Context-Aware Mobile Instant Messenger
(支持情境感知的行動式即時訊息系統)

王卓立

March 13, 2008

香港大學計算機科學系
Agenda

Part I: Pervasive Computing
Part II: HKU Sparkle System
Part III: Context-Aware Mobile Instant Messenger
  - Motivations
  - Three features of our Mobile IM (MIM):
    - Significant location extraction
    - Cooperative place annotation
    - Context-aware presence management
  - Implementation & Evaluation
  - Conclusion and Outlook
Part I: Pervasive Computing

普及運算 or 普適计算
Pervasive/Ubiquitous Computing

無所不在的運算, 隨處運算

- Xerox PARC 科學家 Mark Weiser 1991年提出.
  - 計算與環境融為一體, 信息與計算“唾手”可得
  - *aims to reduce the ”excitement” of information overload -- Calm Technology (Mark Weiser)*

- Other terms:
  - *Invisible Computing*
  - *Unremarkable Computing*: 不值得注意的
  - *Context-aware Computing*: 情境感知的
  - *Ambient (周遭的) Intelligence*: (Philips, 1998)
  - *Everyware (2004: Adam Greenfield)*
Research Work on PvC

US
- MIT: Oxygen
- CMU: Aura
- Illinois: Gaia (Active Space)
- OGI and GIT: InfoSphere
- UC Berkeley: Endeavor
- U. of Washington: Portolano
- HP Cooltwon

Europe
- Ubiquitous Computing in Europe
- 英国: Equator

Read C.L. Wang’s Webpage on Ubiquitous/Pervasive Computing: http://www.cs.hku.hk/~clwang/pervasive.html
PvC is “User Centric” (以「人為中心」的計算)

目前的桌面計算模式：

- Difficult to use. required us to interact with them on their terms, speaking their languages
- 計算機佔據主導地位，人是計算機的“僕人”
  - 人必須處理各種計算任務的細節才能獲得計算和信息服务；
  - User Interface適合機器而不是人;
  - 人必須處理各種計算任務的細節才能獲得所需結果，比如硬件軟件安裝、需記住數據的存放地點等

人的注意力被計算設備所佔據, 而不是要完成的任務
Pervasive Computing (PvC)

Extends both the **time** and **space** scales of computation

“Might happen **anytime** anywhere, last **any duration**, span **any number** and **type of devices**, and which could offer **exciting services you never have conceived before**”.

計算不是固定的，而是隨時可移動的
計算資源是共享的，而不是私有的

無縫移動 + 365天/24小時 永不間斷服務
A Day in a PvC Age

Everything becomes **computerized**

Everything becomes **smart** (情境感知力)

- Your coffee pot “thinks” you are tired and prepares coffee

They can **communicate** with each other

- The refrigerator, coffee pot, sofa, and fan, talk and cooperate with each other to make your life better.
Enabling Technologies:

嵌入式硬件, 嵌入式操作系统和軟體開發的支持
(計算與環境融為一體！)

Wrist watches will monitor our blood pressure and heart beat

Latest top-level BMWs contain over 100 micro-processors
Enabling Technologies: Communication

- High-speed Internet
  - 10-100 Mbps edge connection
  - Bandwidth of single fibers ~10 Gb/s
  - E.g., 10 Gb/s HKU Campus Network

- Wireless
  - mobile phone: 3G(2.4Mbps), 4G(100 Mbps)
  - wireless LAN (> 10 Mb/s), BlueTooth

- New network technologies:
  - Power line networking (PLC)
    - HomePlug: 14Mbps data transfer rates
    - coffee maker “automatically” connected to the Internet
  - Personal area networks
  - Wireless radio frequency (RF) technology

- Everything can be connected !!

HK "Wi-Fi city“: > 3000 Hotspots
Enabling Technologies:
新視覺顯示技術

👉 E-paper: *(Write and Display Everywhere)*
- reusable display material
- high contrast, low energy, flexible
- Rewritable with magnetic pen

👉 Smart Glasses: *(View Everywhere!)*
- Visual information will be written directly onto our retinas (視網膜) by devices in our eyeglasses and contact lenses”

Foldable and Rollable Display

*E-paper from Xerox and 3M*
Enabling Technologies: Sensors/Actuators (傳感器 / 促動器)

- Basic sensors: 壓力、溫濕度、流量、液位、超聲波、浸水、照度、加速感應器 (accelerometer)
- Fingerprint sensor
- RFID
- Infrared
- Location sensors
PvC: Core Research Issues

