshops (2)

- **Pervasive Networked Systems, March 6-7, 2006 (Unit D1 – Communication Technologies)**
  - Sessions:
    - State of the art and vision (6 presentations)
    - Security, privacy and society dimension (5)
    - Applications and industrial perspective (8)
    - Snapshot on IST initiatives (7)
    - Beyond Europe (USA, Japan, Korea)
    - Possible barriers to deployment (4)
  - Among 33 presentations, only 10 mention the word “middleware”
  - Even major actors under-estimate its importance
  - Example: Broadband & Wireless Networking Lab, Georgia Tech:
    - Industrials need middleware for ad hoc / spontaneous / sensor networks
    - The importance of middleware is slowly emerging
    - Academic research leads the race

Middlewares from research teams (1)

- One reference: JXME
  - Extension of JXTA – replacing proxies by ad-hoc communications
- SOUL (Self-Organized Ubiquitous Learning) project – U. Trier
  - JXUv2 – A middleware for multi-hop ad-hoc networks (INET) (2004)
- SELMA – A Middleware Platform for Self-Organizing Distributed Applications in Mobile Multi-hop Ad-hoc Networks (2004)
- PROEM (Mobile Peer-to-Peer Platform) - U. Oregon
  - “Java interfaces and classes for rapid development of mobile peer-to-peer applications called peerlets” (2001)
- Mobile GAlA (U. Illinois)
  - “A Middleware for Ad-hoc Pervasive Computing” (2005)
- STEAM (Scalable Timed Events And Mobility) - Trinity College Dublin
  - “Event-Based Middleware for Wireless Ad-hoc Networks” (2003)
- EXPERIENCE and JMobiPeer - U. Catania
  - “JXTA middleware for mobile ad-hoc networks” (2003)
  - “A middleware for mobile peer-to-peer computing in MANETs” (2005)
- InfoWare – U. Oslo
  - Middleware Services for Information Sharing in Ad-hoc Networks – application to rescue and emergency services (2005)
- WSAMI (Web Services for AMbient Intelligence) – INRIA Rocquencourt, ARLES project
  - Declarative language + core SCAP-based middleware (JXME)
  - Overall memory footprint: 3.9MB (including 3MB for the JVM)
  - Ongoing extension to MANETs (2005)
- Humboldt U. Berlin
Middlewares from research teams (2)

- **MESHMdl (MESH Enables Spontaneous Hosts)** – TU Berlin
  - "A Middleware for Mobile Ad Hoc Networks" (2005)
  - "gives the developer abstractions like mobile agents and space-based communication"

- **EMMA – U. College London**
  - Spontaneous Messaging Middleware for Ad hoc networks (2004)

- **MPP (Mobile Peer-to-Peer protocol)** - TU Munich
  - "combine the MANET principle with the Peer-to-Peer networking architecture leading to a number of innovative "self organizing" network applications" (2004)
  - cross-layer design

- **7DS (Seven Degrees of Separation)** – U. North Carolina
  - "a new peer-to-peer architecture, protocols and implementation that enables devices to share resources, in order to increase their availability. Devices can also serve as ad hoc gateways into the internet. In 7DS, peers can be either mobile or stationary" (2002)

- **Mobile Chedar - U. of Jyväskylä**
  - "an extension to the Chedar peer-to-peer network allowing mobile devices to access the Chedar network and also to communicate with other Mobile Chedar peers" (2005)

- **MIN (formal Methods In peer-to-peer Networking)** – Turku Centre for Computer Science
  - "Integrated Architecture for Peer-to-Peer and Ad Hoc Overlay Network Applications" (2005)
  - "on top on an application-level P2P overlay over a link-level MANET" (2004)

- **Orion (Optimized Routing Independent Overlay Network)** – Nat. U. Singapore
  - "construction and maintenance of an application-layer overlay network that enables routing of all types of messages required to operate a P2P file sharing system"
  - "context discovery platform that supports context publishing and lookup beyond single smart space boundary" (2005)

- **FAME2 (Framework for Applications in Mobile Environments 2)** – U. Kassel
  - "development of middlewares which are deployed on mobile devices such as PDAs where they provide server level functionality"
  - "framework for building application-specific middlewares – example size: 150 KB"

