Adaptation in Mobile Computing

Marcio de Castro Marques, Antonio A.F. Loureiro

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, MG

`marciocm@dcc.ufmg.br, loureiro@dcc.ufmg.br`

Abstract. *Mobile computing is characterized by a mobile device with computing capabilities operating in a wireless environment. In order to lead with the constraints inserted into the environment by this scenario without modifying the applications, a middleware is used. This work presents the specification, design, implementation, and evaluation of a basic adaptation architecture for mobile computing. The proposed architecture defines some functionalities and services that can be provided to the applications. It runs both on the handheld device and on the gateway. A prototype is implemented. Some metrics are defined for the evaluation. Based on them, three experiments are conducted and the results analyzed to validate the specified adaptation architecture.*

1. Introduction

Mobile computing is characterized by a mobile device with computing capabilities operating in a wireless environment. This scenario inserts constraints not present in traditional fixed systems. Mobile devices have power restrictions and their computing capabilities are more restricted when compared to fixed devices. Also, wireless communication brings higher transmission error rates and lower and highly variable bandwidth. In order to offer acceptable services in this environment, mobile devices must provide some mechanisms to manage the environment modifications and to react to those changes. This problem can be mainly solved through adaptation.

Adaptation in mobile computing means the ability an application or algorithm has to output different valid results, depending on the characteristics of the environment where the mobile device is located. This is not common in fixed systems, where the conditions are practically stable. In mobile systems, however, the environment is highly variable, which leads to this solution.

There is no best strategy for adaptation. In each case, a specific strategy will fit better. This non-existence of a best strategy makes the task of defining an adaptation architecture troublesome. However, through the study of some target applications, it is possible to identify aspects that can be adapted that are more common to those applications and, thus, to define a basic adaptation architecture. In this work, a basic adaptation architecture in mobile computing is specified, designed, implemented, and evaluated.

2. Related Work and Fundamentals

Handheld devices have been gaining wide popularity in the past few years. However, applications running in those devices face new challenges introduced by both the wireless network and the devices characteristics: unexpected loss of network connections, lower and variable bandwidth size, low battery power, reduced computing capabilities, among others. In order to lead with those challenges without the need to modify the applications, a middleware should be used. In order to understand the differences between traditional and mobile middleware systems requirements, it is necessary to understand the similarities and the differences between traditional and mobile distributed systems.

2.1. Distributed Systems

A distributed system (DS) consists of a collection of autonomous components, connected through a computer network. These components may have different hardware architectures and operating systems (OSs). In a distributed system, the components may be shared, and the user perceives the whole system as a single computing facility [Coulouris et al., 1994]. The main challenges introduced by the distribution are related to the following aspects: heterogeneity, openness, scalability, resource sharing, concurrency, securty, falut-tolerance, and transparency. [Tanenbaum, 1995].

The definition for a traditional distributed system applies to mobile distributed systems. However, some key differences between these systems make it necessary to develop different middleware systems in each case [Mascolo et al., 2002]: the type of device, the type of network connection, and the type of execution context.

2.2. Middleware Systems

When developing applications for distributed systems, the aspects introduced by the distribution must be addressed. Nonetheless, making every application aware of those complexities by building them on top of the network OS primitives would make the task of programming applications for distributed systems very complex. Instead, a middleware [Bakken, 2001] may be be used.

Middleware systems can be distinguished based on the type of computational load they can execute, the communication paradigms they support, and the type of context representation they provide to the applications. In the following these points are further explained.

- **Type of computational load**: The main goal of any middleware system is to enable communication. What differs middleware systems is the reliability with which the requests are handled. *Heavyweight* systems require a large amount of resources to deliver services to the above application. However, they are more reliable. *Lightweight* systems, on the other hand, run using a minimum set of resources.
- **Type of communication paradigm**: There are two basic types of communication paradigms: synchronous and asynchronous. In the former, both the client and the server must be connected and executing at the same time, in order for the request to succeed. This is not required in the latter one.
- **Type of context representation**: The context representation identifies whether the information about the execution context is shown to the applications (*awareness*) or is used privately by the middleware and kept hidden from the applications (*transparency*).

The main goal of middleware systems designed for fixed DSs is to hide the aspects of distribution from application designers as much as possible. These systems are often resource-consuming, and most of them only support synchronous communication between components.

Middleware designed for mobile distributed systems have different requirements than the ones designed for fixed distributed systems, due to the different type of devices, network connections, and execution context. Mobile applications run on resource-scarce devices, therefore the middleware system must not be resource-consuming. Moreover, the communication between components in these systems is asynchronous. Mobile devices connect to the network for short periods of time, and during this period the available bandwidth is lower than in fixed distributed systems. At last, the context in which mobile systems execute is dynamic. The high variability of this context influences the decisions and choices from the middleware. Some information about the execution context should be passed up to the running applications, so that application-dependent information can be used.

2.3. Mobile Application Support Environment

The European ACTS project OnTheMove [Harmer, 1996] provides support services for distributed mobile multimedia applications. It defines, implements, and demonstrates a mobile middleware called Mobile Application Support Environment (MASE) [Kreller et al., 1998], based on the Universal Mobile Telecommunications Service (UMTS) [Mahony, 1998] concepts. Also, it defines a mobile application programming interface (mobile API) to allow common access to the underlying operating systems and network infrastructure through MASE.

