The Design, Implementation and Evaluation of the CenceMe Application

Artemis Eracleous
Paraskevi Petridou
Outline

• Introduction
• Motivation
• Main contributions
• Design and implementation
• Evaluation and Results
• Related Work
• Future Work
Introduction – CenceMe

• Mobile application which infers people’s sensing presence with the use of sensor enabled mobile phones.
• Software developed on: Nokia N95.
• Sharing of information through social networks (e.g. Facebook, MySpace).
Motivation and Problem Definition

• The people’s need for communication with other people.
• Importance of sensing information (“where are you?”, “what u doing?”)
• Ability of mobile phones to provide sensor networks
  new application domain where people: carriers of sensing devices, sources and consumers of sensed events.
Main contributions

• Design, implementation and evaluation of fully functional mobile sensor application
• Classifiers’ design on mobile phones
• RAM, CPU and energy measurements
• Evaluate the CenceMe application through a user study
Mobile Phones’ Limitations (1)

• OS, API and Operational:
  • N95 uses Symbian operating system and JME
  • use small amounts of memory and computational resources
  • limited programmability
  • not able to fully deploy an application
  • manufacturers maintain devices and operational networks’ closed nature.

• Security:
  • signed keys – protection from attacks
Mobile Phones’ Limitations (2)

• Energy management:
  • waste of battery power: GPS, Bluetooth, GPRS radios and data upload
  • Symbian and JME do not provide duty cycle
design sensing duty cycle with less frequently sensor sampling.
Design Issues (1)

• Split-Level Classification:
  • sensing presence is derived from classifiers
  • classify sensor data by pushing some classification to the phone and some to the servers
  • phone output: primitives stored in a database to be retrieved for more complex classification
  • backend output: facts stored in a database to be retrieved and published
  • advantages: support customized tags, provides resiliency to cellular/WiFi dropouts, minimizes the sensor data sent, reduces the energy consumed, increase the user’s privacy and integrity
Design Issues (2)

• Duty-Cycle:

  • How long can the sleep interval be? The larger the sleep interval the lower the classification responsiveness
  • Idea: minimizes sampling but maintains the application’s responsiveness thus operating as near to real time as possible lower duty cycle but increasing sensing rate when a buddy’s page is accessed. This leads to bandwidth and storage improvements.

• Portability:

  • the majority of mobile phones use java virtual machine to support JME push more to JME
Design and Implementation of CenceMe

• Software running on Nokia N95 and backend infrastructure

• Operations: sensing, raw sensed data classification, people’s presence presentation, primitives’ upload

• Primitives result: sound samples, accelerometer data, scanned Bluetooth MAC addresses, GPS readings, random photos
Software Architecture (1)

Figure 1: Architecture of the CenceMe phone software.
Software Architecture (2)

- **ClickStatus**: visualizes sensing presence
- **WatchTasks**: restarts the processes that fail
- **Each component is a single thread**: if one fails does not affect the rest
Figure 3: Software architecture of the CenceMe backend.
Backend Software (2)

• **Phone ↔ Backend Communication**: data exchange whenever the phone has primitives to upload

• **Presence representation and publishing**: set of icons to represent presence with “push” or “pull” approach
CenceMe Classifiers – Phone Classifiers

• **Audio**: outputs the audio primitive which indicates if the sample is human voice. Two steps:
  • feature extraction: discrete Fourier transform (human voice: 250Hz – 600Hz)
  • classification: machine learning algorithm – training set consists of audio sample of human voice and environmental conditions

• **Activity**: takes accelerometer data and outputs the current activity. Two components:
  • preprocessor: takes the data and extracts features. Calculates the mean and deviation
  • classifier itself
Activity Classifier Results

Figure 6: Accelerometer data collected by the N95 on board accelerometer when the person carrying the phone performs different activities: sitting, standing, walking, and running.

- x, y, z axes: accelerometer readings
CenceMe Classifiers – Backend Classifiers (1)

• Two ways:
  • event triggered: input the primitives and outputs the current activity
  • periodic: run periodically based on the availability of data in a window of time

• **Conversation**: indicates whether a person is in a conversation. Input: audio primitives but are not accurate rolling window of N phone audio primitives (N = 5).
  • If 2 out of 5 indicate voice “conversation” state
  • If 4 out of 5 indicate no voice “no conversation” state
CenceMe Classifiers – Backend Classifiers (2)

• **Social Context**: uses primitives and facts to output the social context:
  
  • neighborhood conditions: indicates whether there are any CenceMe buddies close to the user (Bluetooth MAC Address)
  
  • social status: uses the activity and conversation classifiers’ output to show if a person is with other CenceMe buddies talking, alone, or at a party. Also defines if partying or dancing with an approach that combines sound volume and activity.
CenceMe Classifiers – Backend Classifiers (3)

- **Mobility Mode Detector**: takes as input the GPS estimates. It distinguishes if travelling in a vehicle or not (using speed measurements). Uses JRIP algorithm.