Middlewares for mobile systems

<table>
<thead>
<tr>
<th>Name</th>
<th>Lab</th>
<th>Functional properties</th>
<th>Non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAME2</td>
<td>U. Kassel</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Mobile Chedar</td>
<td>U. of Jyväskylä</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>FAME2</td>
<td>U. Kassel</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Mobile Chedar</td>
<td>U. of Jyväskylä</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>FAME2</td>
<td>U. Kassel</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Mobile Chedar</td>
<td>U. of Jyväskylä</td>
<td>X X X X X</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Turku</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>Orion</td>
<td>Nat. U. S.</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>FAME2</td>
<td>U. Kassel</td>
<td>X X X</td>
<td></td>
</tr>
</tbody>
</table>

- most of the effort on the requirement of adaptability – but the adaptation mechanism of the middleware remains static during its lifetime
- no single middleware addresses all the non-functional requirements
## Middlewares for embedded systems (1)

- **RTZen (U. of California, Irvine)**
  - real-time CORBA:
    - processor resources: thread pools, priority models, portable priorities
    - communication resources: protocol policies, explicit binding
    - memory resources: request buffering
  - Java-based ORB, 5th generation of ORB design
    - 1st generation: static monolithic
    - 2nd generation: monolithic with compile-time configuration flags
    - 3rd generation: dynamic micro-ORB: small kernel + various components linked/loaded on demand
    - 4th generation: dynamic reflective micro-ORB: application description used to “prime” the ORB
    - 5th generation: static reflective micro-ORB:
      - application configuration and needs are learned by dynamic reflective micro-ORB
      - a model-based generator builds a custom-ORB for this application
      - the custom-ORB is compiled and placed in ROM

## Middlewares for embedded systems (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Lab</th>
<th>Functional properties</th>
<th>Non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Brand-new</td>
<td>Resilience</td>
</tr>
<tr>
<td>RTZen</td>
<td>UCI</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Sensor systems (1)

Class V tags
Readers, can power other Class I, II and III tags; communicate with Classes IV and V.

Class IV tags: active tags with broad-band peer-to-peer communication

Class III tags: semi-passive RFID tags

Class II tags: passive tags with additional functionality

Class I/Class II read-only passive tags

Sensor systems (2)

Open Experimental Platform

TinyOS

Services

Networking

WeC 99 “Smart Rock”

Rone 11/00

Dot 9/01

Mica 1/02

Telos 4/04

Robust Low Power Easy to use

Commercial Off The Shelf Components (COTS)

 TinyOS – Sensor networks

ECOTEL 2004

ECOTEL 2004
### Berkeley Motes

<table>
<thead>
<tr>
<th>Processor/Radio Board</th>
<th>MPR410CR</th>
<th>MPR410CB</th>
<th>MPR422CP</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package/Height (mm)</td>
<td>20.4 x 22</td>
<td>20.4 x 22</td>
<td>20.4 x 22</td>
<td></td>
</tr>
<tr>
<td>Weight (g)</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>Battery Capacity</td>
<td>&lt; 75 mAh</td>
<td>&lt; 75 mAh</td>
<td>&lt; 75 mAh</td>
<td></td>
</tr>
<tr>
<td>Sleep Mode</td>
<td></td>
<td></td>
<td></td>
<td>Sleep</td>
</tr>
<tr>
<td>Internet Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>McOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>MACH</td>
<td>MACH</td>
<td>MACH</td>
<td></td>
</tr>
<tr>
<td>RAM (MB)</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>The actual system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Speed (MHz)</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Battery Life (h)</td>
<td>&gt; 2000</td>
<td>&gt; 2000</td>
<td>&gt; 2000</td>
<td>&gt; 2000</td>
</tr>
</tbody>
</table>