The Mobile Application Support Environment is a DS, running on both the mobile device and the mobile gateway. The latter acts as a mediator between the wireless and fixed networks. MASE allows the mobile device to access the UMTS adaptation layer (UAL), which provides applications and middleware components unified access to all possible underlying networks.

The UAL provides monitoring and management facilities to MASE applications, which makes it possible for the applications to be aware of the current network quality of service (QoS). Moreover, the UAL also hides network specific details, selecting the appropriate bearer service and protocol stack according to the requested QoS.

The MASE mobile middleware was implemented and tested over the GSM cellular system. The results showed the advantage of the QoS trading and multimedia conversion. Depending on the conversion method automatic chosen, it achieved up to 70 percent acceleration of the HTTP transmission time. However, it is only focused on HTTP and e-mail communication. It lacks support for applications that where the transmission of video and audio is needed.

2.4. MobiWeb

MobiWeb [Margaritidis and Polyzos, 2000] a proxy-based network architecture designed to enhance the performance of adaptive real-time streams over 3G wireless links. A traditional continuous media application uses a single representation that has fixed bandwidth and timing requirement. On the other hand, an adaptive application may use different representations, separated into Levels of Quality (LoQ). Each LoQ defines the transmission characteristics required by the application in order to achieve an acceptable performance. Through the network knowledge about the wireless link quality, the adaptation decides whether or not to degrade to a lower LoQ.

MobiWeb utilizes a prioritization method for adaptation. Packets from adaptive applications are marked with their respective priority. The initial priority of an adaptive application is provided by the user.

A new stream always starts transmitting with its lowest quality, and it gradually advances to a higher one. The lower quality of the stream is associated with a higher priority. As the stream advances to a higher quality, the priority of the stream is decreased. On the other hand, as the stream drops to a lower quality, it's associated with a higher priority. This method avoids the total degradation of a stream of video, while another one is in its higher quality.

When the link conditions upgrade, the stream with the higher priority is the first to explore the available resources until it advances to a higher LoQ. In contrast, when the link conditions degrade, the stream with the lower priority will be the first one to drop to a lower quality.

Several simulation experiments were performed to demonstrate MobiWeb 's utility. They showed that the use of MobiWeb allowed the achievement of decent quality for adaptive real-time streams when the wireless link was degraded. MobiWeb's filtering mechanisms made it possible to the most important frames to be transmitted, instead of indiscriminately dropping frames. However, MobiWeb lacks support for non-adaptive real-time streams.

3. Architecture Specification

An adaptation architecture consists of a mobile middleware system. It provides high level services to the applications, hiding the complexities of the system from them. An application running in a mobile device may request for its location, for instance, without the knowledge of which mechanism will be used to gather the location data. When defining an adaptation architecture, prior to specifying the functionalities it will provide, it must be specified whether it is designed for a specific application or not, and the aspects it will focus.

3.1. Adaptation Architecture

The different types of devices, network connection, and execution context of mobile systems when compared to fixed ones insert many constraints in the mobile environment not present in the fixed one. Due to those constraints, when defining an adaptation architecture in mobile computing, various aspects should be observed: data, security, quality of service, available resources, costs, performance, and broadcast and multicast issues.

It is also important to consider whether or not the adaptation architecture is addressed to a specific application. Architectures addressed to specific applications often achieve better results, since the target application is known, and the designer may focus on its main characteristics and propose the best adaptation techniques for that application. However, those adaptation architectures can only be used for that target application. Other applications running upon this architecture will not benefit from it.

Generic architectures, on the other hand, do not address a target application. Instead, some general adaptation techniques are implemented. When an application is running upon a generic architecture, it will benefit only from the adaptation techniques that can be applied to it. Since these techniques were not designed specifically for this application, they often will not achieve the best possible adaptation level. However, they will be able to achieve good adaptation levels with many applications.

There is a trade off between generic and specific adaptation architectures. The most generic the architecture is, the most applications it can run. On the other hand, the most specific it is, the better results can be achieved for that specific application.

3.2. Architecture Specification

There are various aspects that can be adapted in a multimedia environment. Depending on the application being adapted, the importance of some aspects, such as synchronization and delay, can be greater or lesser. When specifying a generic adaptation architecture, it is important to define first the target applications for this architecture. Once this is done, it is possible to identify which adaptation techniques can be applied to those applications, and then to define the functionalities the architecture will provide.

3.2.1. Target Applications

Handheld devices computing capabilities have been growing. Some of them already have colored screens. It is already possible to access the Internet through them. Moreover, new transmission technologies providing higher bandwidth have been developed, and some of them, such as UMTS and Wi-Fi, are already in use. When defining target applications for an adaptation architecture, these facts must be considered.

Video on demand is a target application for this work. Users will be able to watch music and news clips, or even movies in their handheld devices. In video on demand, the aspects to be adapted are the video and the audio. Synchronization between the video and the audio is an important issue. In addition, the delay must be constant.

