Problématique de la mobilité dans les systèmes distribués

Guy Bernard
Institut National des Télécommunications, Evry, France

Guy.Bernard@int-evry.fr

Département Informatique - Equipe "MARGE"

- "Middleware pour Applications Réparties avec Gestion de l’Environnement"
- 6 academics, 4 PhD students, 2 post-doc
  - QoS support for multimedia distributed applications
  - mobile agents for fault tolerance in telecommunication systems
  - load balancing policies for large scale distributed services
  - adaptation of distributed applications to execution environment variability
    - at middleware level (object- and/or component-based)
    - adaptation to network connectivity variability in mobile environments
      - disconnection management
      - copy consistency management
      - online multi-players games on mobile phones
    - dynamic context-aware deployment and adaptation
      - deployment at start time
      - adaptation at run time
- member of SAMOVAR CNRS team (UMR 5157)
- Action Spécifique CNRS AS19 "Accès aux données / mobilité" 2001-2002
- Action Spécifique CNRS AS150 "Systèmes répartis et réseaux adaptatifs au contexte" 2003-2004
- Steering Committee of Réseau Thématique Pluridisciplinaire CNRS N°5 "Systèmes Répartis"
**Prologue: Motivation**

- today, more handheld devices than static PCs

![Internet Connectivity Outlook](image)

**INTERNET CONNECTIVITY OUTLOOK**

More handsets than PCs connected to the Internet by the end of 2003!

**Where are the computers?**

- computer science today is focused on 2% of computers
- in the near future, all these computers will be able to communicate together

![Computer Science Focus](image)

[Where has computer science focused?](image) [Where are the processors?](image)

[D. Tennenhouse, CACM 43(5), 2000]
Communicating devices in the future

- larger and larger number of communicating devices
- smaller and smaller devices

Example: Smart Dust

- SPEC (Univ. of California at Berkeley) wireless sensor on a chip
  - micro-radio
    - 902 MHz over 40 feet at 19,200 kbit/s
    - uses 1000 times less power than a cell phone.
  - analog-to-digital converter
  - temperature sensor
  - TinyOS operating system
Challenges (1)

- handling heterogeneity
  - a variety of devices
    - very heterogeneous devices should be able to communicate
    - from tiny sensor to supercomputers
      - Golem Dust (UCB, 2004)
        - solar powered mote with bi-directional RF communications and sensing (acceleration and ambient light)
        - 6.6 mm3 total circumscribed volume

- www.top500.org (June 2006)
  - constellation of 131072 processors, 280600 GFlops

<table>
<thead>
<tr>
<th>rank</th>
<th>manuf.</th>
<th>name</th>
<th>Gflop/s site</th>
<th>#proc</th>
<th>processor</th>
<th>OS</th>
<th>system type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IBM</td>
<td>BlueGene/L</td>
<td>280600 LLNL</td>
<td>131072</td>
<td>PPC440-700MHz</td>
<td>CN/Unix</td>
<td>MPP</td>
</tr>
<tr>
<td>5</td>
<td>Bull</td>
<td>Tera-10</td>
<td>55705 CEA</td>
<td>8704</td>
<td>Itanium2-1.6-GHz</td>
<td>Linux</td>
<td>constellation</td>
</tr>
</tbody>
</table>

- a variety of network technologies
  - WiFi, BlueTooth, GPRS, UMTS, Zigbee, …

Challenges (2)

- handling very large number of devices
  - how to name them? how to locate them?

- handling mobility
  - most devices will be mobile
    - PDAs, smartphones, sensors …
  - devices can appear / disappear in an area when moving
    - the group of communicating devices changes dynamically
    - how to discover reachable devices? how to detect the disappearing of devices? how to distribute computing/storage between them?