### Sensor systems (4)

#### ETRI Smart Sensor Node

<table>
<thead>
<tr>
<th>Product</th>
<th>mica2</th>
<th>互连</th>
<th>Telos</th>
<th>ETRI-SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Crossbow</td>
<td>Intel Lab.</td>
<td>Telos Corp.</td>
<td>ETRI</td>
</tr>
<tr>
<td>Year</td>
<td>2001</td>
<td>2003</td>
<td>2003</td>
<td>2004</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>7MHz</td>
<td>12MHz</td>
<td>20MHz</td>
<td>8MHz</td>
</tr>
<tr>
<td>CPU</td>
<td>ATMel</td>
<td>ARM</td>
<td>Motorola</td>
<td>ATmel</td>
</tr>
<tr>
<td>Flash Memory (KB)</td>
<td>128</td>
<td>512</td>
<td>60</td>
<td>128</td>
</tr>
<tr>
<td>RAM (KB)</td>
<td>4</td>
<td>64</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RF Speed (KBaud)</td>
<td>40</td>
<td>460</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>Radio Type</td>
<td>Chipcon</td>
<td>Zeevo BT</td>
<td>Zigbee</td>
<td>Zigbee</td>
</tr>
<tr>
<td>Operating System</td>
<td>TinyOS</td>
<td>TinyOS</td>
<td>TinyOS</td>
<td>Nano-Quad</td>
</tr>
</tbody>
</table>

---

Intergiciels pour réseaux ad hoc et spontanés - © Guy Bernard / INT 2006
Wireless Sensor Networks architecture

- architecture classique d’un réseau de capteurs
  - données : massives, en flot (éventuellement), hétérogènes
    - aspects système et "sémantique"
  - comment répartir traitement et stockage des données entre les différents niveaux ?
    - architecture hybride :
      - avec serveur central (éventuellement mobile/volatile),
      - entre serveurs centraux aux niveaux supérieurs (ad hoc ou fixes)

Example 1 : GlacsWeb

- [Martinez et al., IEEE Computer 2004]

Capteurs Antarctique
Southampton
Example 2: TinyLIME

[Curino et al., JPMC 2005]

Fig. 2. Operational scenario showing one-hop communication between base stations (top) and sensors and multi-hop communication among base stations and clients (PDAs). Client agents can also be co-located with the base stations (e.g., running on the laptops).

Example 3: CarTel

[Balakrishnan et al., SenSys 2006]

Cars collect data as they drive, and log them to their local ICEDB databases. As connectivity becomes available, data on cars is delivered via CaFNet to the portal, where users can browse and query it via a visualization interface and local snapshot queries.
### Middlewares for sensor systems (1)

<table>
<thead>
<tr>
<th>Name</th>
<th>Lab</th>
<th>Functional properties</th>
<th>Non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILAN</td>
<td>Rochdale</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>Impala</td>
<td>Princeton</td>
<td>X X</td>
<td>Energy X</td>
</tr>
<tr>
<td>AutoSIC</td>
<td>UC Irvine</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>DSWare</td>
<td>Virginia</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X.Yu03</td>
<td>S. California</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>Y.Yu03b</td>
<td>S. California</td>
<td>X X X</td>
<td>Energy X</td>
</tr>
<tr>
<td>SensorWare</td>
<td>UCLA</td>
<td>X X X</td>
<td>Energy</td>
</tr>
<tr>
<td>TinyLime</td>
<td>Polli. Milano</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>EnviroTrack</td>
<td>UIC</td>
<td>X</td>
<td>Energy</td>
</tr>
<tr>
<td>SINA</td>
<td>Delaware</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- no single middleware addresses all the non-functional requirements
- security is not addressed
- **energy saving** is a major concern
  - issue: tradeoff between energy for computing / energy for communication / fault tolerance
  - example: Mica2 Motes: RX=16mA, TX=18mA, computing=8mA, sleeping=10uA
  - send a few bytes whenever a monitored value changes, vs. store, compute, send aggregate values

### Middlewares for sensor systems (2)

- **a french initiative: RFIDi (ObjectWeb)**
  - **RFID Middleware**
    - RFID Reader Interface
      - Provide Consistent Interface to Readers
    - Readers Management
    - Data Filtering and Transport
    - Multi-platform Data Request
      - Provide data for multiple hosts access
    - Legacy System Support
    - RFID Information Network
      - Sharing of RFID data between partners