A target application similar to video on demand is video conference. Instead of making a phone conference with only voice, users will be able to see each other while talking. The aspects to be adapted in this case are the same of video on demand. However, in this case, the delay must be minimum. Another target application is audio on demand. Users could listen to Internet radio stations live, or even request songs to listen from audio on demand servers. In this case, the aspect to be adapted is the own audio. Again, there may be a delay, but it must be constant. A last target application for this work is some common Internet services, such as e-mail, ftp, or the world wide web (www). Users will eventually access the Internet for checking their e-mails, reading the news, downloading files, or visiting web pages. Each one of these services have different characteristics that must be addressed. E-mails and ftp files can be compressed. In www pages, images can be adapted, and texts compressed. There is no need for synchronization between them, and the delay may be variable.

3.2.2. Adaptation Techniques

There are many techniques to adapt each one of the aspects described previously. The best ones to be used depend on the users' handheld devices, and the level of adaptation required.

Video may be adapted in various ways. First of all, it can be colored or in gray scale. Another aspect that can be adapted in a video flow is the quality of the video. Some frames may be discarded. Also, its resolution may be set lower. On the other hand, it is not possible to discard packets while leading with audio. Broken audio is hard to understand. In this case, the aspects to be adapted are the quality of the audio and its sample rate.

Images have a different approach to be adapted. The delay here is not an issue. Nevertheless, customers would not like to wait minutes for an image to load (unless they really want that image in its higher resolution). The adaptation for images can be done in three ways: reduce the number of colors, reduce the sample of the image, or reduce its quality.

A last approach that can be used to decrease the amount of data to be sent over the wireless link is compressing texts before sending them. HTTP has an extensive header, which can also be compressed. Although compression is not considered an adaptation technique, it may be useful sometimes.

3.2.3. Functionalities and Services

Once the target applications and the adaptation techniques for those applications are defined, it is possible to specify the functionalities the adaptation architecture will provide: image resolution modification, image number of colors decrease, video packets discard, video resolution modification, video quality modification, video conversion to black and white, audio sample ratio modification, audio quality reduction, text compression, and HTTP header compression.

The adaptation architecture also provides some services to the applications: gather data location, increase or decrease quality an application quality level, increase or decrease an application priority, and enable or disable the security tools provided by the architecture.

3.3. Proposed Architecture

Many aspects must be observed when proposing an adaptation architecture in mobile computing. It must provide all the functionalities and services defined in its specification. Moreover, the characteristics of mobile systems must be taken in consideration.

The adaptation process must be handled on a gateway in the fixed network before sending the data to the handheld device. Due to the handheld device characteristics, the system running on it must be *lightweight*. Therefore, the adaptation decisions must be made in the gateway. In order to take these decisions, for each handheld device, the gateway must keep track of the applications running, the quality level for those applications, the used bandwidth, and the expected bandwidth.

Another important aspect is the allowance for the users to choose their preferences for the adaptation process. Also, the characteristics of the terminal are influential when defining an adaptation strategy. It was decided that this data should be stored in the handheld device, since each device has its own characteristics and a user with his own preferences.

During an adaptation process, the handheld device must send the information about user preferences and the terminal characteristics to the gateway, allowing the latter to take the adaptation decisions. The handheld device must also be able to request from the gateway the execution of the services provided to the applications by the middleware. In addition, the mobile device must verify the data error rate and send this information to the gateway periodically. Hence, an intelligent module is necessary on the handheld device to make this interaction with the gateway. Observing all those aspects, an adaptation architecture, shown in Figure 1, was proposed.

