Device Ecology: A Micro Digital Ecosystem[1]

Maria Indrawan[1], Sea Ling[1] and Seng Loke[2],
[1]Faculty of Information Technology, Monash University, Australia.
emails: (maria.indrawan, sea.ling) @infotech.monash.edu.au
[2] Department of Computer Science and Computer Engineering, Latrobe University, Australia
e-mail: s.loke@latrobe.edu.au

***Abstract*—A digital ecosystem usually refers to a collection of small and medium enterprise businesses that interacts closely as a system. In this paper, we present a different type of digital ecosystems. We introduce the idea of creating an ecosystem from a number of smart devices. This ecosystem is categorised as a micro ecosystem rather than macro ecosystem. The proposed model of this digital ecosystem is called Device Ecology.**

***Index Terms*—digital ecosystems, smart devices, workflow, web services.**

I. Introduction

Recent development of digital ecosystems technology has focused its discussion on the idea of creating business ecosystems that encompasses small and medium enterprises as entities in the systems[6]. The biology metaphor is used as a model of the interactions among these SMEs and their environment (infrastructure). In this typical realization of the metaphor, a large or macro ecosystem is assumed. In biological ecosystems, it is possible to have a macro ecosystem that is made of a number of micro ecosystems. In this paper, we present a different kind of digital ecosystems. Unlike the popular approach of viewing a digital system as a macro ecosystem, we propose a view of a micro ecosystem as a biological metaphor in managing a collection of smart devices. The smart devices in the proposed systems are managed in a scenario of a smart home.

In this paper, we present our perspective on how the management of smart devices can be achieved through the use of the ecosystems concepts. We start by presenting a general philosophy and architecture that we adopt in our system in section 2. In order for our system to work properly, process management is needed. This issue is discussed in detailed in section 3. In particular, we illustrate how the high level language called Eco can be used to represent user defined workflow. Human plays an important role as an organism in our model. They are the users of the systems and need to interact effectively with the devices. In section 4 we illustrate how this language can be mapped to the BPEL4WS[5] as a low level language to control the operation of the devices. We have developed a prototype for the proposed model as depicted in section 5. Finally we present the conclusion and future work in section 6.

II. Device Ecology Architecture

In a smart home system as an ecosystem, we envisage that all the appliances can perform 'smart' behaviour. The appliances is defined as an electrical device or instrument designed to perform a specific function for household use. Bergman [2] and Norman[7] suggested in the future, more appliances will be built with computational and communication infrastructure support that allow these devices to be unobtrusive and seamless in performing a specific task. These devices are often called information appliances or internet appliances if they have Internet-connectivity. Considering that these appliances are able to perform a smart behaviour, we consider these appliances as organisms that populate our ecosystems. Another type of organism that is part of the proposed digital ecosystem is the human user of these appliances. The whole interactions of the information appliances, human and its environment, such as temperature, humidity, and time are the make up of our digital ecosystems. We called this ecosystem Device Ecology.

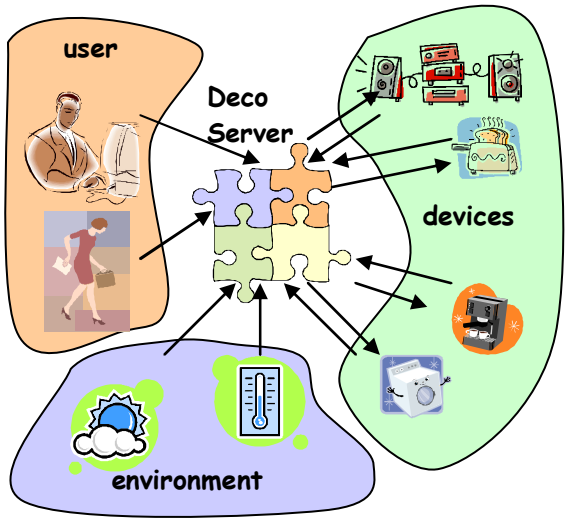

Figure 1. Device Ecology

There are three main types of "organisms" in the Device Ecology; the user, Deco Server and the appliances/devices. Figure 1 depicts the architecture of the Device Ecology. Similar to biological ecosystem, device ecology interact with its environment. Environment factors can influence the operation of the device ecology, for example, it is possible that the air conditioner will be turned on when temperature inside the house reaches certain degree. The information on the environment can be gathered through some sensors mechanism. Together with the user determined profile, the deco server makes appropriate decisions and activate appropriate devices.

