

# Enabling Multimedia Applications in Memory-Limited Mobile Devices

**Raul Fernandes Herbster**

*Federal University of Campina Grande, Brazil*

**Hygo Almeida**

*Federal University of Campina Grande, Brazil*

**Angelo Perkusich**

*Federal University of Campina Grande, Brazil*

**Marcos Morais**

*Federal University of Campina Grande, Brazil*

## INTRODUCTION

Embedded systems have several constraints which make the development of applications for such platforms a difficult task: memory, cost, power consumption, user interface, and much more. These characteristics restrict the variety of applications that can be developed for embedded systems. For example, storing and playing large videos with good resolution in a limited memory and processing power mobile device is not viable.

Usually, a client-server application is developed to share tasks: clients show results while servers process data. In such a context, another hard task for limited memory/processing devices could be delegated to the server: storage of large data. If the client needs data, it can be sent piece by piece from the server to the client.

In this article we propose a layered architecture that makes possible the visualization of large videos, and even other multimedia documents, in memory/processing limited devices. Storage of videos is performed at the server side, and the client plays the video without worrying about storage space in the device. Data available in the server is divided into small pieces of readable data for mobile devices, generally JPEG files. For example, when the client requests videos from the server, the videos are sent as JPEG files and shown at an ideal rate for users. The video frames are sent through a wireless connection.

The remainder of this article is organized as follows. We begin by describing background concepts on embedded systems and client-server applications, and then present our solution to enable multimedia applications in memory-limited mobile devices. We next discuss some future trends in mobile multimedia systems, and finally, present concluding remarks.

## BACKGROUND

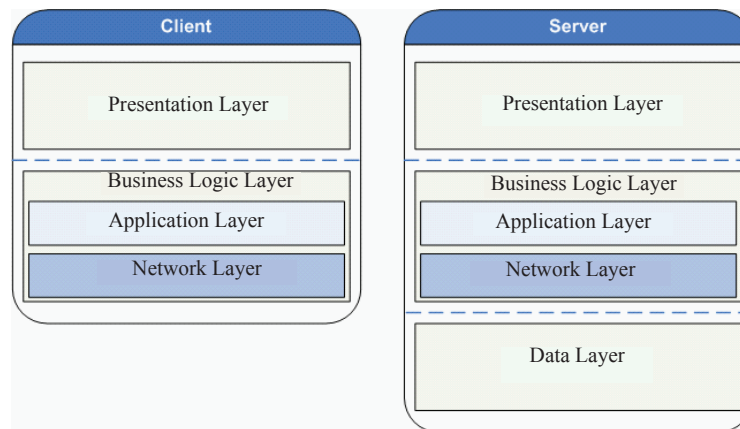
### Embedded Systems

An embedded system is not intended to be a general-purpose computer. It is a device designed to perform specific tasks, including a programmable computer. A considerable number of products use embedded systems in their design: automobiles, personal digital assistants, and even household appliances (Wayne, 2005). These limited systems have some constraints that must be carefully analyzed while designing the applications for them: size, time constraints, power consumption, memory usage and disposal, and much more (Yaghmour, 2003).

These constraints restrict the variety of software for embedded systems. The development of applications which demand a large amount of memory, for example, is not viable for embedded systems, because the memory of such devices is limited. Extra memory can also be provided, but the total cost of application is very high. Another example is multimedia applications, such as video players: storing and playing large videos with good resolution in a limited memory and processing power mobile device is a very hard task.

There are specific platforms that were developed to perform multimedia tasks: embedded video decoders and embedded digital cameras, for example. However, other considerable parts of embedded systems, like personal digital assistants (PDAs) and cell phones, are not designed to play videos with good quality, store large amount of data, and encode/decode videos. Thus, it is important to design solutions enabling multimedia environments in this variety of memory/processing-limited devices.

Figure 1. Client/server architecture



## Layered and Client-Server Architectures

Layered architectures share services through a hierarchical organization: each layer provides specific services to the layers above it and also acts as a client to the layer below (Shaw & Garlan, 1996). This characteristic increases the level of abstraction, allowing the partition of complex problems into a set of tasks easier to perform. Layered architectures also decouple modules of the software, so reuse is also more easily supported. As communication of layers is made through contracts specified as interfaces, the implementation of each module can be modified interchangeably (Bass & Kazman, 1998).

Most of the applications have three major layers with different functionalities: presentation, which handles inputs from devices and outputs to screen display; application or business logic, which has the main functionalities of the application; and data, which provides services for storing the data of the application (Fastie, 1999).