- handling heterogeneous data
  - coming from a variety of devices

- handling continuous data flows
  - coming in a continuous way (sensors),
  - or because of the nature of applications (multimedia)
Parenthesis 1: smart card capabilities

- the smart card IS a real computer
  - processor, memory, custom OS
  - JVM - Javacard 2.2
- current generation:
  - 128K ROM, 64K EEPROM, 4K RAM
  - acts as a slave (server) answering requests from the (client) host terminal
- prototypes
  - Gemplus, 1999: 2 MB of memory
  - Gemplus, 2001: 224 MB of memory
  - keyboard, screen
  - software: SC can be active in the SC -- host terminal interactions

- for SC manufacturers, the tomorrow's computer is the SC
  - Smart phones are just a SC plus a keyboard, a screen and an antenna
- for SP manufacturers, the tomorrow's computer is the SP
  - SC is just used for persistent storage and authentification

Parenthesis 2: Smart Phone

- at least 3 processors: DSP - Control - SIM card
- how to distribute processing between Control and SIM ??
  - SP providers and SC providers are doomed to cooperate
**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 always mobile, large scale, wireless connection with a server

**What is mobile ??**

- user mobility
- terminal mobility
- user mobility with mobile terminal
- code mobility
- network mobility
- case studies
- perspectives

- not considered here:
  - telecommunication infrastructure/services supporting terminal mobility (GSM, WAP, GPRS, UMTS, ...)
  - protocols for supporting mobility (Mobile IP, ...)
  - database aspects (disconnected transactions, ...)
  - adhoc and sensor networks
User mobility (1)

- **Problem**
  - Users are increasingly moving - wish to feel "at home" anywhere
    - Office desktop - laptop - home PC - smartphone/PDA - airport - hotel - ...
    - Need for retrieving one's environment on any machine
      - Programs, preferences, data

- **Tools and solutions**
  - File systems
    - CODA (S. Satyanarayanan, CMU, 1990), Ficus (G. Popek, UCLA, 1990), Bayou (P. Petersen, Xerox, 1995)
    - Automatic hoarding (G. Popek, UCLA, 1997)
  - Using a smart card as a portable computer/data repository
    - RNRT "CESURE" project (Gemplus - INRIA Grenoble - LIFL - INT)
      - Example: tele-meeting
    - Grenoble University campus project (pool of non-personal desktops)
      - User preferences / accounting

User mobility (2)

- **Corollary 1: Handling heterogeneity**
  - Processors, OSes
  - HMI
    - Screen size - keyboard - pointer - mouse - voice command ...
  - Current solution
    - Component-based distributed software
      - Clear separation of application engine and application HMI
      - Dynamic loading of appropriate HMI component

```
ForecastDisplay Interface  ForecastData Interface

UserInterface  WeatherConsult  WeatherServer

run  init  uses  provides  uses  provides
```

- Example: RNRT "CESURE" project: "intelligent" trading
User mobility (3)

- corollary 2: very large scale systems
  - world-wide services (naming, location, ...)
  - robust and scalable algorithms (IP routing, DNS, ...)
  - example
    - GLOBE project (Tanenbaum, VU Amsterdam, 1996)
  - large scale load balancing between distributed servers
    - current algorithms are not scalable
    - synchronous communications, centralized load balancers
    - suited for a large number of hosts on a high speed / short range network - eg., AOL Web cache (> 1,000 nodes) - MSN Hotmail (> 5,000 nodes)
    - need for decentralized / asynchronous algorithms
    - example: OMG’s Load balancing and Monitoring RFP, document orbos/2000-04-27
    - open problem: large scale copy consistency for modifying access patterns

User mobility (4)

- corollary 3: location-dependent data
  - added-value services (weather forecast, restaurants, taxis, ...)
  - solution: extend trading service
    - for static terminals, the geographical position is known
    - if (ll<latitude<l2) && (ll<Longitude<l2) then serveur = Paris
Terminal mobility (1)