- **Middlewares for sensor systems (1)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Lab</th>
<th>Functional properties</th>
<th>Non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILAN</td>
<td>Rochdale</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>Impala</td>
<td>Princeton</td>
<td>X X</td>
<td>Energy X</td>
</tr>
<tr>
<td>AutoSIC</td>
<td>UC Irvine</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>DSWare</td>
<td>Virginia</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X.Yu03</td>
<td>S. California</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>Y.Yu03b</td>
<td>S. California</td>
<td>X X X</td>
<td>Energy X</td>
</tr>
<tr>
<td>SensorWare</td>
<td>UCLA</td>
<td>X X X</td>
<td>Energy</td>
</tr>
<tr>
<td>TinyLime</td>
<td>Polli. Milano</td>
<td>X</td>
<td>Energy X</td>
</tr>
<tr>
<td>EnviroTrack</td>
<td>UIC</td>
<td>X</td>
<td>Energy</td>
</tr>
<tr>
<td>SINA</td>
<td>Delaware</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Middlewares for sensor systems (3)

- **ObjectWeb Stack and RFID**
  - Reader Interface
    - SensorBean, OSCAR
  - Reader Management
    - SensorBean
  - Data Filtering/Transport
    - JORAM, Proactive, XSquare, Octopus, C-JDBC
  - Data Request/Legacy System Support
    - JOnAS, Celtix, Petals
  - RFID Information Network
    - SNAP, Proactive

EU-FP6 research projects (1)

- **OZONE – IST-IP [FP5]**
  - "generic framework to enable consumer oriented ambient intelligence applications"
  - development of the WSAMI middleware (LGPL) – but restricted to one-hop networks
  - october 2001 – june 2004 (32 months)
  - ARLES – LORIA – SARDES – IRISA ...

- **CORTEX – IST-IP [FP5]**
  - "investigate architectures for the ubiquitous, decentralised, and proactive mission-critical computer systems"
  - march 2001 – september 2004 (42 months)

- **EYES (Energy Efficient Sensor Networks) – IST-IP [FP5]**
  - "develop architectures and technologies for building self-organizing, collaborative and mobile wireless sensor networks"
  - march 2002 – february 2005 (36 months)
  - U. Twente – TU Berlin – La Sapienza – CNIT - Infineon

- **AMIGO (Ambient Intelligence for the networked home environment) – IST-IP**
  - "middleware and intelligent user services for the networked home environment"
  - "development of open and interoperable middleware for a networked home system that combines home automation, CE, mobile and PC functionalities"
  - extension of WSAMI to MANETs
  - september 2004 – march 2008 (42 months)
  - Philips – FT – Fraunhofer – ARLES – ARES ...
EU-FP6 research projects (2)

- BETSY (BEing on Time Saves energy) – IST-STREP
  - “multimedia streams on wireless hand-held devices seamlessly adapted to fluctuating network conditions and available terminal resources while reducing the energy consumption of the stream processing”
  - september 2004 – march 2007 (30 months)

- CoBiS (Collaborative Business Items) – IST-STREP
  - “Items […] will have unique digital identities, embodied sensors to monitor […], communicate wirelessly and peer-to-peer, [and] collaborate”
  - “create a framework for the deployment of services across CoBiS networks”
  - August 2004 – February 2007

- COMPARE (COMponent Approach for Real-time and Embedded) – IST-STREP
  - “define a framework for RT/E systems, adapting the CCM for that purpose”
  - june 2004 – december 2006
  - THALES – CEA-LIST – Schneider – UP Madrid …

- RUNES (Reconfigurable Ubiquitous Networked Embedded Systems) – IST-IP
  - “large-scale, widely distributed, heterogeneous networked embedded systems that interoperate and adapt to their environments”
  - “development of adaptive component based middleware that is adaptive and intelligently self-organising”
  - September 2004 – May 2007 (32 months)

FP6 projects (2nd call, 2003) target mainly pervasive computing …
- … but have been set up in 2002/2003

National initiatives

  - “Universal Communications, New Generation Networks, New Security and Safety for the Ubiquitous Networked Society”
  - sensors and RFIDs, ad hoc networks, PAN (UWB)