- **Location**: location estimates used by other classifiers, using GPS samples

- **Am I Hot**: uses metrics (nerdy, party animal, healthy, greeny). Users compare themselves with others
Evaluation and Results

- 8 users
- 1 week period at intervals of about 15 to 30 minutes
- Data collected at different locations carrying the mobile phone in different positions on the body and environmental conditions
General Results

• Activity Classifier:

<table>
<thead>
<tr>
<th></th>
<th>Sitting</th>
<th>Standing</th>
<th>Walking</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sitting</td>
<td>0.6818</td>
<td>0.2818</td>
<td>0.0364</td>
<td>0.0000</td>
</tr>
<tr>
<td>Standing</td>
<td>0.2096</td>
<td>0.7844</td>
<td>0.0060</td>
<td>0.0000</td>
</tr>
<tr>
<td>Walking</td>
<td>0.0025</td>
<td>0.0455</td>
<td>0.9444</td>
<td>0.0076</td>
</tr>
<tr>
<td>Running</td>
<td>0.0084</td>
<td>0.0700</td>
<td>0.1765</td>
<td>0.7451</td>
</tr>
</tbody>
</table>

• uses accelerometer on N95
• difficulty distinguishing sitting from standing given the similarity in the raw accelerometer traces
General Results (2)

• **Conversation Classifier**

<table>
<thead>
<tr>
<th></th>
<th>Conversation</th>
<th>Non-Conversation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversation</td>
<td>0.8382</td>
<td>0.1618</td>
</tr>
<tr>
<td>Non-Conversation</td>
<td>0.3678</td>
<td>0.6322</td>
</tr>
</tbody>
</table>

• High accuracy but high rate of false positives due to a combination of classifier design and mis-annotation by participants
• Reports conversation even if the person is not talking – someone is talking nearby
• The classifier remains in the conversation state for longer time than real conversation (quickly enters a “conversation state”, moves slower out of that state)
Impact of Phone Placement on the Body

- x axis: body position, y axis: accuracy
- Body placement affects the activity inference accuracy
- Different places: pocket, lanyard (length and type affect results), clipped to a belt
- Pocket and belt produce similar results
- Lanyard produces poor accuracy while sitting and relatively accuracy while running
- Lanyard: 88% and 72% accuracy for conversation and no conversation while in pocket: 82% for conversation and 71% for no conversation accuracy is less sensitive

Figure 7: Activity classification vs. body position.
Impact of Environment

- x axis: location, y axis: accuracy
- Activity classification accuracy is independent of the environment
- Conversation classification accuracy is affected by the environment conditions
- Indoor noisy: 85% success
- Outdoor: increase in false positives but high accuracy
- Indoor quiet: lower accuracy because of the background noise

(a) Conversation classifier in different locations.
Impact of Duty Cycle (1)

- x axis: duty cycle, y axis: accuracy
- Little advantage having a sleeping time < 10s but longer duty cycle
- 40% accuracy with the conversation classification for 60s duty cycle
- **Longer duty cycle** → reduction of the conversation classifier rolling window size which means higher misclassification rate
Impact of Duty Cycle (2)

- x axis: false positive, y axis: true positive
- The larger the window the larger the true positives to false positives ratio
- N=5: delay is 1.5 minutes (operating point)
- If N is increased the accuracy is increased but more delay

(c) ROC curves for the conversation classifier.
Power Benchmarks (1)

Figure 9: Details of the power consumption during a sampling/upload interval.

- x axis: time, y axis: power
- Highest spikes: upload of data
- Next highest spikes: audio sampling
- Accelerometer sampling and activity classification: fast and little use power
Power Benchmarks (2)

Figure 10: The tradeoff between energy consumption and data latency in CenceMe.

- x axis: seconds of sensing interval, y axis: average power, second vertical axis: delay
- Combination of two lines: tradeoff between energy and data latency
- There is no optimal sampling interval because users requirements vary at different times
- Audio sampling and processing use a relatively high amount of energy
Memory and CPU Benchmarks

Table 4: RAM and CPU usage

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>RAM (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone idle</td>
<td>2% (+/- 0.5%)</td>
<td>34.08</td>
</tr>
<tr>
<td>Accel. and activity classif.</td>
<td>33% (+/- 3%)</td>
<td>34.18</td>
</tr>
<tr>
<td>Audio sampling and classif.</td>
<td>60% (+/- 5%)</td>
<td>34.59</td>
</tr>
<tr>
<td>Activity, audio, Bluetooth</td>
<td>60% (+/- 5%)</td>
<td>36.10</td>
</tr>
<tr>
<td>CenceMe</td>
<td>60% (+/- 5%)</td>
<td>36.90</td>
</tr>
<tr>
<td>CenceMe and ClickStatus</td>
<td>60% (+/- 5%)</td>
<td>39.56</td>
</tr>
</tbody>
</table>

- Audio sampling and feature vector extraction require more computation
User Study

• Tested with 22 people
• Nokia N95 with CenceMe
• 3 weeks
• Overall: location information is used the most, positive experience, some of the users liked the random pictures’ feature and some disabled it
Related Work

• iCAMS, Twitter and WatchMe: CenceMe provides rich context about a person
• Inter Mobile Sensor Platform: CenceMe operates on less capable devices while remaining effective
• Previous works on activity inference and modeling using sensors: CenceMe implements the activity inference algorithms on commercial mobile phones
Future Work

• Less energy consumption: achieve 48 hour duration of battery life

• Improve privacy policy and ClickStatus interface, thus more powerful ways to browse their friends

• Minimize time for primitives and facts. It is required by the users to have real time access
Conclusion

• Sensing information is important to people to be able to communicate with other people, using sensor networks through mobile phones

• Implementation of an always-on application has many limitations and trade-offs

• **Contribution**: design a fully functional mobile sensor application

• **Strengths**: area with a growing interest

• **Weakness**: limited number of people who tested the application, not all results are accurate
Thank you