普適計算的挑戰

Remote communication  
Fault tolerance  
High availability  
Remote information access  
Distributed security

Mobile networking  
Mobile information access  
Adaptive applications  
Energy-aware systems  
Location sensitivity

Distributed systems  
Mobile computing  
Pervasive computing

"Pervasive Computing: Vision and Challenges"
M. Satyanarayanan [CMU, Aura Project, 2001]
1. Effective Use of Smart Spaces

A smart space is an enclosed area equipped with embedded computers, information appliances, and multi-modal sensors allowing people to perform tasks efficiently by offering access to information and assistance from computers.
2. Invisibility:

Making computers ubiquitous (everywhere) is not enough; we should also strive to make them invisible.

(1) Make it small: fully embedded and physically disappear
(2) Make it smart: The environment continuously meets user expectations and rarely presents him with surprises
3. Localized scalability:

The Problem:

- As smart spaces grow in sophistication, the intensity of interactions between a user’s personal computing space and its surroundings increases.

Consider scalability with physical distance, despite their potential huge number of devices.
4. Masking Uneven Conditioning

Not all spaces are equally smart.

Large dynamic range of “smartness” causing user distraction (分心).

- E.g.: A teacher moves from a well-equipped classroom to a classroom with only a blackboard.

Masking Uneven Conditioning: make user able to act normally, even if the service/application is not available or is not fully functional.
Multi-Fidelity Computation - Re-think our model of computing

**Traditional algorithm**
- fixed correctness criteria (fixed output spec)
- variable amount of resources consumed to meet this

**Multi-fidelity** 多保真度 algorithm
- Multiple notions of “correct”; each is a level of fidelity
- Many allowable outputs/answers:
  - different fidelities ➔ different outputs, resource usage
  - “Do the best you can using no more than X units of resource”
Short Summary

Pervasive/Ubiquitous Computing

Mass deployment of computing in everyday life
- computing anytime, anywhere;

An environment saturated with computing and communication capability, yet so gracefully integrated with users that it becomes a “technology that disappears” -- Invisible Computing;

“Each person is continually interacting with hundreds of nearby interconnected computers without explicitly attending to them” -- Machines sense users’ presence and act accordingly
The Pervasive Expedition
普適計算時代的機遇和新挑戰

Remote communication
Fault tolerance
High availability
Remote information access
Distributed security

Mobile networking
Mobile information access
Adaptive applications
Energy-aware systems
Location sensitivity

Distributed systems  Mobile computing  Pervasive computing

Our Efforts

New PvC system: Context Awareness + Dynamic Adaptation

Smart spaces
Invisibility
Localized scalability
Uneven conditioning

"Pervasive Computing: Vision and Challenges"
M. Satyanarayanan [CMU, Aura Project, 2001]
Our Focus

Context awareness support:
- The software is able to monitor the context of its environment (smart space), itself (program), and its users (user status/intent)
- Deep Awareness

Dynamic adaptation support:
- Resource-aware adaptation:
  - (Bandwidth) Context-aware network sockets
  - (Battery power) Energy-aware power management
- Environment adaptation: (dynamic smart space construction)
  - spontaneously integration or removal of devices and application components.
- Content/data adaptation:
  - Change the data formats (lower resolution, smaller image,..)
- Functionality adaptation:
  - Same functionality but different code/implementation: e.g., trading time for space (different algorithms for tree search, sorting, …use different software for communications: e-mail, SMS, video chat, text chat..)
Deep Awareness

The majority of context-aware computing to date has been restricted to location-aware computing for mobile applications (location-based services).

**Deep Awareness:**
- Make full use of context information
- Make use of “commodity sensors” (e.g., WebCam, RFID, Temp/Light,..)
- Make use of “soft sensors” (weather forecast, work schedule, on-line maps,..)
Part II: Sparkle PvC Systems
Sparkle PvC Systems

- A component-based software architecture with *functionality adaptation* for Pervasive Computing
- Goal: achieve 4 “A”s -- Computing *Anytime, Anywhere, at Any device,* and support *Any Application.*
Functionality Adaptation in Sparkle

**Facet Programming Model:**
- **Facet** = code + facet description
- **Separation of code and data,** preparing for
  - **Adaptation:** code and data can be adapted individually
  - **Migration:** state is kept in container (root facet + UI)
- **Facet is stateless:**
  - makes it throwable & replaceable at run-time
- **Functionality Adaptation**
  - Components of the same functionality have varied granularity and/or feature
  - **Multi-fidelity computing:** Pick the one that meets user’s needed “functions” and resource availability.
Container Concept: “Migration-Ready”

- **Application-like abstraction**
  - Interacts with the user through the UI
  - Provides a place to store run-time state
  - Provides specifications of the root facets

- **Root facet specification**: the functionalities this particular container can offer.