  - ambition: 7% of world RFIDs / Ubiquitous Sensor Networks market by 2010

- USA: 2 NSF initiatives:
  - GENI (Global Environment for Networking Investigations)
    - new networking and distributed system architectures for PERVASIVE COMPUTING, mobile, wireless and sensor networks – 500 M$ – IST-STREP 2006 ONELAB project (LIP6) associated to GENI – 2ME
  - FIND (Future Internet Network Design)
    - vision of global network in 15 years - architecture, mobile wireless and sensor technologies
Standardization activities (1)

- activities about standardization for mobile computing
  - mobile middleware = execution environment + set of generic services (discovery, event notification)
  - Internet Engineering Tasks Force (IETF; http://www.ietf.org/) – most of the standard protocols
    - challenge: how are the numerous Internet protocols incorporated in the middleware?
  - Third Generation Partnership Project (3GPP; http://www.3gpp.org/) and Third Generation Partnership Project 2 (3GPP2; http://www.3gpp2.org/)
    - for middleware: select existing standardized solutions

- Parlay (http://www.parlay.org/) Open Service Architecture
- Wireless World Research Forum (WWRF; http://www.wireless-world-research.org/)
  - pre-standardization forum
- Open Mobile Alliance (OMA; http://www.openmobilealliance.org/)
  - specifies service enablers for the mobile world

Standardization activities (2)

- activities about standardization for mobile computing (cont.d)
  - Object Management Group (OMG; http://www.omg.org/)
    - Recent specifications
      - Wireless CORBA (OMG document: formal/2003-03-64)
      - Super Distributed Objects (OMG document: dtc/2003-04-02)
      - Smart Transducers (OMG document: formal/2003-01-01)
    - Model Driven Architecture (MDA; http://www.omg.org/mda/)
  - World Wide Web Consortium (W3C; http://www.w3.org/)
    - Web Services Activity
      - working groups for Web Services Architecture, XML Protocol (SOAP), Web Services Description, Web Services Choreography
    - Device Independence Activity
      - Device Independence Principles
      - document CC/PP1: Structure and Vocabularies
    - Semantic Web Activity
      - RDF Core Working Group, Web Ontology Working Group
  - Organization for the Advancement of Structured Information Standards (OASIS; http://www.oasis-open.org/)
    - Universal Description, Discovery and Integration (UDDI; http://www.uddi.org/)
Standardization activities (3)

- **activities about standardization for mobile computing** *(cont. d)*
    - Networking Ontology specification (provides the means to describe properties of connectivity)
    - Device Ontology Specification
    - several recent JSRs increasing the functionality of Java 2 Micro Edition (**J2ME**)
    - target of specification: digital and analog set top boxes, service gateways, cable modems, consumer electronics, PCs, industrial computers, cars...
  - **Liberty Alliance Project** ([http://www.liberty-project.org/](http://www.liberty-project.org/))
    - interoperability, privacy, guidelines, best practices
      - Liberty Alliance Version 1.1 Specification Suite: Federated Authentication to enable seamless sign-on
      - Liberty Alliance Version 2.0 Specification Suite: will feature Web Service Framework, authorizing e.g. a Service Provider to access your location information
    - addresses discovery and auto-configuration

"the standardization landscape is quite a mesh" [K. Raatikainen, 2006]
- IETF covers the Internet Protocol Suite but has a vast number of working groups
- middleware services are standardized here and there, including tens of forums
- major forums for wireless world: OMA, W3C, Liberty Alliance, OMG
Standardization activities (5)

**Super Distributed Objects (OMG)**
- Jan 00: creation of the SDO SIG (Hitachi, U. Tokyo, Sun)
- Jun 00: SDO RFI issued
  - answer to RFI by LIFL/Gemplus: "Communicating Mobile Objects"
- Jul 01: White Paper V1.0
- Feb 02: RFP and 1st public draft of the specification
- Mar 03: final draft issued: "PIM and PSM for SDO specification"

Standardization activities (6)

**Super Distributed Objects (contd.)**
- **goal**
  - enable interconnection of various terminal equipments whatever vendors and specifications be
  - targeted to ubiquitous computing (home and office)
- **technical model**
  - equipment can be described as clusters ➔ devices can be attached/detached without altering application program
  - standard interfaces ➔ only selected functions can be added/discard to/from an application program
  - common operation patterns and operation logic is defined as frameworks ➔ large scale systems can be upgraded or modified in stages in subsystems units
  - rights over equipment managed by the system ➔ multiple application programs can use the same device without competing
Major research teams