Figure 1 shows that the central part of the Device Ecol-

[1] The authors acknowledge the support of the Australian Research Council for this project.

ogy systems is the Deco server. The Deco server controlled the interactions among devices. These interactions in the ecosystems is harnessed and controlled through Web services. We model the observable and controllable aspects of devices as Web services as in Matsuura et.al [4]. Such device modeling is not consistent with the emerging standard model for appliances such as the AHAM Appliances Model [1], where each appliances is modeled as a collection of objects categorized according to subsystems. We also note that there will be aspects of the device which are not exposed as Web service. The observable and controllable aspects of devices are harnessed by applying a workflow management model. In our system, we adopt the BPEL4WS [5] as the workflow management model for the Web services. Figure 2 depicts the conceptual architecture of the workflow engine in Device Ecology.

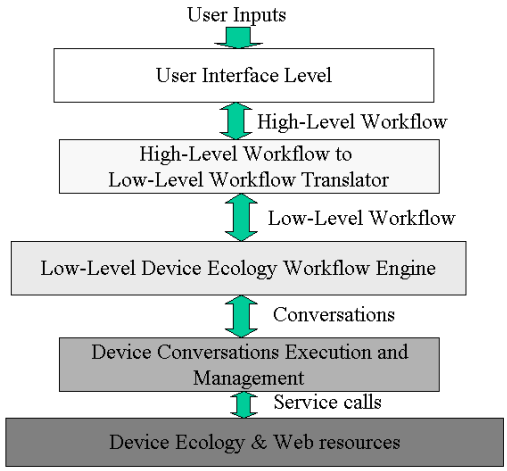

Figure 2. Multilayerd Conceptual Architecture of the Deco Server Workflow Engine

It is possible to expect that the designer and the developer of the Device Ecology to understand BPEL4WS. However, the user of the system needs to be alleviated from the requirement to understand this language in order to program their workflow. To support the user interaction, we develop an interaction language called eco. Description of this language is presented in the next section.

III. USER INTERACTION LANGUAGE

1) *Eco Language*

Eco is an English like interaction language to define a workflow in Device Ecology. It is a collection of abbreviated commands would make end-user programming simpler. Such commands can then map down to lower level language like BPEL4WS, which in turn, is used to control the devices. To illustrate the usage of the Eco as a command language, we consider a small workflow example depicted in figure 3.

We consider a small example of such a two-level model. The top level is a small user command language, which we call *eco*, comprising commands that can be combined for sequential or parallel execution. For simplicity, we consider a device ecology with only a few devices. Inspired by Omojokun and Dewan [8], we consider two kinds of commands, those which affect a single device and those which affect multiple devices. Commands affecting multiple devices are directed to a pseudo device that issues single-device commands to the appropriate devices. In fact, the Device Ecology Workflow Engine performs this function. It need only be provided with a predefined command set, expressed in *eco*, describing how to break down multi-device commands. This procedure can be nested to any depth, where each level of nesting corresponds either to a recursive call to a single instance of the Engine, or to the invocation of a separate instance.

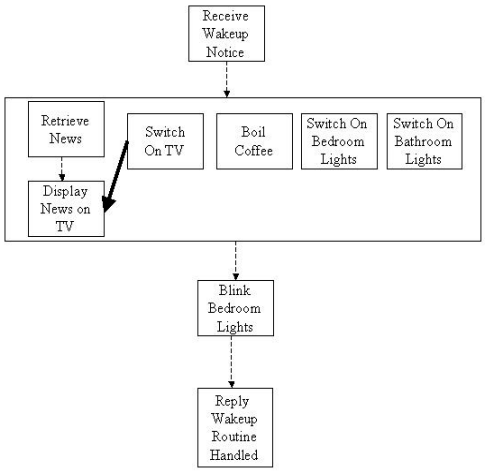

Figure 3: An Example Device Ecology Workflow

An example of an eco defined workflow for the example in figure 1 is as followed:

```
retrieve News; wait for News; activate coffee machine, switch on bedroom light; switch on bathroom light; switch on TV; wait for TV; display News on TV.
```

A sentence in eco represent a single workflow. This workflow may consist of a number of activation and control of devices. Each of these activities is represented as a CLAUSE in eco, for example, `Retrieve News;` clause. Sequential operation of a device or multiple devices is achieved by using the WAIT_CLAUSE, for example, `switch on TV; wait for TV; display News on TV.` The concurrent operations of devices is represented by concatenated the clauses using the symbol ";", for example, `switch on bedroom light; switch on bathroom light; switch on TV;` .Formally, the eco command language can be represented in EBNF format as shown in figure 4.