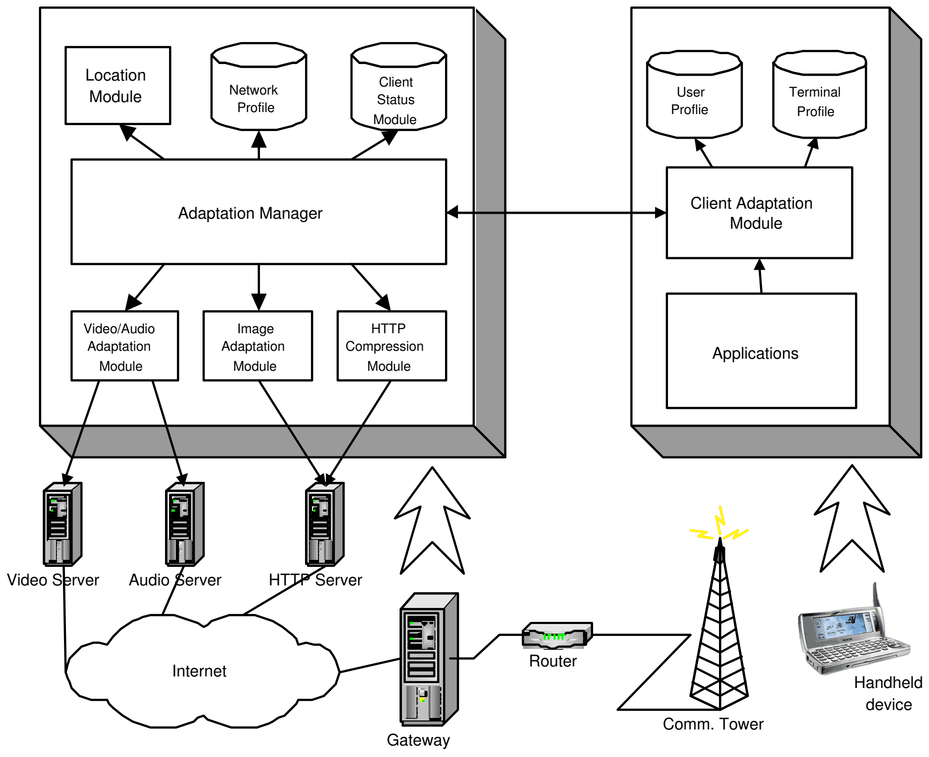


Figure 1: Proposed architecture

The handheld device is responsible for decompressing texts and HTTP headers, and for providing information to the gateway about user preferences, terminal characteristics, and the quality of the received data. The gateway, on the other hand, decides the best adaptation techniques to be used in each case based on the information provided by the handheld device, on the information about the network current characteristics, and on some rules set by the network operator, and adapts video and audio streams, images, and compress texts and HTTP headers.

The terminal characteristics of the mobile device, such as the number of colors, memory, screen and disk size, are stored in the terminal profile at the handheld device. They are important because they allow the architecture to decide what adaptation techniques may be applied *a priori*, independent of the network status and user preferences.

The user profile is also stored at the handheld device. It indicates the order in which the user prefers the adaptation techniques to be applied. It also stores some information utilized when applying those adaptation techniques, such as the maximum wait time for an image, its maximum size, and the maximum packet discard percentage for video.

The adaptation process is entirely performed in the gateway. The client status module stores information about the handheld devices, such as number of running applications, adaptation level for those applications, estimated current bandwidth for the device (based on the location of the mobile device, move direction, and signal), and utilized bandwidth.

A last profile used in this adaptation architecture is the network profile. It is also located in the gateway. It stores some values provided by the network operator, which are used to define whether or not a situation has been achieved, and in case it has, to trigger some action. The stored values are the bandwidth that must be left free after adaptation, the minimum free bandwidth necessary before an increase quality adaptation is performed, and the maximum error rate allowed.

The location module is used to calculate the current geographical position of the mobile device. It may support various mechanisms, such as GPS, to gather location data.

The most important module in the gateway is the adaptation manager. This module is responsible for communicating with location and adaptation modules, updating the client status, deciding the best adaptation to be performed, and issuing requests to the adaptation modules. There are two adaptation modules: Video/audio Adaptation Module, and Image Adaptation Module. In addition, there is also an HTTP Compression Module. These modules are responsible for requesting data from the video, audio, and HTTP servers when requested by the adaptation manager, performing the requested adaptation techniques or compressing the text and HTTP headers, and delivering the adapted data to the handheld device.

When an applications starts to run in the handheld device, it communicates with the client adaptation module. This module will communicate with the adaptation manager in the gateway, informing the desired service and the user and terminal profiles. According to the these profiles, after verifying the client status, the adaptation manager decides the best adaptation to be performed. The adaptation manager then communicates with the adaptation module responsible for that data flow, and indicates the adaptation level to be performed. The adaptation module than performs the adaptation, and the resulted stream is sent to the client adaptation module, which bypasses the stream to the application.

The user must be able to request the application at any time to increase or decrease the level of adaptation. If the application is able to inform the client adaptation module the percentage of lost and damaged packets received, this module can send this information to the adaptation module in the gateway, which will decide whether or not it is recommended to increase or decrease the adaptation level for that application.

4. Implementation

The adaptation architecture defined and proposed is divided in two modules. The first one, which decides the adaptation to use and performs it, runs in the gateway. The second one, responsible for storing the user preferences and terminal profiles, and requesting services to the gateway, runs in the handheld device. A protocol is needed for the communication between the gateway and the handheld device modules.

4.1. Protocol

The traffic between the gateway and the handheld device modules may be of many kinds, and have many different receivers: command or info that the client sends to the gateway, request or info that the gateway sends to the client, or data that the application sends to the server or vice-versa.

In order to allow this information to be identified, a protocol is defined. The data sent is encapsulated into a packet, which allows the gateway and handheld device modules to know where that data should be delivered.

There are three kinds of packets: command packets, request and information packets, and data packets. Commands and data packets packets brings information about the application they are related, and its type. Request and information packets, on the other hand, are only for middleware control and do not need these information.

The sizes for the packets fields were defined taking into account two aspects: first, the packet header should be as compact as possible, adding as lower overhead as possible; second, the protocol should be expansible, allowing that, in the future, new commands, requests or information types are added to the middleware.

Command packets. The command packet format is despicted in Figure 2. Its first two bits are set to 0. They contain information about the application and its type.

The last six bits of the first byte indicate the type of the application. The knowledge of the application type is used to allow the middleware to perform a specific kind of adaptation for each kind of application. The second byte indicates the target application identification. Each application running on a handheld device has a unique identification. The last byte indicates the code of the command to be performed.