The client-server architecture has two elements that establish communication with each other: the front-end or client portion, which makes a service request to another program, called server; and the back-end or server portion, which provides service to the request. The client-server architecture allows an efficient way to interconnect programs that are distributed at different places (Jorwekar, 2005). However, the client-server architecture is more than just a separation of a user from a server computer (Fastie, 1999). Each portion has also its own modules: presentation, application, and data.

## ENABLING MULTIMEDIA APPLICATIONS

Multimedia applications demand a considerable amount of resources from the environment in order to guarantee quality

of service, which can be defined in terms of security, availability, or efficiency (Banâtre, 2001). Embedded systems have several constraints, like limited memory (Yaghmour, 2003), which make it very difficult to implement multimedia applications in an embedded platform.

Today, the growing interest for mobile devices and multimedia products requires the development of multimedia applications for embedded systems (Banâtre, 2001). There are approaches (Grun, Balasa, & Dutt, 1998; Leeman et al., 2005) that try to enhance embedded systems memory and other system aspects, such as processing, to provide better results in multimedia applications. However, most of the solutions available focus on hardware architecture, and a large number of programmers are not used to programming at the hardware level.

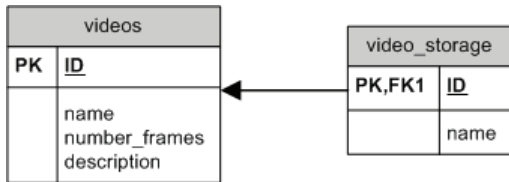
A solution based on client-server architecture is a good proposal for limited-memory/processing mobile devices because harder tasks can be performed by the server side whereas the client just displays results. By designing applications based on an architecture that shares tasks, constraints like limited memory and low computing power are partially solved. In this article, we propose a layered, client/server architecture that allows playing and storing large videos on limited-memory/processing mobile devices. The data is sent through a wireless intranet.

## Client/Server Architecture

In Figure 1, both client and server modules are illustrated. Each module of both elements can be changed at any time, except the application layer because the rules of application are defined on it: if business logic changes, so does the application.

The server architecture is a standard three-tier architecture:

Figure 2. Logic model of video descriptions information



- **Presentation Layer:** This layer interacts directly with the client. Its functionalities are related to display forms so that the user adds multimedia content to the server repository.
- **Business Logic Layer:** This has two sub-layers: the network layer, which manages the connection of server and client, receiving requests and sending responses; and the application layer, which requests services for the data layer, such as document storage and reports.
- **Data Layer:** This layer stores the multimedia documents as JPEG files. Each document has information about management files, including ID, number of frames (JPEG files), and specific description elements, which depends on the multimedia document type. Figure 2 illustrates the logic model of the tables that contain such information. The server stores in a table (video\_storage) the titles and ID of all videos. All the information about a video is stored in another table (videos).

The client is a single two-tier application; the data layer is not defined.

- **Presentation Layer:** This consists of a video player with buttons to select the video and a screen to display the video.

- **Business Logic Layer:** This has two sub-layers: the network layer, which manages the connection with the client, sending requests to the server and receiving data from it; and the application layer, which gets frames from the network layer and also controls the tax rate of displaying the frames.

### Execution Scenario

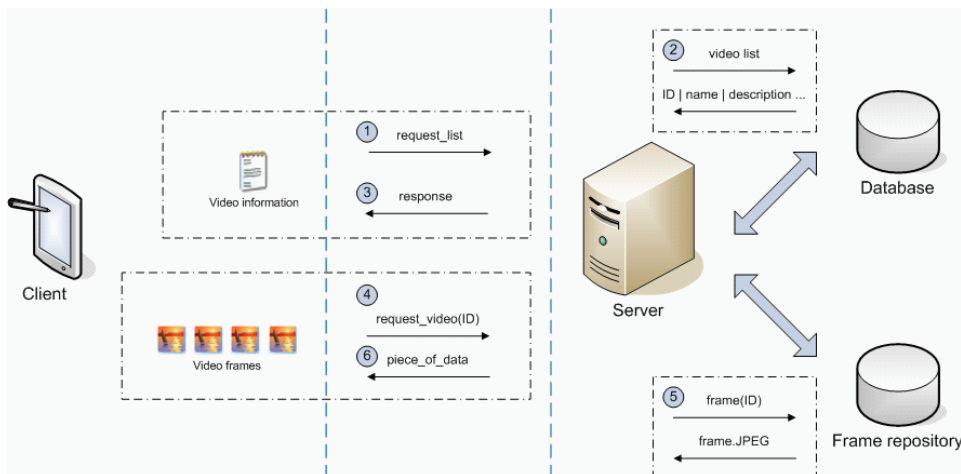
In what follows, we present an execution scenario of the mobile multimedia architecture. For this, consider that, at server side, a video was divided into small pieces of readable data (JPEG files). The quantity of pieces depends on the desired quality of the video that will be played at the client side.