- **major teams abroad**
  - USA : MIT, CMU, Georgia Tech, UIUC
  - Korea : ETRI
  - Japan : DoCoMo

- **major teams in France**
  - ARLES INRIA Rocquencourt (V. Issarny)
    - ambient intelligence
  - ARES CNRS-INRIA Lyon (S. Ubeda / Frédéric Le Mouël)
    - protocols → middlewares
  - ACES IRISA (M. Banâtre)
    - Spontaneous Information Systems (F. Weis)
  - PARIS IRISA (T. Priol)
    - EPI-NETS: epidemic construction of peer to peer networks (A.-M. Kermarrec)
  - LSR-IMAG (A. Duda)
    - DRAKKAR : distributed infrastructure for dynamic composition of services
  - RD2P LIFL-IRCICA (D. Simplot-Ryl)
    - POPS : sensor networks

---

Case study: TinyLIME

- **G.P Picco, Politecnico di Milano [Curino et al, 2005]**
- **middleware for mobile data collection in sensor networks**
- **enables multiple mobile monitoring stations**
  - to access the sensors in their proximity
  - and share the collected data through wireless links
- **needed by the applications where**
  - the sensors are sparse and possibly isolated, and
  - location-dependent data collection is required
- **target: applications where mobile hosts move and gather data from sensors within scope**
  - example: reading only the average temperature sensed around a technician while he walks through a plant
- **extension for sensors of LIME, itself extension for mobility of Linda**
Background: Linda

Linda [Gelernter 1985] – for "standard" distributed systems
- shared memory model
  - memory = set of tuples ("tuple space")
  - tuple = sequence of typed fields (e.g., "<"foo", 9, 27.5>")
- coordination between processes through writing and reading tuples
- operations
  - **out(t)** add tuple t
  - **in(p)** remove one tuple following pattern p
    - actual value "<"foo", 9, 27.5>
    - formal value "<"foo", ?integer, ?float>
      - if several tuples match, 1 is non-deterministically selected
  - **rd(p)** non-destructive read
  - **in** and **rd** are blocking – operation suspended until a matching tuple appears → synchronization
  - asynchronous variants **inp** and **rdp** ("poll") – return null if no tuple matches
  - in some variants of Linda: **ing** and **rdg** ("group") retrieve all matching tuples at once

- tuple space is stored and managed at a well-known location, which is supposed to be always reachable
- processes interact by inserting tuples in the TS (**out** operation) and issuing **rd** and **in** operations to read and remove data from the TS
- example: producer-consumers
  - producer **outs** tuples describing jobs
  - consumers **in** jobs tuples based on patterns related to their capabilities
  - if needed,
    - results of job execution are **outted** by the consumers
    - retrieved by any process with **in**
Background: LIME

- **LIME [Picco at al, 1999] – Linda In a Mobile Environment (ad hoc)**
  - Communication in Linda is decoupled in **time** and **space**
    - Senders and receivers need not be available at the same time
    - Mutual knowledge of their identity and location is not necessary
  - Good model for mobile ad hoc environments
  - However, there is no location to place the tuple space so that mobile entities can access it at all times
  - LIME extensions to Linda:
    - Multiple tuple spaces (each attached to a mobile entity)
    - Transient sharing
      - Content of different spaces is shared only when entities can communicate
    - Mobile hosts and mobile agents
      - Agents can migrate for host to host with their own tuple space
    - Reactions
      - Code fragment executed whenever a tuple matching a pattern appears in the federated tuple space

TinyLIME

- **[Curino et al, 2005]**
- Extends LIME
  - By providing features and middleware components specialized for sensor networks
  - While maintaining LIME's coordination for ad hoc networks
- Implementation for Crossbow MICA2 mote platform
  - TinyOS
  - 128 KB program memory
  - 512 KB user data
  - Special board (gateway) for converting laptops to base stations (serial port)
**TinyLIME**