Request and information packets. The first two bits of request and information packets are set to 01. They are used by the middleware to exchange data between its two modules. The packet format is shown in Figure 3. The remaining six bits in the first byte indicate the type of request or information.

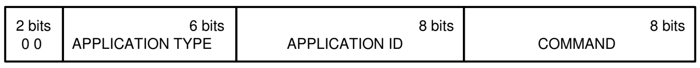


Figure 2: Command packets format

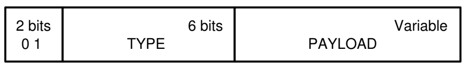


Figure 3: Request and info packets format

Requests are only sent by the gateway module. There is no payload in request packets.

Information may be sent either by the gateway or the client module. It may be sent upon request, or after some condition is reached. The information is sent in the payload, which has a different size depending on the information.

Data packets. Data packets are not related to the middleware module. They contain data the application and the server send to each other. In these packets, the first two bits are 10. The remaining six bits indicate the type of the application.

The next byte indicates the application identification. It indicates the application that sent the packet (in case it was sent to the server), or the application that should receive the packet (in case it was sent by the server).

The third byte indicates the size of the payload. It has up to 255 bytes, depending on the size value. This maximum payload size was defined due to the fact that a big packet could allocate resources for a long time when the network bandwidth is low, degrading the quality of other applications. The packet format is shown in Figure 4.

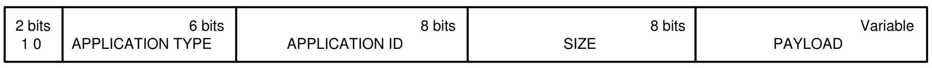


Figure 4: Data packets format

4.2. Gateway

The gateway module is the biggest piece of the adaptation architecture. It keeps track of every mobile device connected, its current bandwidth, and its utilized bandwidth. When an adaptation must be performed, the gateway module chooses the application to adapt and the adaptation to be performed, and performs the adaptation.

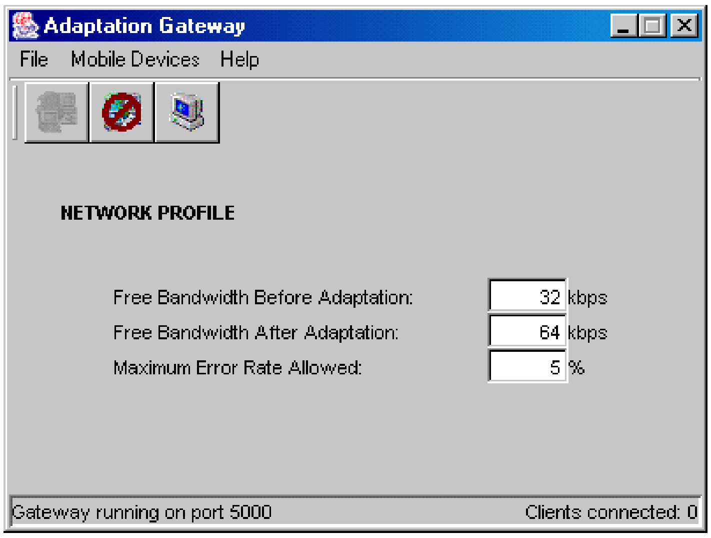
The gateway module was implemented using The Java 2 Standard Edition (J2SE) programming language. It is important to notice that any programming language could be used, since the gateway module runs in the fixed network with no battery power or computing capabilities restrictions. The Java programming language was choosen due to the existence of a similar technology for developing applications to mobile devices.

4.2.1. Interface

The gateway interface is used by the network operator to interact with the gateway. It allows the gateway to be started or stopped. The network operator can also set some parameters the gateway will use when performing an adaptation. Furthermore, it also allows the operator to look at the information for each mobile device connected to the gateway.

The main window, shown in Figure 5, allows the network operator to define the parameters for the network profile. Through this window, he can start or stop the gateway. He can also access the mobile device windows through the mobile devices menu.

In the mobile device window, shown in Figure 6, there is information about the current bearer, the current bandwidth, the utilized bandwidth, number of applications running, and bytes and packets sent to the client. In order to experiment the gateway, it is possible to modify the current bearer and the current bandwidth through this window.

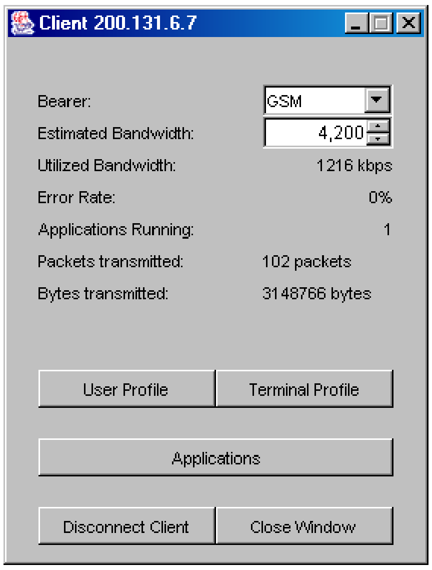


NETWORK PROFILE

Free Bandwidth Before Adaptation: 32 kbps
Free Bandwidth After Adaptation: 64 kbps
Maximum Error Rate Allowed: 5 %

Gateway running on port 5000 Clients connected: 0

Figure 5: Gateway interface



Bearer:	GSM
Estimated Bandwidth:	4,200
Utilized Bandwidth:	1216 kbps
Error Rate:	0%
Applications Running:	1
Packets transmitted:	102 packets
Bytes transmitted:	3148766 bytes

Figure 6: Mobile device window

The network operator can also look at the user profile, the terminal profile, get information about applications running, such as priority and adaptation level, or disconnect the client.

