Pervasive Computing

Mobile computing is characterized and driven by portable, lightweight hardware, wireless communication, and innovations in application and system software. The technology drives new applications which in turn fuel the demand for the new technology. Pervasive computing uses small, battery-powered, wireless computing and sensing devices embedded in our environment to provide contextual information to new types of applications. The trend of technology getting smaller and more portable is likely to continue for the foreseeable future.

See chapter 4.

The success of mobile computing is contingent on how gracefully the system adapts to changes in the environment. There are two approaches to developing adaptive mobile systems: the *application transparent* approach and the *application-aware* approach. The application-transparent approach uses system software, such as the underlying operating system and networking software, to adapt to changes in operating conditions, such as a substantial reduction in available bandwidth or a drop in the level of battery power, in an application independent manner. In contrast to this approach, application-aware adaptation uses collaboration between the system software and the application software to adapt to changes in the availability of computing and communication resources.

A step further in the direction of application-aware adaptation is the *context-aware computing paradigm.* **In context-aware computing, the application adapts not only to changes in the availability of computing and communication resources but also to the presence of contextual information,** such as who is in the vicinity, the time of day, where the system currently resides, the current emotional state of the user, the action the user is performing, and the intention with which that action is being performed. A context-aware (or context-sensitive) application requires contextual information that must be gathered from various sources, such as sensors that are embedded in the environment, devices that are worn by end users, repositories of historical data tracking use of the application, and information contained in user profiles.

Example:

You are a tourist visiting Kolkata for the first time. You have your PDA with wireless service from a service provider that covers Kolkata. You have not made hotel reservations in advance. However, you have a context-based Web information service (WISE) available to you from your wireless service provider. The first thing you do is ask WISE for suggestions for hotels. WISE takes into account information such as your itinerary, preferences (such as price range), and current location and provides you with a list of hotels that are nearby. You then select one of the hotels and invoke an online hotel reservation form. Most of the details in the form are filled in automatically based on your preferences, which the system reads from your user profile. For example, it knows that you prefer to pay the hotel bill using your credit card, which earns you frequent flier miles, and that you favor staying in nonsmoking rooms.

How such context-aware applications can be developed?

History of Ubiquitous/Pervasive Computing

Pioneering work in context-aware computing was started in the early 1990s at Xerox PARC Laboratory and Olivetti Research, Ltd. (now part of AT&T Laboratories Cambridge), under the vision of *ubiquitous computing.*(Since the mid-1990s, ubiquitous computing also has been known as *pervasive computing.*) Marc Weiser, in his seminal paper entitled, "The Computers of 21st Century" (Weiser, 1991), envisioned that in accordance with Moore's law, future computing environments would consist of very cheap (disposable) interconnected specialized computers all around us, some embedded in our surroundings and others worn by us (Fig. 4.1). However, if the usage model of ubiquitous computing systems follows the trend of the usage model of mainframe and personal computers, where a substantial effort is required on the part of users to accomplish any computing or communication tasks, then we would be constantly distracted by these numerous devices.

What is Ubiquitous/Pervasive Computing?

The aim of ubiquitous computing is to design computing infrastructures in such a manner that they integrate seamlessly with the environment and become almost invisible. This is analogous to the "profound technologies," such as the electric motor (a typical automobile has more than 25 motors) and writing technology (our environment is equipped with whiteboards, notepads, Post-It notes for various needs without being a distraction), that "weave themselves into the fabric of everyday life until they are indistinguishable from it" (Weiser, 1991).

The context-aware mobile computing is essential to meet the goals of ubiquitous computing. Motion is an integral part of our daily life, and any ubiquitous system that does not reasonably support mobility of computing devices will have difficulty in becoming "invisible" to the user. Essentially, this form of computing is broader than mobile computing because it concerns not just *mobility of computers* but, more important, *mobility of people*.

What Is a Context? Definitions and Types of Contexts

We routinely use contextual information, such as who is in our vicinity or where we are, to modulate our responses to or interactions with other people. As we mature, we learn which contextual information is important given a particular situation. For example, when we are with our close friends, we rarely worry about how we speak: However, in a formal situation, we are careful in what we say and how we say it.