Note that some of the commands, although different in the command language, might invoke the same Web services with different parameters. The commands in the task language are a combination of a verb and a noun, some commands with parameters such as 'news'. In practice, the vocabulary of verbs and nouns can be based on a task on-

tology such as CLEPE [3]. Alternatively, the nouns and verbs can be extracted dynamically by a service discovery process. This is the approach we have taken here, which involved dynamically creating the parser's lexical analyzer from terminal symbols discovered at runtime The reason for the additional level of commands above the BPEL4WS level is abstraction, so that the user does not think of device ecology workflows in terms of Web services but rather in terms of what he/she observes or would expect of the devices in everyday terms. For example, a user who does not know anything about Web services can still command the device ecology based on the high-level commands. Moreover, many of these high-level commands are applicable in different settings. For example, any room that has lights can be commanded with "turn on all lights" but such a command will translate into a different low-level workflow, depending on what lights are available in the room itself. Such translations from high-level commands to low-level workflows can be pre-defined (e.g., by an administrator who is either a vendor or a savvy user) for each room using methods such as that presented above. Hence, the high-level command to turn on all lights has a different meaning or interpretation which is predicated on the actual room (i.e., the actual device ecology) the command is issued against. We term such commands *polydeco commands*, referring to commands whose meaning is device ecology dependent. There would also be commands that are applicable for different devices, and depending on the device, such commands will take on different meanings (and interpretation). For example, the command "switch on" can be applied to a light or to the television and the command "open" can be applied to drapes or to doors. We term such commands *polydevice commands*. Similar to polydeco commands, the actual meaning of the polydevice commands can be predefined, i.e., mappings from each command on a device can be mapped to a conversation with the device.

Ideally, a user, through experience of and general knowledge about the world, knows intuitively how to command devices and device ecologies, and so, does not need to learn about Web services or learn a new command set for every room visited or for every device encountered. There will be devices that an individual would not know about (e.g., new innovations) or would not know the full features of - the user will then need to learn new commands, perhaps adding to those already available by general knowledge.

```
SENTENCE::= clauses:CLAUSES  "."
END_INPUT;

CLAUSES::={clause:CLAUSE ";" ...}+;

CLAUSE::= activity_name_clause:
  ACTIVITY_NAME_CLAUSE |
  activity_clause:ACTIVITY_CLAUSE |
  wait_clause:WAIT_CLAUSE;

ACTIVITY_NAME_CLAUSE::="call"["it"]
  activity_name:ACTIVITY_NAME;

ACTIVITY_NAME::=name_word:NAME_WORD;

NAME_WORD::=word_nw:WORD;

ACTIVITY_CLAUSE::=[prepositions_v:
  PREPOSITIONS_V]verb_word:VERB_WORD
  [modifier_words:MODIFIER_WORDS ]
  [ prepositions_n:PREPOSITIONS_N]
  noun_word:NOUN_WORD ;

PREPOSITIONS_V::=PREPOSITIONS;

PREPOSITIONS_N::=PREPOSITIONS;

VERB_WORD::=word_v:WORD;

NOUN_WORD::=noun_n:NOUN;

MODIFIER_WORDS::={prepositions_modifier:
PREPOSITIONS_MODIFIER ... }+ ;

PREPOSITIONS_MODIFIER::=
  prepositions_m:PREPOSITIONS]
  modifier:MODIFIER ;

PREPOSITIONS::={preposition_word:
  PREPOSITION_WORD ... }+;

PREPOSITION_WORD::=preposition:
  PREPOSITION;

MODIFIER::=word_m:WORD ;

WAIT_CLAUSE::="wait" "for"
  wait_for_noun:WAIT_FOR_NOUN ;

WAIT_FOR_NOUN ::= noun:NOUN ;
```

