Context – Aware Computing

Unit IV-First part
Ubiquitous/Pervasive Computing

• The general trend of computing is to have devices seamlessly integrated into the life of users and having services readily available to everywhere users go

• It is an emerging paradigm to free everyday users from manually configuring and instructing computer systems

• Allow us “to do more, by doing less”
What is Context?

- Context is that which surrounds, and gives meaning, to something else
- Context is any information that can be used to characterize the situation of an entity
  - Typically the location, identity and state of people, groups, computational and physical objects
  - May come from disparate sources and has a relatively transient lifetime
    - But historic data about context is important, anyway
Context-Aware Computing

• Not just “deliver any service at any time, anywhere”, but rather “delivering the right service at the right moment”

• Mobile computing is introducing the possibility that the physical and logical context of a user might influence the behavior of services called for
  – Mobile computing decouples function from location
    – User location is transparent to function
  – Recent trends are extending this concept of context to include many other facets of the user’s physical environment
    – Many sensors are being added to characterize context
Types of Context

- **Enumeration based**: In which context is defined in terms of its various categorization.
- **Role based**: In which context is defined in terms of its role in context-aware computing.
Enumeration based Context

• Categories:
  • **Computing context:** Includes network connectivity, communication cost, communication bandwidth, and local resources, such as printers, display, and workstation.
  • **User context:** Include user profiles, location and people in the vicinity of the user.
  • **Physical context:** Include 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.
The following five W’s of context can form the core of different context types used by application:

- **Who**(social context): This consists of information such as user identification and identification of people near the user
- **What**(Functional context): This consists of information about what task the user is performing
- **Where**(Location Context): This consists of information about where the system is currently located
- **When**(temporal): Includes time of day, week, month, and season of the year.
- **Why**(Motivational context): This specifies why the user is performing a certain task.
Context-Aware computing and Applications

• It responds to change in the environment in an intelligent manner to enhance the computing environment in an intelligent manner to enhance the computing environment for user.

• Context aware application tend to be mobile application for the reason:

  • The user's context fluctuates most frequently when a user is mobile

  • The need for context aware behavior is greatest in a mobile environment
• **Core capabilities for Context awareness:**

  - Contextual sensing
  - Contextual adaptation
  - Contextual resources discovery
  - Contextual augmentation

• **Types of Context aware application:**

  - **Function or service type:** Application have been developed to perform various tasks.
  - **Initiation:** Application either can be initiated explicitly by users (Manual) or can be invoked implicitly by the application
  - **Adaptation:** Adaptation performed by a context-aware application consist of various types: information, system, user interface, and command (behavior)
Developing Context-Aware Application

- **Steps for developing context-aware application:**
  1. Identify relevant context
  2. Specifying context-aware behaviors.
  3. Integrating with mechanism for acquisition of contextual information
- The first step is application dependent and third is platform dependent
- **Hence for second steps two different approaches to context-aware behaviors.**
  1. Context triggered action
  2. Stick-E notes
Context triggered action

- **Watchdog and contextual for active badges**
- The *watchdog program* was designed for a *UNIX environment* that is coupled with an *Active Badge Location System*
- The *active Badge* is an *electronic tag* that periodically broadcasts a unique identifier for the purpose of *determining the location* of the wearer
- Watchdog monitors Active Badge activity and execute relevant *UNIX shell commands* as required.
- The user specifies the context triggered action in a configuration file when the watchdog program is first started
- File contains *Active Badge events* and *action to perform*
- **File Format:** *John location event-type action*
- Whenever an event of the *event-type* is generated by the badge at the specified *location*
- *E.g. John any attention “emacs –display $NEARESTHOST: 0.0”*
- Specifies that *emacs window* should *start* at a *nearby host* whenever the *attention signal* is received from *tag john*
• **Contextual Reminder** is an application for ParcTabs
• Contextual Reminder provide a more expensive way of specifying reminders
• *E.g.* shows a set of predicates that makes use of date, time, location, and proximity to another person to trigger a reminder

  After April 15  
  Between 10 and 12 noon  
  In room 35–2200  
  With {user Adams}  
  With {Type Display} having {Feature Color}
Stick-E Note

- **Stick-E Note** is a technology that has been developed to facilitate the creation of context-aware applications by nonprogrammers.
- It is motivated by the paradigm of the *Post-It Note*, those yellow sticky notes used to put down reminders at prominent spots in one’s environment.
- The **Stick-E Note** is designed with the assumption that users moving with PDAs.
- The PDA has wireless connectivity to a communication network and is equipped with various sensors such as a GPS transceiver.
- Each Stick-E note consists of two parts:
  - **Context**: A context that the Stick-E Note is attached that contains location, the identity of nearby users and a time (where, who, and when).
  - **Content**: The content that the note represents, i.e., information, actions, and interfaces.
A Stick-E Note is a Standard Generalized Markup Language (SGML) document with a context section and a body.

E.g. Stick-E Note will display a reminder on one’s PDA to pick up a library book when the person is in the vicinity of particular library.