- **mobile terminal: PDA / Smart Phone** with wireless network access
  - limited resources
    - CPU, memory, screen, IHM, battery
    - the real limitation is the energy
      - 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) – flash memory
      - 100g battery = 1g gas
  - wireless network access
    - network access is costly (battery - particularly on emission, money??)
      - favour disconnected mode - delay and/or group communications
    - bandwidth is variable
    - disconnections are "normal" events
      - fixed terminals / wired network
        - unreachability is considered as a failure and handled as such
      - mobile terminals / wireless network
        - unreachability is frequent - being up is a failure :-(

Terminal mobility (2)

- **implications**
  - contradictory demands on system design
    - favoring reliance on servers
      - resource-poor clients
      - poorer security and robustness of clients
    - favoring ability to operate alone
      - you may not be able to contact servers
      - communication may be expensive
  - hence mobile systems must be adaptive
    - rely on servers when possible
    - function autonomously if needed
    - monitor and adjust to current conditions
      - the system must dynamically react to a decrease in resource availability and take benefit from an increase
Terminal mobility (3)

- **adaptation strategies**
  - **laissez-faire**
    - adaptation is entirely the responsibility of individual applications
    - **pros**
      - no system modification is required
      - applications get precisely the adaptation behavior they want
    - **cons**
      - does not support application concurrency
      - each application can only observe the part of the resource that it uses
      - incompatible decisions can be taken
        - eg., 2 simultaneous videos, increasing bandwidth being possible only for one

Terminal mobility (4)

- **adaptation strategies**
  - **application-transparent**
    - the system bears responsibility for both adaptation and resource management
    - **pros**
      - allows legacy applications to run unmodified
      - does not complicate the programming model of the applications
      - provides a central point of control for shared resources
    - **cons**
      - applications handling the same data type may wish to make different adaptation decisions
        - eg., video player: reduce quality, preserve timing -- scene editor: reduce timing, preserve quality
      - example: CODA (Satyanarayanan, CMU, 1990)
        - a file system proxy hides mobile issues from applications and emulates file server services on the mobile terminal
        - cache management: hoarding, emulating, reintegrating
Terminal mobility (5)

- adaptation strategies
  - application-aware
    - collaborative partnership between the system and individual applications
      - the system is best positioned to know what is available at the mobile terminal
        - monitoring, allocation decisions, usage optimisations of shared resources
      - the applications are best positioned to know their exact needs
        - must be informed by the system of significant changes in res. availability
        - can react to changes in their own way
    - example: Odyssey (Satyanarayanan, CMU, 1997)
      - dynamic client-side adaptation of data-intensive applications (eg., audio/video)
      - several levels of fidelity for remotely accessed documents
      - bandwidth monitoring and ways to switch between fidelity levels
      - modifications of the underlying OS (addition of signals and system calls)
      - does not handle network disconnections
      - recent work: adaptation to available energy

Odyssey
Terminal mobility (6)

- adaptation of applications to disconnected modes
  - 2 kinds of disconnections
    - involuntary (shadow, handoff)
    - voluntary (save battery or money, jumping into a plane, ...)
  - example 1: Rover (Gifford & Kaashoek, MIT, 1995)
    - Relocatable Data Objects
      - application is made of “data objects” which can be dynamically duplicated from server to terminal
  - Queued RPCs
    - asynchronous RPCs
    - requests and replies are buffered (in stable storage) on terminal and server
    - queues are drained whenever possible (connection available)
  - example 2: DOM (Conan et al., INT, 2002)
    - how to adapt distributed object-based CORBA-compliant applications to disconnected modes
    - with minimal changes to legacy code
    - tools: Object by Value & Portable Interceptors

Rover

- diagram of Rover system
  - Object Cache
  - Access Manager
  - Operation Log
  - Network Schedules
  - Mobile host
  - Server
Disconnection Operation Management (DOM)

Terminal mobility (7)