**Adaptable GUI (UI ML)**

**Storage Area**

**Application Functionalities**
- Root Facet Spec. 1
- Root Facet Spec. 2
- Root Facet Spec. 3
- Root Facet Spec. 4
- Root Facet Spec. 5

User Interface Markup Language (UIML)
Functionality Adaptation

Facet Dependency Graph

- Facets may call upon other facets to achieve their functionality.
- May have more than one facet fulfilling the functionality (e.g., i, j, k for A).
- Dependency types:
  - "compulsory"
  - "optional": "if-then-else"

```
FuncID = A (sorting)

i: quick sort; i: bubble sort; k: merge sort
```

```
FuncID = B (FFT)

p q r
```

Inactive facets can be thrown away (GCed)

- Only active facets are kept in memory.

Active Facet - currently running

Inactive Facet - already executed completely

Facet which has not yet been brought in/loaded

Inactive facets can be thrown away (GCed)

- Only active facets are kept in memory.

Active Facet - currently running

Inactive Facet - already executed completely

Facet which has not yet been brought in/loaded

```
A

i j k

p q r

i j k

s t

A
```

```
C
```

```
X
```

```
i: quick sort; i: bubble sort; k: merge sort
```
Functionality Adaptation: How?

- The proxy compares the *resource requirement* of facets with the resource availability in the client.
- Proxy will send a facet whose resource requirement + the resource requirements of all its dependencies together is less than the resource availability in client.

```
...// Facet R’s Resource Requirement
<dynamic>
    <input_variables>
        <parameter name="m"> 1 </parameter>
        <parameter name="n"> 2 </parameter>
    </input_variables>
    <formula> n^2+5m </formula>
</dynamic>
```

Resource Consumption of Facets
Sparkle Runtime

Proxy

Proxy to Proxy Communication

Facet Request & Facet Return

Facet Transfer

Context retrieval & notification

Proxy

Client Device

Execution Delegation

Peer to Peer Communication

Execution Servers

Facet Servers

Context Servers

http/XML
Part III: Context-aware Mobile Instant Messenger
The pervading IM

“42% of internet users report using instant messaging”

“53 million adults in U.S. exchange instant messages and 24% of them swap IMs more frequently than email”

--- “Pew Internet and American Life” Survey 04

ICQ, MSN Messenger, Yahoo!, gtalk, AIM, QQ (78% in China)…

Increased Internet connection time booming population.
IM User in Feb. 2006
April 10, 2006 – reported by comScore Networks

Europe: 82 million people, (49% of the online population), used IM applications to communicate online.

North America: 69 million people (37% of the online population), used IM.

Latin America: 64% of its online population chatting to each other via IM.
Why people like IM?

- People can locate and be located by others in cyberspace while maintaining various degrees of control over their privacy and the timing in which they are willing to communicate.

  Presence-awareness (在場; 存在)
  - Indicates a user’s responsive status

  Sense of “Buddy” (親密夥伴,拍檔,)
  - Keep a friend-to-friend network
  - Stay social and connected

One-click distance

Free talk in any dialect
  - “gd nite & cu tmr”

  表情符號 (日語: 「顔文字」)
  - “麻吉language”

Emoticon 表情符號
### 表情符號 (Emoticon = Emotion icons)