The communication process between client and server is illustrated in Figure 3 and is described as follows. Whenever the server receives a connection request from a client, it sends to the client the list of available videos (steps 1 and 2). The client receives data and displays this information whenever required (step 3). Then, the client sends to the server the ID of the requested video (step 4). The server receives the request and starts the transmission of the video, piece by piece (steps 5 and 6).

At the client side, the pieces of data (JPEG files) are received in a tax rate that depends on the network (traffic, band, etc.). However, to display frames to the client in a constant tax rate and guarantee quality of service, it is necessary to maintain a buffer, which is controlled by the video player.

In the architecture described, the data is sent through a wireless intranet. The pieces of data are JPEG files, but could also be in Motion JPEG (MJPEG) format. The tax of frames depends on the quality of the video player on the client side: generally, a high video quality rate is 30 frames per second, whereas a low video quality rate is 10 frames per second.

Figure 3. Client/server communication



In a wireless network, this solution needs a large part of the network bandwidth. Therefore, there is a tradeoff between memory/processing capacity and network bandwidth. Nevertheless, considering home entertainment environments, such a tradeoff is worth the cost mainly because wireless networks in home environments have enough bandwidth to be used in such a context.

The architecture can also be used for other kinds of multimedia documents. For example, large PDF documents cannot be visualized with good quality on memory/processing-limited devices. The PDF documents can be also shared as JPEG files and sent to the client.

## **FUTURE TRENDS**

A protocol defines communication between client and server. It is an important element to guarantee QoS. As for future work, we suggest a deeper study of protocols enabling a good service for a given situation. It is important to focus on protocols that do not demand a lot of resources from the network, such as bandwidth.

There are some protocols that are implemented over UDP, for example, trivial file transport protocol (TFTP) (RFC 783, 1981) and Real-Time Transfer Protocol (RTP) (RFC, 1996). These protocols were implemented to demand few resources of the network, and to transfer a considerable number of files through the network.

Another interesting research approach is to measure variables of the network while using an application based on the architecture described, for example, to define how many devices running such applications the network supports.

## **CONCLUSION**

Multimedia applications demand a considerable amount of resources from systems. To develop multimedia applications for embedded systems, it is necessary to tackle constraints inherent to such platforms, such as limited memory and processing power.

In this article, we described a general architecture used for enabling multimedia applications in memory/processing systems. The architecture has two parts: a server, which receives requests and sends responses to clients; and clients, which make requests. Both parts have a layered architecture.

The solution proposed is relatively simple to implement and is easy to maintain, because the modules are decoupled and can be modified interchangeably. However, the architecture demands a considerable amount of network bandwidth, because the number of packages sent by the server to the client is large. Nevertheless, considering that the application is implemented over a wireless network in home environments, the bandwidth tradeoff is worth the cost.

## **REFERENCES**

Banâtre, M. (2001). Ubiquitous computing and embedded operating systems design. *ERCIM News*, (47).

Bass, C., & Kazman. (1998). *Software architecture in practice*. Boston: Addison Wesley Longman.

Fastie, W. (1999). Understanding client/server computing. *PC Magazine*, 229-230.

Grun, P., Balasa, F., & Dutt, N. (1998). Memory size estimation for multimedia applications. *International Conference on Hardware Software Codesign, Proceedings of the 6th International Workshop on Hardware/Software Codesign* (pp. 145-149), Seattle, WA.

Yahgmour, K. (2003). *Building embedded Linux systems*. CA: O'Reilly.

Jorwekar, S. (2005). *Client server software architecture*.

Leeman, M., Atienza, D., Deconinck, G., De Florio, V., Mendías, J.M., Ykman-Couvreur, C., Catthoor, F., & Lauwereins, R. (2005). Methodology for refinement and optimisation of dynamic memory management for embedded systems in multimedia applications. *Journal of VLSI Signal Processing*, 40(3), 383-396.

RFC 783. (1981). *The TFTP protocol (revision 2)*. Retrieved April 6, 2006, from <http://www.ietf.org/rfc/rfc0783.txt?number=0783>

RFC 1889. (1996). *RTP: A transport protocol for real-time applications*. Retrieved April 6, 2006, from <http://www.ietf.org/rfc/rfc1889.txt?number=1889>

Shaw, M., & Garlan, D. (1996). *Software architecture: Perspectives on an emerging discipline*. Englewood Cliffs, NJ: Prentice Hall.

Wolf, W. (2005). *Computer as components: Principles of embedded computing system design*. San Francisco: Morgan Kaufmann.

## **KEY TERMS**

***Please provide 8-10 Key Terms and definitions that apply to this paper.***