- **standards**
  - OMG’s RFP Wireless Access and Terminal Mobility, document telecom/99-05-05 ("Wireless CORBA")
    - Nokia - Vertel - HighComm - Inprise
    - GIOP mapping onto Internet transport protocol (TCP or UDP) over wireless links
      - WAP Wireless Transaction Protocol (WTP)
        - mechanism for initial access to a new mobility domain
        - mechanism for service discovery in a mobility domain
        - mechanism for advertising CORBA services available on a mobile terminal
        - mechanism for handoff between mobility domains
Parenthesis: network world’s vision (1)

- **system community vision of adaptability**
  - the terminal (application/system) has no action on the network’s behaviour
  - the software (terminal and maybe servers) has to adapt itself
    - eg., available bandwidth
- **network community vision of adaptability**
  - the network has no action on the users’ behaviour
    - eg., number of simultaneous sessions of VoD
  - trend in network infrastructure: self-awareness, including:
    - self-provisioning: ability of Network Elements to allocate, configure, maintain network resources in order to process service requests
    - Cisco IE2100, Nortel SELFCON, Telcordia NESTOR
    - self-monitoring: ability of NE to monitor the usage of its network resources
      - Cisco Netflow, Ipanema
    - standardization bodies
      - IETF (Zeroconf WG), Universal Plug&Play

Parenthesis: network world’s vision (2)

- **these 2 visions are complementary, not contradictory**
  - the network has to do its best for satisfying users demand ...
    - … but it will not be able to satisfy all demands !!
    - thus the “terminal” has to be aware of (some) changes in network context and react to them
      - probably with user’s implication - some examples :
        - choose between B/W images or mute sound
        - choose between stopping an application or continue in degraded mode
      - some changes in context are on the terminal side
        - the user decides to switch to a least quality at the benefit of a lower price
        - the battery on the mobile terminal is almost exhausted
  - some concerns are common
    - monitoring usage bandwith for billing
    - abnormal terminations
  - some cooperation is needed
    - what has to be done by the network ? by the “terminal” ?
**User mobility with mobile terminal (1)**

- the wireless-connected terminal moves when the user moves
- the data accessed can be location-dependent
  - accessing location-dependent data assumes location determination
  - terminal location changes in time => location determination must be dynamic
    - requires a support from telecom provider / GPS / …
  - survey of location systems for ubiquitous computing: IEEE Computer, August 2001
    - GPS
      - 24+3 satellites - accuracy: 1-5m (95-99%) - not indoor
    - E911
      - US Federal Communications Commission initiative
      - wireless phone providers must develop a way to locate any phone that makes a 911 emergency call
      - many companies are developing a variety of location systems to determine a cellular phone’s location
      - these systems will also support new consumer services (find the nearest gas station, post office, or automated teller machine)

---

**User mobility with mobile terminal (2)**

- a lot of applications
  - eg., guiding firemen, or search-and-rescue team for the best route to quickly locate all the avalanche victims)
- tentative solution: trader federation
  - a lot of opened problems, eg.
    - where are the traders located?
    - how do they cooperate?
    - how many traders?
    - how does the terminal knows where are the traders?
    - …
**Code mobility**

- referenced code
  - RPC, RMI, HTTP/CGI - synchronous
- remote evaluation
  - Java servlets - synchronous - code is pushed
- on-demand code
  - Java applets - asynchronous - code is pulled
- mobile code (mobile agents)
  - code and state move together - asynchronous
  - neither "push" nor "pull" model - rather, self mobility (proactive migration)
    
    ```
    ...instruction i ; go ("somewhere") ; instruction i + 1 ; ...
    ```
  - a network connection is needed only during agent transfer between sites
  - survey of mobile code system-related problems: Fuggetta et al., 1998
- fits well with wireless terminals and terminal mobility
  - application code can jump from a nearby server to current user location
  - is proactive migration (mobile agents) really needed??