4.3. Handheld Device

The handheld device module plays an important role. It receives all connections from applications willing to access network services, and communicates with the gateway module. When receiving data from the network, it must identify to which application that data must go. The handheld device module is also responsible for keeping the terminal and user profile, and send them to the gateway module when requested.

It was decided to use the Sun's Java 2 Micro Edition (J2ME) technology to implement the mobile device module, since the gateway was already developed in Java. J2ME technology uses an application execution environment based on Java, and is designed for many types of devices, from pagers to set boxes. It defines a type of class called *midlet*, which can be executed in a handheld device. *Midlets* are the main classes when defining an interface with the user.

4.3.1. Interface

The user interface on the handheld device module was designed following the principles of usability. Upon starting the middleware, the user is requested to fill in the user profile. First, the user must indicate some information that will be used during adaptation. The screen is shown in Figure 7.

2 bits	6 bits	8 bits	8 bits
0 0	APPLICATION TYPE	APPLICATION ID	COMMAND

Figure 7: Filling the user profile: indicating parameters

The next step is filling in the audio, image, and video preferences for adaptation. This is done in screens with choices, where the user chooses his preferred technic first. After the user profile is complete, the user may start the middleware. It will run on background and the user will not have access to it anymore.

5. Evaluation

The implemented architecture itself does not perform adaptation. In orther to realize this task, it needs adaptation modules to be attached to it. Hence, a video adaptation module was implemented to evaluate the proposed architecture.

5.1. The Video Adaptation Module

The implemented video adaptation module sends the video file directly to the handheld device, without passing through the gateway module. However, it still must inform the gateway module about the utilized bandwidth.

It was decided to implement the video server at the same module as the video adaptation module. In that way, instead of requesting a video file to a video server, the video adaptation module actually opens and reads the file it wants to transmit to the handheld device.

A video was encoded in MPEG-1 format [Group, 1989], in different screen sizes and bit rates. When a new video application is launched in the handheld device, the gateway will start the video adaptation module, and verify the terminal profile to decide which adaptation techniques can be applied without modifying the user perception. Once this is done, this module opens the respective file, starts to send data to the mobile device through a UDP socket.

During its execution, the video adaptation module can be requested to perform an adaptation by gateway. Only an adaptation is performed at a time. If this adaptation is not enough to make the utilized bandwidth lower than the current value, another adaptation is requested. This guarantees that, when two applications with the same priority are running on the client, their quality will be increased or decreased at the same rate.

5.2. Performance Metrics

Utilizing an adaptation architecture and verifying if the received video has good quality or not would be the best way to evaluate it. However, defining whether the quality of a video is good or not is hard. Some users may consider it good, while others not. Due to this fact, it was necessary to define more technical performance metrics to evaluate the implemented adaptation architecture.

The number of transmitted bytes is an important issue that must be measured. The transmission of bytes cost money, and the network operator is not willing to send bytes that will not be delivered due to lack of bandwidth. Moreover, transmitting more bytes than the network can deliver means a bigger lost byte rate.

The lost byte rate is another important performance metric. When transmitting audio or video, a great loss produces a broken video, or a non intelligible audio. The adaptation architecture is expected to perform adaptation in order to decrease this percentage.

The adaptation architecture, however, adds overhead to the communication. The implemented middleware works on both the gateway and the mobile device. Therefore, some packets must be sent between them to control adaptation. It is important to measure the amount of control bytes related to the amount of data bytes to verify this overhead.

5.3. Experiments Description

Once the performance metrics were defined, some experiments should be performed to evaluate the implemented middleware. In those experiments, the available bandwidth would vary and the performance adaptation should be verified. However, the available bandwidth oscillation should be the same for all the experiments, otherwise those experiments results could not be compared. A scenario where bandwidth variation would occur and adaptation would be required was defined, and the experiments performed for that scene.

In order to simulate the network bandwidth, a buffer was implemented. This buffer accepts up to 640 kilobytes of data. Packets are removed from this buffer in the defined current network bandwidth rate. When packets are arriving in the buffer in a speed higher than the speed they are removed (the utilized bandwidth is higher than the current bandwidth), the buffer size grows. If it is full, packets that arrive will be discarded.

Three experiments were conducted. The same video was transmitted from the gateway to the handheld device. The available bandwidth variation followed the defined scenario.

Experiment 1. In the first experiment, the video was transmitted with no adaptation. The user received the file at his best quality. Figure 8 shows both the available and the utilized bandwidth. It is important to notice that, for the most part of the time, the utilized bandwidth was higher than the available bandwidth.