<note>
<at>“Noble engineering Library”
<body>
Pick up the book from interloan library section.
Introduction to Mobile Middleware

Unit IV - Second part
What is Mobile Middleware?

- **Middleware** is software that supports *mediation between* other software components, fostering(assist) interoperability between those component across *heterogeneous platforms* and *varying resource levels*.

- *E.g. Middleware* can serve as plumbing, allowing application that *do not normally support disconnected operation* to do so through clever use of data hoarding

- Ideally, beyond bridging heterogeneous system, middleware should be *transparent, robust, efficient, secure, and based on open standards*.
Adaptation

• Finally will narrow the gap between mobile devices and traditional wired devices.
• The gap never really narrows. The hunger for computing resources is insatiable, and new, hungrier applications always appears.
• The new desktop computer shames (a thing that is not what it is purported to be.) the newest PDA, and 10-GB Ethernet makes third-generation (3G) wireless seem not so fast after all.
• Therefore, mobile application must fight to make the computing experience of their users tolerable -to make that gap seem not quite so wide.
• So mobile application must adapt their behaviors and expectations to conserve scarce resources and to adjust quality of service (QoS). Essentially, a guarantee of performance – to sustainable level
Agents

- **Mobile agents** put the action where the data are, allowing programs to move autonomously about a network in order to access remote resources.
- In a **mobile agent system**, programs migrate directly to servers, gain access to data or computational resources and potentially migrates again, eventually returning to their “home base” to deliver results.
- Benefits is **disconnected operation** is easily supported.
- Another benefits is that **agents can gain** access to large amounts of data to solve a problem, even if a mobile users network resources are scant (limited).
- Third benefits is that **mobile agent system allow** the functionality of servers to be **expanded dynamically**.
Service Discovery

• Service discovery middleware extends the client-server (CS) paradigm to introduce dynamic discovery of services and more dynamically interaction between clients and services.

• This type of middleware directly supports the extended client server model.

• It is appropriate for both traditional wired networks and wireless networks but is particularly exciting for mobile computing environments because it allows peripheral-poor mobile devices to discover needed services on demand.
Middleware for application Development: Adaptation and Agent

Unit IV-Third part
Adaptation

• Mobile computers must execute user- and system-level applications subject to a variety of resource constraints that generally can be ignored in modern desktop environments.

• The most important of these constraints are power, volatile and nonvolatile memory, and network bandwidth, although other physical limitations such as screen resolution are also important.

• For example, given a sudden severe constraint on available bandwidth, a mobile audio application might stop delivering a high-bit-rate audio stream and substitute a lower-quality stream.
The spectrum of adaptation

- At one end of the spectrum, adaptation may be entirely the responsibility of the mobile computer’s operating system (OS),

- That is, the software for handling adaptation essentially is tucked (Push or insert) under the OS hood, invisible to applications

- i.e. adaptation may be entirely the responsibility of individual applications; that is, each application must address all the issues of detecting and dealing with varying resource levels.

- While applications are involved in adaptation decisions, the middleware and/or OS provides support for resource monitoring and other low-level adaptation functions
Resource monitoring

• All adaptation strategies must measure available resources so that adaptation policies can be carried out.
• For some types of resources—cash, for example—monitoring is not so difficult. The user simply sets limits and appropriate accounts.
• The Advanced Configuration and Power Interface (ACPI) provides developers with a standardized interface to power-level information on modern devices equipped with “smart” batteries.
• Measuring network bandwidth over multihop networks is more difficult.
• Whatever methods are used to measure resource levels have a direct impact on the effectiveness of the entire adaptation process because accurate measurement of resource levels is critical to making proper adaptation decisions.
• * Multi-hop, or ad hoc, wireless networks use two or more wireless hops to convey information from a source to a destination.
Figure 6.1 At one end of the spectrum of adaptation, applications are entirely responsible for reacting to changing resource levels. At the other end of the spectrum, the operating system reacts to changing resource levels without the interaction of individual applications.
Characterizing adaptation strategies

- We describe these—fidelity, agility, and concurrency
- Fidelity measures the degree to which a data item available to an application matches a reference copy.
- The reference copy for a data item is considered the exemplar, the ideal for that data item—essentially, the version of the data that a mobile computer would prefer given no resource constraints.
- Fidelity spans many dimensions, including perceived quality and consistency.
- For example, a server might store a 30 frame-per-second (fps), 24-bit color depth video at 1600 × 1200 resolution in its original form as shot by a digital video camera.
- This reference copy of the video is considered to have 100 percent fidelity.
Cont..

- **Agility** measures an adaptation middleware’s responsiveness to changes in resource levels.
- *For example,* a highly agile system will determine quickly and accurately that network bandwidth has increased substantially or that a fresh battery has been inserted.
- An adaptation middleware’s agility directly limits the range of fidelity levels that can be accommodated.
- The last measure for adaptation middleware that we will discuss is *concurrency*.
- Thus it is reasonable to expect that even the least powerful of mobile devices, not to mention laptops that run desktop operating systems, will execute many concurrent applications, all of which compete for limited resources such as *power and network bandwidth*. 
An application-aware adaptation architecture: Odyssey

Figure 6.2 The Odyssey architecture consists of a type-independent viceroy and a number of type-specific wardens. Applications register windows of acceptable resource levels for particular types of data streams and receive notifications is when current resource levels fall outside the windows.
• **Wardens.** A warden is a type-specific component responsible for handling all adaptation-related operations for a particular sort of data stream (e.g., a source of digital images, audio, or video).