Figure 4. ECO language in BNF format

2) Workflow models in Eco

A set of tasks can be configured into a workflow as sequential and/or concurrent tasks. In Eco, concurrent tasks are represented as a series of clauses separated by the sysmbol ";". For example,

```
Switch on bedroom light; switch on bath-
room light; turn on radio;.
```

Sequential tasks can be represented using the "WAIT_CLAUSE". For example,

```
Switch on bathroom light; wait for bath-
room light; turn on the bathroom tap;.
```

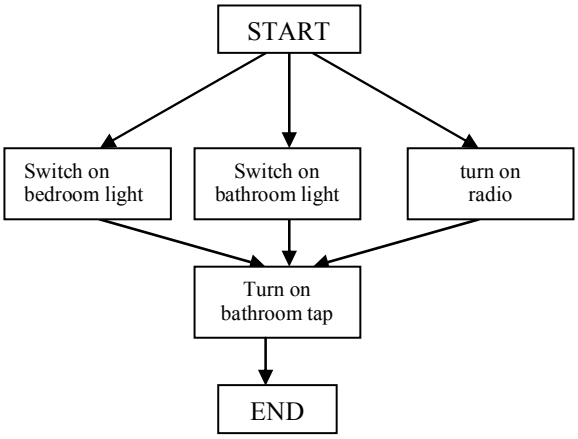

Figure 5. Grouping Devices with Wait Clause

The "WAIT_CLAUSE" can also use to synchronise operations of a multiple devices. For example the workflow diagram depicted in figure 5 can be expressed in the Eco as:

```
Switch on bedroom light; switch on bath-
room light; turn on radio; wait for bed-
room light, wait for bathroom light,
wait for radio, turn on the bathroom
tap;.
```

In the execution of the workflow, the WAIT_CLAUSE will cause the subsequent device to be push into a stack and will be activated accordingly when the device that cause the wait return a ready status to the workflow engine. In some workflow scenarios, the WAIT_CLAUSE operation can cause reduction in efficiency. Figure 6 depicts a scenario where the use of WAIT_CLAUSE can cause a decrease in efficiency.

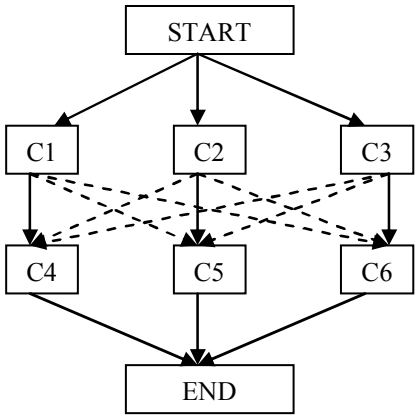

Figure 6. The efficiency issue in the ECO

Figure 6 shows six tasks in the workflow. The intended workflow is to have three concurrent paths, START-C1-C4-END, START-C2-C5-END and START-C3-C6-END. In Eco, the workflow will be written as:

```
Activate C1; activate C2; activate C3;
wait C1; wait C2; wait C3; activate C4;
activate C5; activate C6;.
```

The clauses wait C1, wait C2 and wait C3 before the activation of C4, C5 and C6 implies that C5 cannot start until C1 and C3 complete. The dotted lines in the diagram depict the implied WAIT_CLAUSEs that are assumed by the workflow engine during the execution of the example Eco statement. Considering this matter, nevertheless, under a normal circumstance the workflow will be executed according to the user request, although, it may not be in the most efficient manner due to the Eco design of the WAIT_CLAUSE. The decreased in efficiency become more noticeable in the case of device failure. Assume that a device that needs to execute task C1 is failed. In this situation, the whole workflow will come to halt due to the implied WAIT_CLAUSE for C5 and C6. One possible solution is to detect the existence of the implied WAIT_CLAUSEs in the workflow engine and remove it from its execution. The Eco language needs to be kept as simple as possible so that the novice user can use it to program the device ecology. The complexity need to be pushed to the lower levels, for example at the low-level workflow language or at the device conversation level. In the next section, we present the low-level workflow language in the device ecology and a way to translate Eco into this low level format.

IV. LOW LEVEL WORKFLOW DEFINITION

Each command in the top level language is translated into BPEL4WS format. We assume that there is a device ecology workflow engine for executing BPEL4WS specifications in the Deco server.