To use contextual information in adapting applications, there should exist a means to capture, store, and process context. A computer friendly definition of context fall under two broad categories such as *enumeration-based* and *role-based*. Context consists of the following categories:

Enumeration-based:

- *Computing context* includes network connectivity, communication costs, communication bandwidth, and local resources, such as printers, displays, and workstations.

- *User context* includes user profiles, location, and people in the vicinity of the user.

- *Physical context* includes lighting and noise levels, traffic conditions, and temperature.
- *Temporal context* includes time of day, week, month, and season of the year.

- *Context history* is the recording of computing, user, and physical context across a time span.

As we develop and use increasingly context-aware applications, a universally acceptable definition of context may evolve. The following five W's of context can form the core of different context types used by an application (Abowd and Mynatt, 2000):

- *Who (social context):* This consists of information such as user identification and identification of people near the user. A context-aware system can use the identification of the person who is using the system to determine how to respond based on the user's preference.

- *What (functional context):* This consists of information about what tasks the user is performing.

- *Where (location context):* This consists of information about where the system is currently located. This information can be raw location information, such as the latitude and longitude of the user, or it can be obtained at a higher level, such as the number of the room in which the system is currently operating. Location context is the most prolifically used type of context for developing context-aware applications.

- *When (temporal context): Temporal context* includes time of day, week, month, and season of the year, which is same as mentioned earlier.

- *Why (motivating context):* This specifies why the user is performing a certain task. This is one of the most difficult types of contextual information to determine.

In addition to these different kinds of contextual information, there are two more contexts:
Emotional context: user's emotional state

Environmental context: information about the environment, such as room temperature and illumination level

Context can be categorized further into low-level context and highlevel context.

Low-level context information can be sensed directly using sensors or through simple processing, e.g., by accessing a database, e.g., room temperature or devices that are in the vicinity of a user.

High-level context information may involve the amalgamation of low-level context information as well as sophisticated processing, such as machine vision or artificial intelligence (AI) techniques.

For example, low-level context information such as the current location of the user and the current time can be combined from the user's calendar to obtain information about a user's current social situation, inferring that the user "is in a meeting,", "attending a lecture," or "waiting at the airport."

Role-based:

Context is defined from another perspective -- the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user.

Based on this definition, two types of context can be identified -- *Active context and Passive context*

Active context is the contextual information used by the application to adapt its behavior, whereas *passive context* is the contextual information that is not critical for application adaptation but is provided to the user to enhance his or her understanding of the situation.

Context-Aware Computing and Applications

Context-aware computing devices and applications respond to changes in the environment in an intelligent manner to enhance the computing environment for the user. Context-aware applications tend to be mobile applications for obvious reasons:

(1) The user's context fluctuates most frequently when a user is mobile

(2) the need for context-aware behaviour is greatest in a mobile environment.

A context-aware application can perform various tasks, which may include providing a context-aware user interface, presenting contextual and non contextual information to the user, context-sensitive information services, and either proactive (such as automatic reconfiguration) or reactive (such as context-sensitive querying) context-aware adaptation of behaviour.

Core capabilities for context awareness

Applications should posses some of the following capabilities in order to be characterized as context-aware:

- *Contextual sensing* refers to the detection of various environmental states and how they are presented to the user. For example, the location information obtained from a location sensor such as global positioning satellite (GPS) device can be presented to the user via a map display annotated with a "you are here" marker.

- *Contextual adaptation* is the capability of the system to adapt its behaviour by using contextual information.

- *Contextual resource discovery* is the capability by which a system can discover available resources, which it can use to better adapt to the user's needs.

- *Contextual augmentation* refers to having the capability to associate contextual information with some digital data.

Types of context-aware applications

Context aware applications can be classified along three different dimensions: function or service type, initiating agent, and adaptation.

Function or service type. Context-aware applications have been developed to perform various tasks. However, they can be broadly classified into applications whose primary task is related to providing information or actuating some command.