• **Wardens** sit between an application and a data source, handling caching and arranging for delivery of data of appropriate fidelity levels to the application.

• **A warden** must be written for each type of data source.

• **Viceroy.** In Odyssey, the viceroy is a type-independent component that is responsible for global resource control.

• All the **wardens** are statically compiled with the **viceroy**.

• The **viceroy** monitors resource levels (e.g., available network bandwidth) and initiates callbacks to an application when current resource levels fall outside a range registered by the application.
A sample Odyssey application

Figure 6.3 Architecture of the adapted video player in Odyssey.
The `xanim` video player was modified to use Odyssey to adapt to varying network conditions, with three fidelity levels available—two levels of JPEG compression and black-and-white frames.

The JPEG compression frames are labeled 99 and 50 percent fidelity, whereas the black-and-white content is labeled 1 percent fidelity.

Integration of `xanim` with Odyssey is illustrated in Fig. A “video warden” perfectness frames from a video server with the appropriate fidelity and supplies the application with metadata for the video being played and with individual frames of the video.
Figure 6.4  (a) Illustrates the overall Puppeteer architecture, where client applications interact with data servers through proxies. DMI is the Data Manipulation Interface of the applications, which allows Puppeteer to view and modify data acted on by the application. The relationship between client-side and server-side proxies is illustrated in (b).
The Puppeteer architecture is depicted in Fig. The Puppeteer provides a kernel that executes on both the client and server side proxies, supporting a document type called the Puppeteer Intermediate Format (PIF), a hierarchical format, and neutral format. The kernel also handles all communication between client and server sides. To adapt a document, the server and client side proxies communicate to establish a high-level PIF skeleton of the document. Adaptation policies control which portions of the document will be transferred and which fidelities will be chosen for the transmitted portions. The import driver and export driver parse native document format to PIF and PIF to native document format, respectively. For example, a Puppeteer transcoder may reduce the quality of JPEG images or support downloading only a subset of a document’s data. (Adaptation policies)
Mobile Agents

• Almost all computer users have used mobile code, whether they realize it or not—modern browsers support JavaScript, Java applets, and other executable content, and simply viewing Web pages results in execution of the associated mobile code.

• For security reasons, the mobile code often is prevented from touching nonlocal resources.

• Mobile agents are a significant step forward in sophistication, supporting the migration of not only code but also state. Like adaptation middleware, mobile agent systems support execution of mobile applications in resource-limited environments.

• Like adaptation middleware, mobile agent systems go far beyond allowing local applications to respond to fluctuating resource levels.

• An agent can migrate whenever it chooses either because it has accomplished its task completely or because it needs to travel to another location to obtain additional data.
Why mobile agents? And why not?

• A wide variety of applications can be supported by mobile agent systems,
• Covering electronic commerce (sending an agent shopping)
• Network resource management (an agent might traverse the network, checking versions of installed applications and initiating upgrades where necessary),
• Information retrieval (an agent might be dispatched to learn everything it can about Thelonious Monk).
• Mobile agent systems provide the following set of technical advantages:
  1. The limitations of a single client computer are reduced.
  2. The ability to customize applications easily is greatly improved.
  3. Flexible, disconnected operation is supported.
Agent architectures
• The major Typescript components are illustrated in fig. Tom has just dispatched an agent which has not yet arrived at the theater server.
• When Tom’s agent arrives, it will interact with the static agent in the box office place to arrange for theater tickets.
• Daryl previously dispatched an agent to purchase tickets and has a connection with her agent in the box office place, so she can actively negotiate prices.
• Daryl’s agent and the box office agent have identified each other through their respective authorities and permits associated with Daryl’s agent have been evaluated to see what actions are permitted.
• The static agents in the drugstore and music store places, which both reside on a shopping center server, are currently idle.
• To interact with the drugstore or music store agents, Daryl or Tom’s agents will have to travel to the drugstore and music store places.
• There are a number of important components in the Telescript architecture:
  • **Places**: In a mobile agent system, a network is composed of a set of places—each place is a location in the network where agents may visit.
  • **Travel**: Travel allows agents to move closer to or to collocate with needed resources
  • **Meetings**: Meetings are local interactions between two or more agents in the same place.
  • **Connections**: Connections allow agents at different places to communicate and allow agents to communicate with human users or other applications over a network.
  • **Authorities**: An agent’s or place’s authority is the person or organization (in the real world) that it represents.
  • **Permits**: Permits determine what agents and places can do—they are sets of capabilities
  • **Other issues** A number of details must be taken into account when designing an architecture to support mobile agents