A command expressed in eco might be translated into a set of alternative workflows in the lower level language, where if one alternative fails during execution, another can be tried. The following general guidelines are used in translating eco into BPEL4WS:

1) A single eco sentence is represented as a single <process> element.
2) The devices activated in the workflow are defined as <partnerLink> elements. The <partnerLink> has three attributes; *name, partnerLinkType* and *role*. The attribute *name* represents the device name. The *partnerLinkType* and the *role* are used to relate the BPEL4WS and the WSDL during the execution. In the case of multiple devices involved in a single workflow, the element <partnerLink> may be encapsulated within <partnerLinks> element.
3) Element <variable> is used to represent the modifier in an eco sentence.

4) The start and end of a device activity are represented by the elements <receive> and <reply>. The element <receive> denotes the start of an activity, the element <reply> to denotes the end of an activity.
5) Encapsulated within the <receive> and <reply> will be either the elements of <flow> or <link>, depending on whether the operations are sequentially or concurrently executed. For a sequential execution, the <link> element is used and the <flow> is used for a concurrent execution of devices.
6) The combination of the <source> and <target> elements are used to represent the eco WAIT_CLAUSE.

The following eco command

```
Retrieve News; wait for News; activate
coffee machine.
```

is translated to the following BPEL4WS

```
<process name="morningActivity">
<partnerLinks>
  <partnerLink name="newsChannel"
    partnerLink Type="newsChannel:newsChannelLT"
    role="newsChannel"/>
  <partnerLink name="coffeeMachine"
    partnerLink
Type="coffeeMachine:coffeeMachineLT"
    role="coffeeMachine"/>
</partnerLinks>
<sequence>
  <receive partnerLink="newsChannel"
           portType="newsChannelPT"
           operation="retrieve"
           variable="newsIn"/>
  <flow>
    <links>
      <link name="L1"/>
    </links>
    <invoke partnerLink="newsChannel"
          portType="newsChannelPT"
          operation="retrieve">
      <source linkName="L1"/>
    </invoke>
    <sequence>
        <invoke partnerLink="coffeeMachine"
            portType="coffeeMachinePT"
            operation="activate">
            <target linkName="L1"/>
        </invoke>
   </sequence>
   </flow>
   <reply partnerLink="newsChannel"
          portType=:newsChannelPT"
          operation="retrieve"
          variable="newsOut"/>
</sequence>
</process>
```

Note that some tags may seem redundant but they are the result of a general template. During the actual implementation of a Device Ecology, it is possible to help the user to start the system with templates of common workflows. As a result, the user only needs to modify the preferences. This option may be considered very limiting for some users who want to design their own workflows. There are a few issues that Device Ecology has to support in order to give this flexibility. One of the most important issues is the validation of the workflow before it is deployed in the real system. The validation can be done in terms of two aspects. The first aspect is to validate the 'correctness' of the workflow from modelling point of view. In this case, we can use Petri-Net to validate the workflow. The second aspects related to the semantic operation of the workflow, for example, the deadlock detection. We have included in our Device Ecology, a mechanism to detect the existence of a deadlock in a given workflow defined in BPEL4WS language. The aforementioned validations are conducted before deploying the workflow in the system. It is still necessary to perform some runtime validation in the status of the workflow during execution.

V. PROTOTYPE

The conceptual architecture for the engine that executes and manages device ecology workflows is shown in Figure 4. Roughly, each layer of the architecture shows the components required to manage a level of abstraction. Note that user operations on an executing workflow such as cancel, pause, etc, can be issued during workflow execution - not shown in the diagram. The system keeps track of the correspondences between device conversations and the associated tasks in the low-level workflow, and between tasks in the high-level workflow and the associated low-level workflows, in order that exceptions are correctly reflected up (if required) through the abstraction levels. If there is an exception in a call to device, it can be traced to its corresponding low-level workflow task and then to its corresponding high-level workflow task. For example, if there is an exception in the call to switch on a light, it can be traced to the high-level task of turning on all lights, and the system might report to the user that there is an error in carrying out that high-level task but allow the user to drill down to specific faults in the lower levels. We are currently developing a prototype of our system with a subset of BPEL4WS as the low-level workflow language.