**Network mobility**

- constellations of LEO satellites (eg., Teledesic)
  - viewed by satellites, even mobile users are almost still
  - dual problem of mobile users within a network of static servers
    - if satellites embed applicative code, this code moves over users
      - code jumping to user terminals when seen by a satellite??
  - no outstanding research work in this area
Case studies

- handling user mobility
  - CESURE
- application-aware adaptation
  - to bandwidth variability
    - ODYSSEY
  - to network disconnections
    - ROVER
    - DOM
- middleware for sensor systems
  - TinyLIME

User mobility: RNRT CESURE Project

- Modelling and exploitation of the concept of service application to mobile users of the network

- Service application
  - distributed application providing an added-value service to users

- Mobile users
  - users can connect to the service from anywhere, any kind of terminal (including mobile terminals: smart phones, PDAs, ...)

- November 1999 - November 2001
- Gemplus - INRIA Grenoble - LIFL Lille - INT
- funded by Ministry of Research
CESURE: Objectives

- provide tools and system infrastructure for mobile applications enabling
  - building added-value services
  - creating dynamically instances of applications adapted to the user
- architecture based on 2 types of elements
  - downloadable and configurable components
  - remote services

Based on a smart card acting as a central point for the application

CESURE: Approach

- application design according to a modular approach (distributed software components)
- application description in a smart card
- deployment and execution platform for instantiating the configuration the best suited to the context
CESURE: Component-based application

- **example application: weather forecast**
  - available anywhere, from any terminal type
  - the user gets weather forecast:
    - relative to his/her current location
    - according to service subscription options (3 days, terrestrial, in french)
- **adaptation requirements: the application configuration must be adapted**
  - to the terminal (HMI, processor/network characteristics, ...)
  - to the current user location
  - to subscription parameters
  - to the state of the environment (load of servers, ...)

 CESURE: Application layout

- **WeatherServer**
  - provides raw forecast data
- **WeatherConsult**
  - receives raw forecast data and adapts them according to subscription parameters
- **UserInterface**
  - displays the data according to terminal type
CESURE: Component description

```xml
<component name="WeatherServer">
  <provides type="ForecastData" name="data" />
</component>

<component name="WeatherConsult">
  <provides type="ForecastDisplay" name="display" />
  <uses type="ForecastData" name="data" />
</component>

<component name="UserInterface">
  <uses type="ForecastDisplay" name="display" />
  <run name="run" />
</component>
```

- More information has to be associated to components -- eg, some description of provided functionality

Application description

```xml
<deployment name="WeatherForecastService">
  <property name="default_directory" value="dir.meteo.com:3000" />

  <instance type="WeatherServer" name="meteo" lifecycle="existing" />
  <instance type="WeatherConsult" name="client" />
  <instance type="UserInterface" name="hmi" />

  <constraint "hmi.host = 'localhost'" />
  <constraint "client.host = 'any'" />

  <connexion from="client.display" to="hmi.display" />
  <connexion from="client.data" to="meteo.data" />
</deployment>
```
Application deployment - Installation script

default_directory = "dir.meteo.com:3000"
params.directory = default_directory
    // looking for existing components
meteo = locate_component ( params, "WeatherServer" )
    // instanciation of new components -- local or remote
params.host = 'localhost'
hmi = create_component ( params, "UserInterface" )
params.host = 'any'
client = create_component ( params, "WeatherConsult" )
    // component connexion
connect ( BASIC, client.display, hmi.display )
connect ( BASIC, client.data, meteo.data)
    // service customization
client.init ()
    // application starting
hmi.run ()

NB - this is a generic logical description of the application -- some params (eg, terminal type, will be determined at deployment time)

CESURE: Application interface for iPAQ
CESURE: Application interface for PC

CESURE: evaluation

- **main contribution**
  - provide a generic architecture for the deployment of distributed component-based applications
    - dynamic
      - just-in-time deployment
      - adaptive to
        - user terminal type
        - user preferences / subscription parameters
        - to the state of the environment (load of servers, ...)
  - relies on a smart card and trading service (CORBA) extensions
    - the smart card is not mandatory - major benefits: security - data accessible even when no network connection is available
    - could be replaced by some file