Initiation. The context-aware application either can be initiated explicitly by the user (manual) or can be invoked implicitly by the application.

Adaptation

Contextual selection refers to the selection and presentation of physical or virtual objects based on the user's context. The most important type of contextual selection in mobile computing is *proximity selection* where the selection of objects is based primarily on the location context of the user, e.g., (1) selection of resources and devices such as printers or speakers in the vicinity of the user, (2) selection of places such as gas stations and restaurants closest to the current location of the user, and (3) selection or addressing of objects with which the user is currently interacting, such as the group of people in the room to whom the user wants to send information. Under the umbrella of ubiquitous computing, many other types of contextual selections are possible. For example, one can perform selection based on the social context of the user.

Contextual selection applications need to employ various user-interface techniques to assist the user in the selection of objects in the physical or virtual world based on the user's context. In proximate selection, applications may employ various visual effects to convey the relative ordering of the objects to the user. For example, different sized fonts can be used to display a spatial relationship between an object and the user (the closer the object, the larger is the font size).

In order to minimize computation and communication resources, a contextual selection application may fine-tune the granularity and accuracy of information presented to the user. Suppose that it is almost lunchtime, and you are driving around in an unfamiliar city looking for a restaurant. Under such a scenario, the application may update the information regarding closer restaurants more frequently than that of distant restaurants.

Contextual information applications modulate their responses based on the context of the user. In contrast, *contextual selection applications* provide information about the context

itself. These applications can be viewed as parameterizing user's queries with contextual information. The main contextual information that is used is the location of the user.

Similar to contextual information applications, *contextual commands* modulate their behaviour based on the current context of the user. For example, a print command may, by default, print to the printer nearest to the user.

Automatic contextual reconfiguration applications automatically adapt the system configuration in response to a change in context. For example, the application may configure itself to use the display device available to the user.

Context-triggered actions are simple IF-THEN rules used to specify how context-aware software should respond automatically to contextual changes. The condition in the IF-THEN statement is based on the contextual predicates. There is an action associated with each IFTHEN rule that is performed when the associated condition becomes true.

Developing context-aware applications

In general, the following steps are used for developing context-aware applications:

1. Identifying relevant context
2. Specifying context-aware behaviours
3. Integrating with mechanisms for acquisition of contextual information

The first step is application-dependent, and the third step is platform dependent. Thus the discussion will be restricted to the techniques to achieve the second step, which is used in several applications. Let us look at two different approaches for specifying context-aware behaviour such as context-triggered actions and Stick-E notes.

Middleware Support

Context-aware applications need support for the acquisition and delivery of contextual data. The main complication in developing context-aware applications stems from the very nature of contextual information:

- It is acquired from various heterogeneous and distributed sources:
    - *Hardware and software sensors*—obtained from various sensors, such as motion detectors, noise and temperature sensors, and location systems.
    - *System recorded input*—such as user-system interaction history. Context information history is essential for applications such as context-based retrieval.
    - *Other applications:*
        - *User's personal computing space*—such as those obtained from schedules, calendars, address books, contact lists, and to-do lists.

- *Distributed computing environment*—such as those obtained from applications running in the vicinity of these devices, e.g., services provided by the infrastructure of a shopping mall or freeway system.

Summary

Context-aware computing is a new and rapidly evolving field. Already we are seeing availability of context-aware applications. For example, the Google search engine has started using location information to provide location-dependent results to search queries (see *http://local. google.com/lochp*).

This topic has described some fundamental issues related to context-aware computing, such as what is a context and how can a context-aware behaviour be specified. We also looked at some of the middleware support and services that are being developed to help in the development of more sophisticated context aware applications.

Location and Service discovery

Service discovery is the process of automatically detecting devices and **services** on a network. **Service discovery** protocol (SDP) is a networking standard that accomplishes detection of networks by identifying resources.

In a micro services application, the set of running **service** instances changes dynamically. Instances have dynamically assigned network **locations**. Consequently, in order for a client to make a request to a **service** it must use a **service-discovery** mechanism. A key part of **service discovery** is the **service registry.**