Figure 9 shows information about bytes sent by the gateway, and received by the handheld device. Without adaptation, the video server sent about 55.07 megabytes of data. However, only about 16.36 megabytes were received. The remaining data were lost. For each received byte, about 3.37 bytes were sent. There was no overhead, since the adaptation architecture was not active.

Experiment 2. In the second experiment, the adaptation architecture was working. The user defined his video preferences in the following order: packet discard up to 10%, video reduction, quality reduction, and color to black-and-white transformation. The network profile was defined in a way that the adaptation quality would only be increased after the free bandwidth were 64 kbps or over, and adaptations would try to leave at least 32 kbps of free bandwidth.

The available and utilized bandwidth during the video transmission are shown in Figure 10. The adaptation architecture guarantees that the utilized bandwidth will be lower than the available bandwidth, performing adaptation in the video when the available bandwidth diminishes. It also tries to increase the utilized bandwidth when the free bandwidth gets higher than 64 kbps.

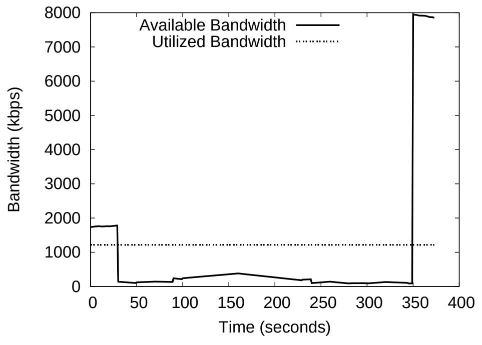


Figure 8: Available BW and utilized BW on first experiment

Gateway	
Bytes sent	57,753,501
Handheld Device	
Bytes received	17,159,892
Error rate (bytes)	70.29%

Figure 9: Bytes and packets sent and received in the first experiment, when no adaptation is performed

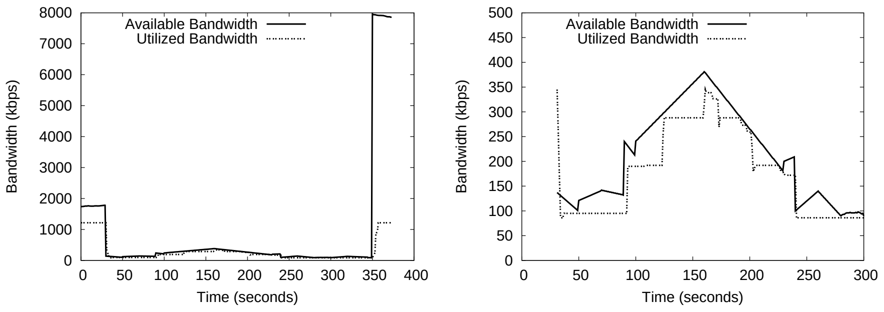


Figure 10: Available bandwidth and utilized bandwidth on second experiment

Figure 11 shows information about bytes sent by the gateway, and received by the handheld device. In this case, about 14.58 megabytes of data was transmitted. Only 1.59% of the sent bytes were not delivered. The overhead was of 40,289 bytes, or 0.26% of the transmitted data, which is small considering the gain in transmitted bytes and the smaller error rate.

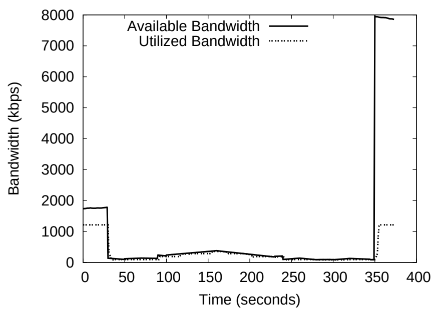
Gateway	
Bytes sent	15,287,436
Handheld Device	
Control bytes received	40,289
UDP bytes received	15,044,410
Error rate (bytes)	1.59%

Figure 11: Bytes and packets sent and received in the second experiment

Experiment 3. The third experiment is similar to the second one, except that the network profile now was set in a way that the adaptation quality would only be increased after the free bandwidth were 32 kbps or more, and adaptations would try to leave at least 16 kbps of free bandwidth.

The available and utilized bandwidths during the video transmission are shown in Figure 12. It is similar to the figure for the second experiment (Figure 10). However, more adaptations were performed, for the network profile in this experiment indicated that the adaptation architecture should try to increase the utilized bandwidth when the free bandwidth got higher than 32kbps. Figure 13 shows a comparison between the utilized bandwidth on both experiments.

The information about bytes sent and received are shown in Figure 14. In this case, since adaptation was performed more often, the number of bytes sent by the gateway (about 14.85 megabytes) were greater than in experiment 2. However, the overhead (47,385 bytes, or 0.30%) was



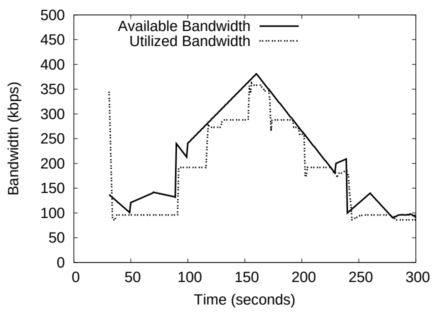


Figure 12: Available bandwidth and utilized bandwidth on third experiment

greater than in the previous experiment. Nevertheless, it is still a small number. The percentage of bytes not delivered was also small (1.51%).