- **limitations**
  - the deployment engine is not fully generic yet - no ADL
  - adaptation to user location not taken into account yet
**Odyssey**

- School of Computer Science, CMU (Satyanarayanan 1997)
- execution environment for applications needing some kind of adaptation for accessing remote data from mobile terminals
- design principles
  - **minimalism**
    - minimal set of extensions to an existing OS: NetBSD
  - **provide a mechanism rather than fixing a policy**
    - applications are free to decide the fidelity of data according to environment conditions
    - Odyssey provides the mechanisms for observing the state of the environment
  - **end-to-end argument conformity**
    - functionalities or knowledge that are necessary for external levels are duplicated at internal level only if strictly necessary (e.g., resource estimation for enabling coordination between concurrent applications)
- inherits of AFS and CODA (design philosophy, some code)

**Odyssey: notion of "fidelity"**

- "fidelity": degree to which the copy of data presented for use at a client matches the reference copy at a server
- many dimensions
  - consistency: applies to all data items
    - especially suited for DB of file systems (deltas can be easily expressed)
    - quality of data is type-dependent => other dimensions are also dependent on the type of data to be degraded
    - video: frame rate, image quality of individual frames
      - reduced frame rate => reduced bandwidth requirements
      - increased compression => reduced BW, increased processing
    - maps: feature sets (eg., keep only rivers and roads), minimum feature size (eg., keep only divided highways or larger roads)
  - dimensions = axes along which mobile clients can adapt resource usage to fluctuating supply and demand
    - changes in one or more dimensions => changes in resource consumption
    - if informed of resource availability, the client can take the appropriate decisions for adapting data access
Odyssey: global view

- **fidelity is a type-specific notion**
  - Odyssey must have some way of knowing the type of objects used by applications -- through the file system

- **the applications must know the current state of their environment**
  - Odyssey must provide applications with a way to name the resources

- **the applications must be able to tell the system which resource changes would be significant**
  - API extension (resource request)

- **Odyssey must notify applications when significant changes occur**
  - resource monitoring and asynchronous mechanism (upcall) by a global component ("Viceroy")

- **fidelity changes can depend on data type**
  - one component ("Warden") by data type, in charge of type-specific operations and of communication between client and server
  - general mechanism ("type-specific operation")

Odyssey: evaluation

- **example of "application-aware" adaptation on client side**
  - the system provides applications a mechanism for:
    - expressing their needs (special API: tsop)
    - monitoring resource usage and asynchronous notification
  - the applications can build a policy over this mechanism

- especially suited to data-intensive applications in which fidelity takes an intuitive meaning (video, images, sound)
  - how to define "fidelity" between database replicas and switch transparently between consistency algorithms levels??

- accommodates BW variability -- no support for handling disconnections

- relies on a modified OS
Rover

- Lab. for Computer Science, MIT (Kaashoek 1997)
- software toolkit for designing and executing client-server distributed applications in mobile environments
- supports a set of programming and communication abstractions that enable the construction of both
  - mobile-transparent applications: proxies for system services, that hide the characteristics of the mobile environment
  - mobile-aware applications: use these abstractions in the application code
- client-server architecture
  - client applications on mobile devices - servers on stationary hosts
  - communication between 2 clients on the same mobile or between one mobile and one server

Rover: key abstractions

- 2 abstractions
  - relocatable dynamic objects (RDOs)
    - objects (code & data) that can be dynamically loaded (copied) between a server and a client, in both directions
    - goal: reduce client-server communications
  - queued remote procedure call (QRPC)
    - comm system enabling non-blocking RPCs even if a host is disconnected -- requests and responses are exchanged upon network reconnection
    - goal: support disconnections
- programmer’s task
  - define RDOs for the data types handled by the application and for data transported between client and server
  - partition the code in 2 parts (CL & SV) that communicate by QRPCs
  - define methods for updating objects (including code for conflict detection and resolution)
- application - runtime cooperation: import objects - invoke methods on these local objects - export logs of method invocations - reconcile CL & SV copies
Rover: global architecture

