Unit – 4

Middleware For Pervasive
Introduction Mobile Middleware

- What is Mobile Middleware
- Mobile Middleware for Enterprise
- Adaptation
- Agents
- Service Discovery
What is Mobile Middleware

Definition

• “Middleware is software that supports mediation between other software components, fostering interoperability between those components across heterogeneous platforms and varying resource levels”.
Definition:
- Mobile middleware allows for the implementation of distributed applications connecting mobile and enterprise applications over wireless networks.
- Provide the “black box” technology that connects mobile devices on the front lines of the enterprise to the back-end applications running on corporate servers.
Middleware for Enterprise

Example Applications

- Wireless email
- Speech middleware
- Firewall and mobile VPN (Virtual Private Network)
- Network connectivity
- Device management
- Enterprise Access
  - CRM (Customer Relationship Management)
  - EAI (Enterprise Application Integration)
  - Enterprise data & information integration
  - Insurance Claim
3 major types of middleware for mobile computing:

1. Adaptation
   - it allows applications to offer reasonable performance to users across different environments.

2. Mobile Agent
   - it extend the reach of data servers & help mobile devices conserve energy.

3. Service Discovery
   - it allow mobile devices to change configuration quickly and easily, depending on available services.

Note: Context Aware Computing can also be encapsulated in middleware frameworks
Adaptation

Tasks

• Adapt behavior and expectations to conserve scarce resources
• Adjust quality of service (QoS) – guarantee performance

How should adaptation be supported?
  – Monitor resources and adapt appropriately

Eg. Video player
Agents

- Allowing programs to move autonomously about a network in order to access remote resources
  - Migrate to servers -> access data or computational resources -> migrate again -> return to home base

Benefits
- Disconnection is easily supported
- Access to large amount of data to solve problem
- Allow the functionality of servers to be expanded dynamically
Service Discovery

- Extend the client-server paradigm
- Discover needed service on-demand
- Bluetooth Service Discovery Protocol
1. Adaptation Middleware

- Mobile computers must execute user- and system-level applications
- But they are 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
- Provide user with reasonable computing environment
- adapt to limited or fluctuating resources level
Example of Adaptation Middleware

- Constraint on Bandwidth – Mobile audio application – Stop delivering High bit Audio Stream and Substitute lower quality Audio Stream

- Video Application – From High frame rate colour video to black n White video to colour still images to black n white still images

- Video Game application – Adjusting decreased battery levels by modifying resolution and disabling 3D features
## 1.1 Spectrum of Adaptation

- Two ends of 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 under the OS hood, invisible to applications.

  • At the other end, 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.
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.
• Between these extremes, a number of application-aware strategies are possible, where the OS and individual application each share some of the burden of adaptation.

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

- All adaptation strategies must measure available resources so that adaptation policies can be carried out.

- Whatever methods are used to measure resources levels have a direct impact on the effectiveness of the entire adaptation process because accurate measurement of resources levels is critical to making proper adaptation decision.
1.3 Characterizing adaptation Strategies

- The odyssey Project was the first application aware middleware system and a good model for understanding.
- Several measures are proposed that are useful for classifying the goodness of an adaptation strategy.
- 3 Strategies – Fidelity, Agility, Concurrency
  - Fidelity (Exact copy) – Measures degree to which data items available to an application matches a reference copy(appropriate Model).
  - Agility (Fast and graceful) – Measures Adaptations middleware responsiveness to changes in resource level.
  - Concurrency (Distributed) – Will execute many concurrent applications all of which compete for limited resources such as power and network bandwidth
Fidelity

- *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.
• Owing to resource constraints such as limited network bandwidth, a mobile host may have to settle for a version of this video that is substantially reduced in quality (assigned a lower fidelity measure, perhaps 50 percent).

• or even for a sequence of individual black-and-white still frames (with a fidelity measure of 1 percent).

• If the video file on the server is replaced periodically with a newer version and a mobile host experiences complete disconnection, then an older, cached version of the video may be supplied to an application by adaptation middleware.