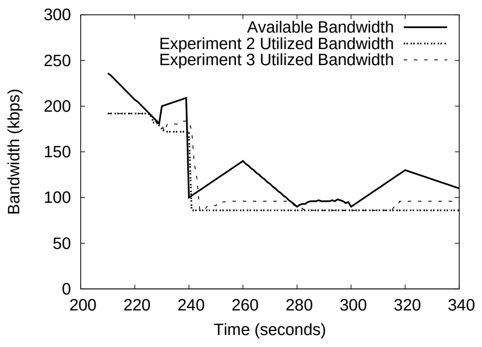


Figure 13: Comparison of utilized bandwidth between first and second experiments

Gateway	
Bytes sent	15,568,525
Handheld Device	
Control bytes received	47,385
UDP bytes received	15,333,178
Error rate (bytes)	1.51%

Figure 14: Bytes and packets sent and received in the third experiment

5.4. Results Analysis

The results obtained through the performed experiments showed the gains introduced by the use of the adaptation architecture. The error rate when no adaptation was performed was higher than 70%. In the experiments where the architecture was active, however, this error rate was lower than 2%, indicating a better quality of service.

The adaptation architecture also allowed the gateway to avoid sending more bytes than the network could deliver. In the experiment without adaptation, over 55 megabytes of data were sent by the gateway to the handheld device, although the network could only deliver 16.36 megabytes. In the experiments with adaptation, on the other hand, the quantity of data sent was lower than the maximum the network could deliver, avoiding the wasteful use of resources. Also, the results showed that the overhead introduced by the architecture is lower than 0.5%.

Finally, the experiments results also showed that the quantity of data sent by the gateway to the handheld device when the architecture is active gets closer to the maximum value the network can deliver with lower values for two network profile parameters: free bandwidth that should be left after adaptation, and minimum free bandwidth that should exist before the adaptation quality is increased. This makes sense since, with lower values, adaptations are performed more often. Thus, the utilized bandwidth is moved closer to the available bandwidth more often.

6. Concluding Remarks and Future Work

The experiments conducted through the evaluation of the implemented adaptation architecture showed that the use of this architecture allowed the gateway not to send more bytes than the network could support, adding inexpressive overhead data. Different values were used for the free bandwidth that should be left after adaptation, and for the minimum free bandwidth that should exist before the application quality were increased. It was possible to notice that the choice of lower values for those two parameters led to a number of bytes transmitted closer to the magnitude the network could support, without significantly increasing the error rate and the overhead. However, more quality and screen size changes were performed. When these changes are constant, the received video has a worse quality than if it was kept in the lower quality or screen size value. Hence, allowing the network operator to define these values in the network profile is an important point not present in other adaptation architectures. The network operator understands his network, and has the knowledge to decide which values would avoid constant application adaptation.

The implemented adaptation architecture needs adaptation modules to be added to it in order to perform adaptation. Throughout this work, only a video adaptation module was implemented. It would be interesting to implement adaptation modules for other data types, like audio and images, and evaluate the architecture for those data types as well. Moreover, it would also be attractive to evaluate the implemented architecture when adapting two or more data types at the same time.

It would also be useful to allow the network operator to define some rules the middleware should follow in certain circumstances. He is responsible for the network and has the knowledge to decide what should be done in specific situations. Some of these situations could be when a new application is loaded, the bearer changes, the battery power is lower, the bandwidth increases or decreases, or the error rate increases.

Finally, another interesting work would be verifying the gateway scalability. This could be done by connecting more clients to it at the same time and running applications on them, and verifying how it would affect the adaptation performed for the clients.

References

Bakken, D. E. (2001). *Encyclopedia of Distributed Computing*, chapter Middleware. Kluwer Academic Publishers.

Coulouris, G., Dollimore, J., and Kindberg, T. (1994). *Distributed Systems - Concepts and Design*. International Computer Science Series. Addison-Wesley Longman, Inc., 2nd edition.

Group, M. P. E. (1989). Mpeg homepage. http://www.chiariglione.org/mpeg/index.htm.

Harmer, J. (1996). The onthemove project. In *Proceedings of the 3rd International Workshop on Mobile Multimedia Communications*, Princeton, New Jersey, USA.

Kreller, B., Park, A. S.-B., Meggers, J., Forsgren, G., Kovacs, E., and Rosinus, M. (1998). Umts: A middleware architecture and mobile api approach. *IEEE Personal Communications*.

Mahony, D. O. (1998). Umts: The fusion of fixed and mobile networking. *IEEE Internet Computing*, 2(1):49–56.

Margaritidis, M. and Polyzos, G. C. (2000). Mobiweb: Enabling adaptive continuous media applications over 3g wireless links. In *IEEE International Conference on Third Generation Wireless Communications*, Silicon Valley, San Francisco, California.

Mascolo, C., Capra, L., and Emmerich, W. (2002). Mobile computing middleware. In *Networking 2002 Tutorials*, pages 20–58. Springer.

Tanenbaum, A. S. (1995). *Distributed Operating Systems*. Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632.