Rover: RDOs and QRPCs

- all RDOs have a primary copy on the server
- clients import RDOs in a local cache and export modified RDOs to the primary server
- RDOs are migrated using QRPC
- QRPC invocation
  - the call is queued in a local stable support ($log$ = UNIX file)
  - control is immediately returned to the application
  - Rover invokes a callback routine when the result is available
- the log is managed by the Network Scheduler during connection phases
- QRPC supports a split-phase communication model
  - if mobile disconnected during request and reply, the server tries periodically to contact the client to deliver it the reply
- stable log of RPCs + separation in 2 phases $\Rightarrow$ the mobile can be shut off while awaiting for results
Rover: evaluation

- example of “application-aware” adaptation on both client and server side
- Odyssey focuses on the adaptation of “fidelity” of server data according to BW variations
  - ingredients: mechanism for expressing the application’s needs and asynchronous upcalls
  - applications can use this mechanism to design their policy for accessing remote data
- Rover focuses on the means to support concurrent updates on server and client sides in case of network disconnections
  - ingredients: object cache, object migration and QRPCs for accommodating mobility
  - mechanism for detecting conflicting updates
  - applications can use this mechanism to design their consistency policy between primary copy and local copy
- main drawbacks
  - application programmers must design the division of functions between client and server and “think” in terms of RDOs and QRPCs
  - proprietary approach - does not rely on a standard distributed object architecture

Disconnected Operations Management (DOM)

- Computer Science Department, INT (Conan 2002)
- Objective
  - Provide a framework for the adaptation of distributed applications
  - Support of voluntary and involuntary disconnection
    - Voluntary: decided by the user
      - For saving money and energy
      - For running the application when no wireless connection is possible
      - For minimising the probability of unexpected disconnection
    - Involuntary: during unexpected connection breakdowns
- Design criteria
  - End-user perception
    - Simple interface for voluntary disconnection / reconnection
    - Transparent switching between modes for involuntary disconnection
  - Compliance to OMG CORBA 2.4
- Part of the VIVIAN ITEA project 2000-2002 (Nokia, Philips, HUT, …)
## DOM: global view

- **Client / server applications without DOM**
  - "GUI" on the mobile terminal and servers on the wired network

- **Keeping working when disconnected**
  - Creation of a local copy of the server objects on the mobile terminal
    - How: State transfer
    - When: Connection monitoring

- **Switching between modes**
  - Transparently or programmatically

- **Reconciling after the reconnection**
  - Logging of operations executed when disconnected
    - Reconciliation by transmitting the operations of the log
    - Which operations: Design of the local copy

## DOM: CORBA implementation (1)

- **CORBA Portable Interceptors**
  - Hooks into the ORB through which ORB services can intercept the normal flow of execution of the ORB
  - Very few modifications of the server part in order to "tag" the server objects that can have a local copy on the mobile terminal
    - CORBA Policy of Portable Object Adaptor
**DOM: CORBA implementation (2)**

- using Portable Interceptors for transparent switching

**DOM: evaluation**

- example of "application-aware" adaptation on (mainly) client side
  - with a small impact on server side (for state transfer)
  - through a generic framework
  - can be applied to adapt standard distributed object-based applications
    - currently CORBA
    - other middleware supporting an "interceptor" notion could be considered too
  - as in Rover, deciding which remote objects must be copied on the mobile terminal is left to the application programmer
  - same for reconciliation of copies
  - focus on handling voluntary and involuntary disconnections
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

  - 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: inp and rdp ("group") retrieve all matching tuples at once
Background: Linda

- 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 outed 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