- **LIME adaptations for sensors**
  - a mote in direct communication with a base station (BS) is represented by an agent on this BS
    - associated tuple space contains the set of data provided by the sensors of the mote
  - reactions can be enriched with conditions
    - e.g., react only if \(20 < \text{temperature} < 30\)
  - time epochs
    - data may be useful only if recent enough – not necessary to transmit obsolete data
    - epoch = fixed time slice (system-wide, defined at deployment time)
    - data is recorded only once per epoch
    - each mote maintains an epoch count (since its boot time)
    - not a wall clock – only relative values are significant,
      e.g. "10 to 5 epochs ago from now"

- **data aggregation**
  - for sensors, communication is more expensive than computation and sampling
    - MICA2: RX = 16 mA, TX = 18 mA, computation = 8 mA, sleep = 10 uA
  - aggregation should be done on sensor nodes whenever possible
  - 2 extensions for data aggregation
    - enrich the format of tuples and templates with aggregation function (min, max, average, variance)
    - active and passive behaviour of sensors
      - active: autonomously and periodically, sample data on which aggregation is to be done
      - passive: sampling explicitly activated by the client program for a given number of epochs
TinyLIME architecture

**interaction between client and base station**
- sensor data is retrieved only on demand
  1. C: place a query tuple in "config" TS
  2. C: register a reaction in "motes" TS
  3. BS: react at query
  4. BS: retrieve data from sensors
  5. BS: post data in "motes" TS
  6. C: react to tuple arrival
  7. C: delivers data to client

**TinyLIME architecture**

**interaction between BS and motes**
- Aggreg & Logging: record sensor data that have been requested
- Filtered Comm: eliminates duplicates
- Core:
  - extracts sensor type and condition
  - gets suitable logged value or asks for a fresh one from appropriate sensor
  - assembles a packet
- Generic Comm: send/rcv to BS

**source code size (# lines)**

<table>
<thead>
<tr>
<th>Component</th>
<th>Language</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>moteLimeTupleSpace</td>
<td>Java</td>
<td>623</td>
</tr>
<tr>
<td>moteAgent</td>
<td>Java</td>
<td>587</td>
</tr>
<tr>
<td>TDMoteAccess</td>
<td>Java</td>
<td>701</td>
</tr>
<tr>
<td>On-Mote Components</td>
<td>nesC</td>
<td>1050</td>
</tr>
</tbody>
</table>

Figure 6. Architecture of components installed on motes. Shaded components have been developed for TinyLIME, while the others are provided by TinyOS.
TinyLIME: conclusion

- departs from the traditional setting where sensor data is collected by a central monitoring station
- adapted to applications involving static sensors and mobile devices
- powerful and flexible programming model
- context-awareness provided by communication with reachable sensors only
- energy saving by
  - requesting sensor data only when needed
  - use of sleeping / awake modes of motes
  - data aggregation functions on motes rather than on base station
- references

Conclusions (1)

- nomadic systems
  - the pervasive computing domain has been heavily investigated
    - a lot of projects, especially european and asiatic
    - the simplest domain (plenty of resources, restricted mobility, central server available)
- ad hoc systems
  - partial solutions
    - more complex domain (lack of resources, full mobility, full decentralization needed)
    - limited resources not always taken into account (energy)
    - security deserves more attention
  - domain just going out of protocol layers
    - example: Michaël Hauspie's PhD, Lille, January 2005)
    - opportunity for french/european teams
- sensor systems
  - very complex domain (from electronics to applications)
  - a lot of activity
    - "6K papers on sensor and actor networks written the last 5 years – a lot of epsilons" (I. Akyildiz, March 2006)
    - opportunity for the french/european teams which own the complete chain of skills
Conclusions (2)

- **main challenges**
  - ad hoc systems
    - decentralization, decentralization, potential contribution of peer-to-peer techniques
  - sensor systems
    - how to distribute computing/storage between sensor nodes and “server”
      - tradeoff between energy for computing / energy for communication / fault tolerance
      - send a few bytes whenever a monitored value changes vs. store, compute, send aggregate
  - genericity vs. feasibility
    - to which extend can a generic middleware support application design/execution?
      - tradeoff between development time and code efficiency
  - standardization
    - will an equivalent of JavaCard for ad hoc / sensor systems emerge?

References (1)

References (2)