<table>
<thead>
<tr>
<th>符號</th>
<th>意思</th>
<th>符號</th>
<th>意思</th>
</tr>
</thead>
<tbody>
<tr>
<td>o - -) 0。0)</td>
<td>給你一拳~~</td>
<td>m(_ _)m</td>
<td>萬事拜託(跪地，嘆頭)</td>
</tr>
<tr>
<td>(/__)</td>
<td>看不到~眼睛被手遮住了</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(⊙。⊙)</td>
<td>目瞪口呆</td>
<td>⊙。⊙</td>
<td>睜大眼</td>
</tr>
<tr>
<td>(^ 0^)/</td>
<td>舉手歡呼</td>
<td>－－</td>
<td></td>
</tr>
<tr>
<td>?~?</td>
<td>疑問、疑問</td>
<td>(T_T)</td>
<td></td>
</tr>
<tr>
<td>Q Q</td>
<td>流淚</td>
<td>Q０Q</td>
<td></td>
</tr>
<tr>
<td>〒△〒</td>
<td>哭</td>
<td>T△T</td>
<td></td>
</tr>
<tr>
<td>U／／／U</td>
<td>臉紅紅～</td>
<td>≥△≤</td>
<td></td>
</tr>
<tr>
<td>(－_－)。</td>
<td>兩手一攤～</td>
<td>= ＝ #</td>
<td></td>
</tr>
<tr>
<td>= 3 =</td>
<td>嘟嘴(</td>
<td>(－_－)y-----</td>
<td></td>
</tr>
<tr>
<td>(/＾ III －)/</td>
<td>抓你來咬!!!</td>
<td>(Q o Q) b</td>
<td></td>
</tr>
<tr>
<td>*(^_^)/ *</td>
<td>拿彩球、為你加油～</td>
<td>(# ^ ^)</td>
<td></td>
</tr>
<tr>
<td>(&gt;_&lt;)</td>
<td>小生氣</td>
<td>_/#/</td>
<td></td>
</tr>
<tr>
<td>(<em>^。^</em>)</td>
<td>親一個!!!</td>
<td>(＠^ 0^＠)/★</td>
<td>晚安～</td>
</tr>
<tr>
<td>(&gt;。&lt;)</td>
<td>不!!!!!!!!!!!!!!!11</td>
<td>(*+。<del>+)</del>@</td>
<td>受不了~受不了</td>
</tr>
<tr>
<td><del>(<em>∩_∩</em>)</del></td>
<td>獻上最可愛的笑容</td>
<td>(x_x)</td>
<td>昏倒</td>
</tr>
<tr>
<td>........(&lt;&gt;)/</td>
<td>哇!!出現了!!</td>
<td>= =b</td>
<td>冒冷汗</td>
</tr>
<tr>
<td>^ ^&quot;</td>
<td>笑笑的無奈</td>
<td>(－_－) z Z</td>
<td>睡著了啦～</td>
</tr>
<tr>
<td>－▽－ y</td>
<td>耶！</td>
<td>U__U</td>
<td>恩恩～</td>
</tr>
<tr>
<td>((。(^_^。))</td>
<td>期待，期待</td>
<td>(＜(￣▽￣)＞</td>
<td>哈哈哈～(我是壞人～)</td>
</tr>
<tr>
<td>&quot;(/&lt;&gt;)/</td>
<td>阿達～</td>
<td>(&gt;c&lt;)</td>
<td>唉唉叫～</td>
</tr>
</tbody>
</table>
Context-aware Instant Messenger

Exploit the usage of IM on mobile devices in future pervasive communication:

Pervasiveness: 4 “A”s:
- Anytime, Anywhere, on Any devices, for Any Applications

Deep (context) awareness:
- Know when, where, and how to communicate.

Buddy-like conversation
- among “Anything”.
- with familiar interface

<table>
<thead>
<tr>
<th>P</th>
<th>D</th>
<th>S</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

P – Person
D – Device
S – Software
O – Other entities
Pushing IM into Pervasive Computing Environments
Current Desktop IM

“Presence cues” could be

- 在線, 離線 (Log in or not), 忙碌 Busy, 馬上回來 Away, keyboard activity, 開心?..
- 狀態更新: Usually manual input status & custom text

What’s the problem?

- Limited: not rich presence
- Static: don’t change very often
- Fixed: once set, same display on all buddies’ IM.
Mobile Instant Messenger

When it comes to mobile situation:

- User’s status changes more frequently
  - location, activity, environment, etc
- Since user’s moving, they only want to pay minimum effort to update mobile presence.

Existing MIM

- merely a “slim” version (with similar functions) of their Desktop one
  - e.g., MSN mobile, Agile Messenger, QuickIM, IM+ All-in-One Mobile Messenger
Context-Aware MIM for PvC

- Everything as your buddy and can be communicated using real-time message exchange
- Three main features
  - Context-aware (情境感知的) presence management
    - Context (情境, 上下文) as presence
    - Different buddies see different status
  - Dynamic grouping (Buddies management at IM client)
    - Location-based Grouping (“buddy discovery“)
    - Activity-based Grouping (“task centric“)
  - Resource buddy services
    - extend the concept of “buddies” to all software and hardware components in your working space
    - IM as the unified communication interface
    - Buddy understands your dialect
Issues on Mobile Presence

How to interpret the raw location data (stream of GPS coordinates 座標) to symbolic, human-readable annotations -- “significant location”?