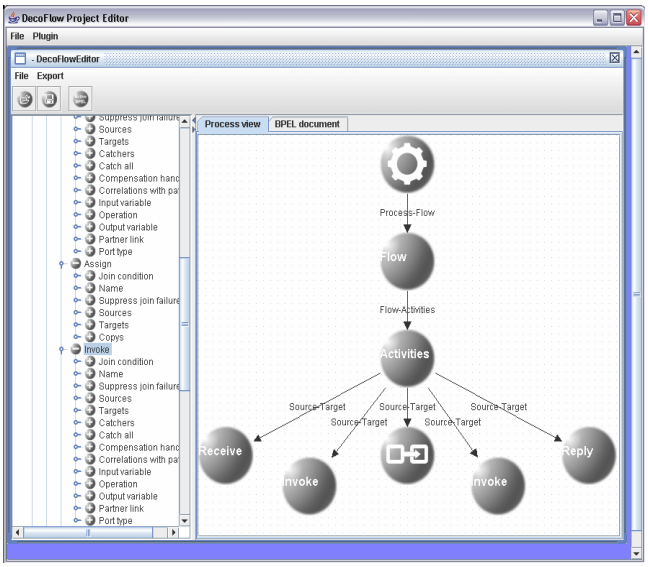

Figure 7. Sample screen from the prototype

Figure 7 shows the DecoFlow application with a sample BPEL4WS document loaded for visualisation. On the left side of the editor window is a tree displaying all of the information obtained from the BPEL4WS document. On the right hand side is the visualisation of the BPEL4WS document which is represented by a series of connected nodes which represent the process and activities from the BPEL4WS document.

VI. Conclusion

We have presented Device Ecology as an example of a micro digital ecosystem. The ecosystem is made up of human, devices and the environment. The ecosystem is modelled in three levels of abstractions: high level workflow, low level and device conversation. In this paper, we put forward the definitions and examples of the high level and low level workflow and how the translation can be done from the high to the low level. The translation is one of the many issues that need to be addressed in building the device ecology. Ongoing and future work is still actively investigated. Some of the issues that currently are investigated areas:

- Handling fault.
  Device ecology relies on web services to manage the processes in the workflow. Since web services is stateless, a way of monitoring the device status and its impact on the workflow execution is important. It is also possible that a device become inoperable. Hence, device substitution may be required. How can the device be substituted?
- Validation of the workflow
  Validation of the workflow can be done during design or/and run time. We currently investigate the possibility of using Petri Net as a tool to validate the workflow. Some of the design validations include deadlock detection.
- Diagrammatic user interface
  Currently we have Eco as a command based user interface to program the device ecology. Some users may prefer to work with diagrammatic tool to design the workflow. This tool will be useful in particular in designing a complex workflow.

We have so far considered a centralised engine for executing the decoflow. It would be interesting to investigate further a possibility of using the multiagent distributed workflows using decentralised peer-to-peer model in order to provide greater flexibility and facilities to perform on-the-fly and just-in-time ad-hoc workflow. It will alleviate the problem of a single point of failure in the system.

VII. Reference

[1] AHAM, Connected Home Appliances – Object Modelling, 2002, AHAM-CHA-1-2002.

[2] E. Bergaman, Information Appliances and Beyond, 2000, Morgan Kaufmann.

[3] Ikeda, M., Seta, K., Kakusho, O., and Mizoguchi, R. "An Ontology for Building a Conceptual Problem Solving Model.", In *ECAI98 Workshop on Applications of ontologies and problem-solving model*, 1998, pages 126–133, Brighton, England.

[4] K. Matsuura, T. Hara, A Watanabe and T. Nakajima, "A New Architecture in Home Computing," in *Proceedings of 1AD: First International Conference in Appliance Design, 2003*, pp 57-63.

[5] Microsoft, IBM, Siebel, BEA, and SAP (2003). *Business Process Execution Language forWeb Services Version 1.1*. Available at http://www-106.ibm.com/developerworks/library/ws-bpel/.

[6] Nachira, F. 2002. Towards a Network of Digital usiness Ecosystems Fostering the Local Development. European Commission Discussion Paper. Accessed 25th September 2006 at http://www.digitalecosystem.org/html/repository/dbe_discussionpaper.pdf

[7] D. Norman, The Invisible computers, 1999, The MIT Press.

[8] Omojokun, O. and Dewan, P, " A High-Level and Flexible Framework for Dynamically Composing Networked Devices." In *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems and Applications,* 2003.