Figure 1. In Looz connected mobile hosts transiently share the tuple spaces of the agents executing on them.
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”*

---

**Fig. 2.** Operational scenario showing one hop communication between base stations (laptops) 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).
TinyLIME

- **LIME adaptations for sensors**
  - **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>moteLinuxTupleSpace</td>
<td>Java</td>
<td>635</td>
</tr>
<tr>
<td>moteAgent</td>
<td>Java</td>
<td>587</td>
</tr>
<tr>
<td>TCSEndPointAccess</td>
<td>Java</td>
<td>704</td>
</tr>
<tr>
<td>On-Mote Components</td>
<td>netC</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.](image)

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
Perspectives: reflective middleware ? (1)

- **Reflection**: "allows a program to access, reason about, and alter its own interpretation"
  - first emerged in the programming language community, to support the design of more open and extensible languages [Kiczales, 1991]
  - 2 (complementary) styles are being used in middleware platforms
    - structural reflection
      - underlying structure in terms of objects or components (interfaces, connectors)
      - applicability to mobility: context information (physical location, battery level, network performance)
        - example: insert or not a compression module for network communications
    - behavioral reflection
      - activity in the underlying system
      - applicability to mobility: arrival and dispatching of invocations (interceptors, dynamic proxies)
        - example: disconnection management

Perspectives: reflective middleware ? (2)

- **Example 1: CARISMA (W. Emmerich, UCL)**
  - **Application profile**
    - meta-information which associates, per application, policies (MW behavior) to context configurations - encoded in an XML document

![Diagram](image_url)

Fig. 1. User and application profiles. Fig. 4. Roles and Responsibilities in the Reflective Process.
Prospects: reflective middleware ? (3)

- changes in execution context
  - `<RESOURCE name="battery">`
  - `<STATUS operator="lessEqual" value="x">`
  - `<BEHAVIOR policy="disconnect"/>
  - `</RESOURCE>`

- service request
  - `<SERVICE name="accessData">`
  - `<BEHAVIOR policy="copy">`
  - `<RESOURCE name="memory" <STATUS operator="greaterEqual" value="x">`</RESOURCE>
  - `</BEHAVIOR>`
  - `<BEHAVIOR policy="link">`
  - `<RESOURCE name="bandwidth" <STATUS operator="greaterEqual" value="y">`</RESOURCE>
  - `<RESOURCE name="memory" <STATUS operator="less value="x">`</RESOURCE>
  - `</BEHAVIOR>`
  - `</SERVICE>`

![Fig. 3. Application profile.](image_url)

Prospects: reflective middleware ? (4)

- example 2: ReMMoC (G. Blair, Lancaster)
  - ability to develop applications independently of MW technologies and to adapt MW behavior to context information
  - 2 key components frameworks for binding and service discovery
    - binding framework
      - can be re-configured between different interaction protocols (CORBA, SOAP, publish/subscribe, ...)
    - service discovery framework
      - can change between one or more different service discovery technologies (SLP, Jini, UPnP, ...)
  - information obtained from service discovery == reconfiguration of the binding framework to the appropriate interaction protocol
  - current status
    - prototype based on their own "OpenCOM Light" component model
    - extension of Microsoft COM for Windows CE
    - SOAP, IIOP and publish/subscribe implemented, SLP and UPnP on the way
Conclusion

- **wireless terminal mobility raises old problems** …
  - resources are scarce (especially energy)
  - networks are slow and unreliable
  - … but in a quite new context
  - very large scale, terminals are moving => classical client-server model is no longer valid
  - network disconnections are the rule and no longer exceptions

- **user mobility and ubiquitous access raise new problems**
  - need for retrieving user’s environment anywhere, anytime
  - heterogeneity is the rule

- **key word: adaptation**
  - to current user terminal
  - to current user location
  - to bandwidth variations
  - to network disconnections
### Bibliographie (1)


### Bibliographie (2)