- Mobile phone has limited computing power and small memory
- Call for lightweight location extraction algorithm

With location info, how to enhance the richness of presence status?
Mobile Presence in our MIM

Presence is extended to

**Status: activity @location**

- **Location:**
  - **i-Cluster:** GPS-based, extraction of “significant locations”
  - “Cooperative Geo Tagging” (place annotation)
  - Support Google Maps (no pre-installed map needed)

- **Status:**
  - Availability and willingness for communication based on the current activity and buddy relationship.

- **Activity:**
  - Based on registered calendar information
  - Google Calendar
What's the location here?
Location Recommendations:
my buddies said:
- The Yummy Restaurant by 
- My favorite restaurant by 
- Pokfulam Road by 
Others:
HKU, Water Street, ...

Buddy's presence shown to user:
- Available: meeting@Starbucks
- Available: meeting@Starbucks
- Away: unknown@unknown

Presence is extended to status: activity@location

Buddy's presence shown to user:
- Busy: meeting@Starbucks
- Busy: meeting@Starbucks
- Busy: meeting@Starbucks

Google Calendar:
- Time: 2:00pm-4:00pm
- Location: Starbucks
- Participant: 
- Activity: Meeting

creates a group activity with in the Calendar.
(1) GPS-based Location Extraction

Raw GPD Data Points:
*Where have I stopped by?*

Error rate
Signal Lost

Conference venue

Traditional approach ➔
View location extraction problem as that for *identifying densely clustered regions* – high time and space complexity
**i-Cluster algorithm**

- **Time-based Clustering** [Kang:2004]
  - clusters the locations along the time axis (only recent coordinates within some time)
  - Sought only location visit that has a *recognizable duration*

- **i-Cluster algorithm**
  - We consider places *revisited shortly*
  - Entrance of a parking lot
  - Junctions of street
  - Shortly come back due to a disrupted task
(2) Cooperative Location Annotation: Geo Tagging

From the raw location data to symbolic, human-readable annotations by selecting a place label recommendation created by their buddies based on

- **Location proximity**: distance
- **Hit number**: how many times this label used by others
- **Subjective**: depend on user’s focus of interest (e.g., current activity) on the spot

1. Gis Convention Center (23)
2. GCC/NTU (14)
3. EUC venue (8)
4. No.85, Roosevelt Road, Sec. 4 (5)
5. MRT KungGua Station (3)
(3) Context-aware presence management

How your current status is displayed in your buddies’ IM?
- Different presence status is shown depending on your current location, activity & the social relationship between YOU and your buddy
- Produce customized presence to different buddy based on an Ontology-based reasoning engine

Presence is extended to:
Status: activity @location

In class, bored, sleepy, quiet...
At home, just quarreled, annoyed
Example: C.L. Wang’s Current Presence Display at his Buddy’s IM client

- **My boss**
  - Status: Busy
  - Activity: Presenting paper
  - Location: Hall A

- **C.L. Wang’s Social Network**
  - Status: 24x7 available
  - Activity: Conference
  - Location: Taichung/Taiwan

- **Conference Program Chair**
  - Status: Busy
  - Activity: Presenting paper
  - Location: Hall A

- **My student**
  - Status: Not Available
  - Activity: Conference
  - Location: Taiwan

- **My wife**
  - Status: Busy
  - Activity: Trip
  - Location: Taiwan
Presence Ontology

Language used: Web Ontology Language (OWL)
# Reasoning rules for customized presence