• Even if this cached version is of the same visual quality as the current, up-to-date copy, its fidelity may be considered lower because it is not the most recent copy (i.e., it is stale).
Agility

- 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 should notice that power levels have dropped substantially before critical levels are reached.

- Otherwise, a user enjoying a high-quality (and power-expensive) audio stream may be left with nothing, rather than a lower-quality audio stream that is sustainable.

- Agility, however, is not simply a measure of the speed with which resource levels are measured; accuracy is also extremely important.
• Handling adaptation at the left end of the spectrum, where individual applications assume full responsibility for adapting to resource levels, is probably not a good idea.

• To make intelligent decisions,
  • each application would need to monitor available resources,
  • be aware of the resource requirements of all other applications,
  • and know about the adaptation decisions being made by the other applications.

• Thus some system-level support for resource monitoring, where the OS can maintain the “big picture” about available resources needs and resource levels, is important.
An application-aware adaptation architecture: Odyssey

- In the spectrum of adaptation, Odyssey sits in the middle—applications are assisted by the Odyssey middleware in making decisions concerning fidelity levels.

- Odyssey provides a good model for understanding the issues in application-aware adaptation.

- The Odyssey architecture consists of several high-level components: the interceptor, which resides in the OS kernel, the viceroy, and one or more wardens.
1.3 An application-aware adaptation architecture

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.
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.

An application typically must be partially rewritten (or an appropriate proxy installed) to accept data through a warden rather than through a direct connection to a data source, such as a streaming video server.
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.
- The types of resources to be monitored by the viceroy in Odyssey include network bandwidth, cache, battery power, and CPU.
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.
Figure 6.3  Architecture of the adapted video player in .y.
• A “video warden” prefetches frames from a video server with the appropriate fidelity
• supplies the application with metadata for the video being played and with individual frames of the video.
• Odyssey maintained average fidelities of 73, 76, 50, and 98 percent for step up, step down, impulse up, and impulse down, respectively, all with less than 5 percent dropped frames.
• In contrast, trying to maintain the 99 percent fidelity rate by transferring high-quality video at all times, ignoring available network bandwidth, resulted in losses of 28 percent of the frames for step up and step down and 58 percent of the frames for impulse up.
Puppeteer.

- For applications with well-defined, published interfaces, it is possible to provide adaptation support without modifying the applications directly.
- The Puppeteer architecture allows component-based applications with published interfaces to be adapted to environments with poor network bandwidth without modifying the application.
- This is accomplished by outfitting applications and data servers with custom proxies that support the adaptation process.
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).
Working of a typical Puppeteer-adapted application

- When the user opens a document, the Puppeteer kernel instantiates an appropriate import driver on the server side.

- The import driver parses the native document format and creates a PIF format document. The skeleton of the PIF is transmitted by the kernel to the client-side proxy.

- On the client side, policies available to the client-side proxy result in requests to transfer selected portions of the PIF (at selected fidelities) from the server side.

- These items are rendered by the export driver into native format and supplied to the application through its well known interface.

- At this point, the user regains control of the application. If specified by the policy, additional portions of the requested document can be transferred by Puppeteer in the background and supplied to the application as they arrive.
Coordinating adaptation for multiple mobile applications

- When multiple applications are competing for shared resources, individual applications may make decisions that are suboptimal.

- At least three issues are introduced when multiple applications attempt to adapt to limited resources
  - conflicting adaptation,
  - suboptimal system operation,
  - And suboptimal user experience.
Suboptimal system operation

• For example, if two or more applications with automatic backup features were executing.

• Imagine that the mobile host maintains a powered-down state for its hard drives to conserve energy.

• Then, each time one of the automatic backup facilities executes, a hard disk on the system must be spun up.

• If the various applications perform automatic backups at uncoordinated times, then the disk likely will spin up quite frequently, wasting a significant amount of energy.