**Using Jena Semantic Web Framework**

**Reasoning Rules (examples)**

<table>
<thead>
<tr>
<th>Cases</th>
<th>Antecedents</th>
<th>Consequents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the presence shown to a colleague</td>
<td>hasTime(CurrentTime, “Work time”)</td>
<td>hasAvailableStatus(?x1, ?x2) hasSameGroup(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td>hasWorkRelation(?x1, ?x2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasSameActivity(?x1, ?x2)</td>
<td></td>
</tr>
<tr>
<td>Determine the presence shown to family member</td>
<td>hasTime(CurrentTime, “Work time”)</td>
<td>hasBusyStatus(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td>hasFamilyRelation(?x1, ?x2)</td>
<td>hasActivityHidden(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasLocationShown(?x1,?x2)</td>
</tr>
<tr>
<td>Determine the presence shown to friend</td>
<td>hasTime(CurrentTime, “Work time”)</td>
<td>hasAwayStatus(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td>hasFriendRelation(?x1, ?x2)</td>
<td>hasActivityHidden(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasLocationHidden(?x1,?x2)</td>
</tr>
<tr>
<td>Determine the presence shown to colleague</td>
<td>hasTime(CurrentTime, “Off-duty”)</td>
<td>hasAwayStatus(?x1,?x2)</td>
</tr>
<tr>
<td>when it’s off-duty</td>
<td>hasWorkRelation(?x1, ?x2)</td>
<td>hasActivityHidden(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasLocationHidden(?x1,?x2)</td>
</tr>
<tr>
<td>Determine the presence shown to friends</td>
<td>hasTime(CurrentTime, “Off-duty”)</td>
<td>hasAvailableStatus(?x1,?x2)</td>
</tr>
<tr>
<td>when it’s off-duty</td>
<td>hasFriendRelation(?x1, ?x2)</td>
<td>hasActivityShown(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasLocationShown(?x1,?x2)</td>
</tr>
<tr>
<td>Determine the presence shown to family member when it’s off-duty</td>
<td>hasTime(CurrentTime, “Off-duty”)</td>
<td>hasAvailableStatus(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td>hasFamilyRelation(?x1, ?x2)</td>
<td>hasActivityShown(?x1,?x2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasLocationShown(?x1,?x2)</td>
</tr>
</tbody>
</table>
MIM System Implementation

- **Context Listener**
  - Jena
  - Server
- **Location DB Operator**
- **Database**
- **Google Calendar Updater**
- **Packet Listener**
  - Jabber
  - Server
- **Google Map Generator**

**Server Side**

**Dynamic Buddylist**

**XMPP Packets**

**GPS Info**

**WiFi or GPRS**

**MIM Client composes**
- Location, Schedule, Map packets

**IM Functions**
- IM Functions
- i-Cluster component
- Map Display

**Internet**

**MIM Client composes**
- Location, Schedule, Map packets

**WiFi or GPRS**

**IM Functions**
- IM Functions
- i-Cluster Component
- Map Display
Communication Sequence Diagram

- **Mobile Client**: register()
- **Jabber Middleware**: location clustering
- **Location Packet**: Schedule Packet
- **Dynamic Buddylist**: pushBuddylist()
- **Map Packet**: fileTransfer (image)
- **Jena Server**: updateLocation() returnOptions()
- **LocationDB Operator**: subscribe() &update()
- **Calendar updater**: subscribe() &update()
- **Map Generator**: requestMap(location) Google Map Image
Implementation Details

**MIM client**

**Hardware:**
- C720W Smartphone, (New: Nokia N73)
- GPS receiver: Holux GPSlim236 with Bluetooth connection

**Software:**
- J2ME, Windows Mobile 5.0 Operating System, based on moJab

**Code size:**
- ~1.1 MB without GPS data (source code 196KB).
Evaluation

Parameter setting

- $d = 40$ meters
- $t = 300$ seconds
- $t_{intv} = 1200$ seconds
- $l = 60$. (1 min history samples)

The values of $d$ and $t$ are determined according to the knee point in [3].

9373 GPS data points in 2.6 hours
(a) Seven extracted places:
   a: the King George V Memorial Park
   b: a 7-Eleven convenience store
   c: a Pizza-Box store
   d: a Bus station
   e: the Flora Ho Sports Centre
   f: the Pokfulam Road Playground
   g: a restaurant

(b) The zoom-in view of location points nearby place b

(c) The centroid of clusters in Tempplaces where Cluster 4 and 10 are merged
Snapshots

Snapshots generated by emulator

(a) MIM Login GUI  (b) Buddylist of Jo  (c) Map with buddy locations

MIM on Nokia N73
Demo

MIM Demo

http://sparkle.cs.hku.hk/wiki/index.php/Project
Conclusion and Outlook

**MIM**
- GPS-based location extraction
- Cooperative place annotation
- Context-awareness presence management

**Future work**
- More efficient and accurate clustering algorithm
- Integration with indoor location service
- Incorporate more presence cues (IM contents)
- Use more public Web services
Thank You!
Q&A

More in

@Taiwan