• If the applications coordinated to perform automatic backups, on the other hand, then disk writes could be performed “in bulk,” maximizing the amount of time that the disk could remain powered down.
Conflicting Adaptation

• Imagine that one application is adapting to varying power, whereas another application is adapting to varying network bandwidth.

• When the battery level in the mobile device becomes a concern, then the power-conscious application might throttle its use of the network interface.

• This, in turn, makes more bandwidth available, which might trigger the bandwidth-conscious application to raise fidelity levels for a data stream, defeating the other application’s attempt to save energy.
Suboptimal User Experience

• A third issue is that in the face of limited resources, a user’s needs can be exceedingly difficult to predict.

• Imagine that a user is enjoying a high-bandwidth audio stream while downloading a presentation she needs to review in 1 hour.

• With abundant bandwidth, both applications can be well served.

• However, if available bandwidth decreases sharply, should a lower-quality stream be chosen and the presentation download delayed?

• Or should the fun stop completely and the work take precedence?
2. Mobile Agents

- Supporting migration of code as well as state.
- Mobile agents move freely about networks, making autonomous decisions on where to travel next.
- Support execution of mobile applications in resource limited environments
- It is dynamic client server architecture
- Mobile agents migrate to remote machines to perform computations then return home results.
2.1 Advantages

- The limitation of single client computer are reduced
- The ability to customise applications easily is greatly improved
- Flexible, Disconnected operation is supported
2.2 Agents Architectures

- Telescript architecture was one of the first mobile agent system.

- Components in Telescript architecture: Agents, Places, Travel, Meetings, Connections, Authorities, Permits
2.2 Migration Strategies

- To support migration agents, it must be possible to capture the state of an agent or to spawn an additional process that capture the state of an agent
- This Process state is transmitted to remote machine to which agent will migrate.
- Similar to process check pointing
2.2 Communication Strategies

- Traditional CS Communication, RPC, RMI, mailboxes, meeting places, coordination languages
- Degree of temporal and spatial locality exhibited by communication schemes
- Temporal Locality – *Communication among two or more agents must take place at the same physical time.*
- Spatial Locality – *Unique names must be associated with agents and their names must be sufficient for determining their current location*
Mobile Agents: Motivation

Mobile agent systems support dynamic migration of bits of software.

These “bits of software” are agents, and perform tasks remotely in support of some local computation.

Some advantages:

- Disconnected operation
- Customized, “do once” operations without customizing server software
- Attacks resource-poverty problem for mobile computing
  - Less bandwidth, power, CPU needed
Searching Tech Reports: No Agents

Queries, responses, transfer of documents
Searching Tech Reports: Agents

MH

Mobile agent

Tech report server

local queries

<<creation of document summaries or Changes in format, e.g. PDF → TXT>>
Mobile Agents: Another Example

AGENT TRAVEL Box office PLACE MEETING Drugstore PLACE Music store PLACE Shopping center server Theater server

AUTHORITY/PERMIT evaluation

Tom Daryl CONNECTION

Music store PLACE
Mobile Agents: Roadblocks

Security

Users’ trust in mobile code has been badly damaged

Applets, Javascript, other scripting languages are far less sophisticated than agents, but…

Infrastructure

Lack of a “killer” application

Standardization

(Security)
Security!

Implementation language for agents
   Interpreted languages easier
   Don’t want to insist on homogeneous architectures!

Migration strategy
   Again, easier with languages like Java
   Otherwise, becomes a process checkpointing problem

Communication strategy
   Temporal locality: must communicate with other agents while they’re around, or not?
   Spatial locality: must be able to name and locate other agents to communicate
Agent Communication: Traditional

Sockets, RMI, …
Strong temporal locality
Other guy must be around to communicate
Other guy’s “name” reveals his location
Agent Communication: Meeting Places

A “meeting room” (aka service points) for agents

Agents must migrate to the meeting room to communicate

Temporal locality—other agents must be present to communicate

Spatial locality—communicate with other agents in the meeting place, can dynamically determine who’s