In: R. Steinmetz, L. C. Wolf (eds.), Proc. of IDMS'97 - European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (Lecture Notes in Computer Science, Vol. 1309), p.272-281. Springer-Verlag, 1997 (ISBN 3-54063-519-X).

An Approach for an Adaptive Visualization in a Mobile Environment

Luc Neumann[1] and Alberto B. Raposo[2]

[1] Computer Graphics Center (ZGDV e.V.)
Rundeturmstraße 6, D-64283 Darmstadt, Germany
neumann@zgdv.de
[2] DCA–FEEC–UNICAMP (State University of Campinas)
Caixa Postal 6101, 13083-970 Campinas, SP, Brazil
alberto@dca.fee.unicamp.br

Abstract. With the evolving availability of wireless communication services and of affordable mobile devices such as notebooks or Personal Digital Assistants, mobile computing is becoming widely accepted and applied. It is one further step towards the vision of information access for anyone, anytime, anywhere. Now, mobile multimedia applications are needed to make the vision more real. However, because of the narrow bandwidth of wireless wide-area networks and the limited resources of mobile devices in comparison to stationary systems, the handling of multimedia data faces severe problems. This leads to a need for effective solutions that enable the interactive handling with multimedia services even over a wireless link.
In this paper we present an approach to optimize the rendering process in terms of response time in a mobile environment that is composed of a mobile client and several stationary servers. We propose an architecture that adapts the rendering tasks to the available resource environment. The main idea is to use the knowledge about the application semantic data, the resource environment, and the user preferences to find a good trade-off between the limitations.
As a first experiment for the adaptation platform, we present a WWW rendering application using VRML 2.0 (Virtual Reality Modeling Language), which filters VRML scenes in order to render only parts selected by the user. It illustrates the handling of application semantic data that can be used to adapt the rendering process.

1 Introduction

The recent development in the number and quality of wireless communication services leads to an increasing interest in mobile computing. Therefore, it is one of the most rapidly growing areas of the computer industry. Mobile users equipped with portable computer such as Personal Digital Assistants (PDAs) or notebooks may access information or services anywhere and at anytime (e.g., tourist information about schedules, hotels or museums). The connection to a stationary server providing remote services and data can be established with a mobile phone or with communication facilities provided by the mobile device itself (like the Nokia 9000 Communicator). This scenario of a mobile application is illustrated in Figure 1.

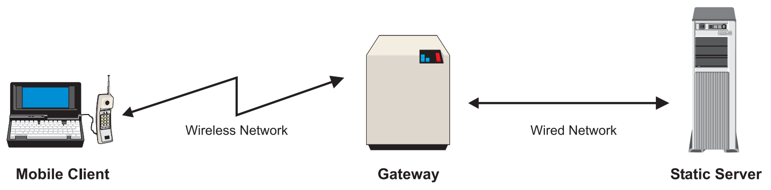

Fig. 1. Mobile Application Scenario.

Mobile users expect to handle all types of multimedia data with mobile devices as with desktop systems. The fulfillment of this requirement will strongly influence the acceptance of mobile computing technology. However, a mobile environment is more restricted than a stationary desktop environment. The restrictions are mainly caused by the limited resources of mobile devices and the narrow bandwidth of wireless Wide Area Networks (WANs). Therefore, it is a challenge to achieve a similar level of interactive multimedia data handling within mobile environments as in desktop environments.

Concepts for an optimized utilization of the available resources within a mobile environment are needed. One approach is to enable mobile applications to adapt to the current resource configuration. For example, to avoid overloaded mobile clients additional operations could be performed on resource-rich stationary servers (e.g. operations to convert the representation of a document). In this paper we propose an architecture that supports adaptation for a rendering application using metainformation such as information about the properties of resources and of data. Furthermore, a first experiment will be presented.

In the next section we will pinpoint the challenging problems associated to mobile computing and present approaches used to overcome them. After that, in Section 3 we will introduce an architecture distributing tasks among the mobile client and stationary servers. This architecture supports application adaptation applying resource dependent identification and delegation of subtasks to fulfill a task. The usage within a mobile rendering scenario and the possible delegation strategies of a rendering task are discussed. Then, the integration of the introduced architecture into the WWW is described. In Section 4 a first experiment for the identified architecture will be presented. It illustrates the usage of application semantic data to reduce the amount of data transferred and to reduce the utilization of the client resources. In Section 5 we come to the conclusions and highlight our future steps.

2 Mobile Computing

A mobile application accessing information in the WWW should be regarded as a distributed application, but this kind of application faces different challenges than an application using a wired stationary environment [2][10]. These new challenges are originated by the heterogeneous, limited and dynamic resources of the mobile environment.

The following properties characterize a mobile environment [7]:

1. Limited resources of the mobile devices in terms of storage, battery power, memory and processing power relative to non-portable devices.

2. Low bandwidth of wireless (WANs). Typical bandwidths of currently available cellular systems are 9.6 Kbps (GSM — Global System for Mobile Communication) or 19.2 Kbps (CDPD — Cellular Digital Packet Data).
3. High costs and low reliability (because of the frequent temporary disconnections) of wireless WANs.
4. Imbalance of resource availability between the mobile device and the stationary servers.
5. Users have no fixed location and can move during a connection.

In order to guarantee the success of highly interactive applications in a mobile environment, it is necessary that the requirements arising from the handling of non-textual bulky data (e.g., graphics, animation) can be fulfilled. This is one of the major challenges of mobile computing.

Comparing available data rates for wireless WANs with the required data throughputs for multimedia applications we can conclude that the available data rates are insufficient to fulfill the requirements of multimedia applications. Therefore, appropriate techniques (e.g., compression, progressive refinement, previewing) are needed for mobile environments. At lower levels of a mobile application Mobile IP [13], wireless TCP [1] and systems like ARTour [6] provide technologies to support mobility of a host [1] and to utilize the bandwidth efficiently (e.g., compressing protocol headers and data). At higher application–oriented layers, the following communication optimization techniques may be applied [15]:

1. *Data compression* increases the effective bandwidth compressing the information to be transmitted. Distillation [3] is an example of data specific compression developed in order to improve response times.
2. *Caching* uses local storage to reduce communication and to cope with voluntary and involuntary disconnections. Coda [11] is an example of a mobile file system that uses caching to reduce bandwidth requirements and to work disconnected. This technique, however, requires powerful storage resources in the mobile device.
3. *Prefetching* tries to anticipate data requests. This technique hides network latency, starting data transmission before the user requires it. However, if the prediction fails, communication and storage resources will be used unnecessarily.
4. *Data reduction* filters the information to be transmitted. This technique is normally used in the context of resource negotiation. Odyssey [11] is a system that uses this technique to provide "application-aware adaptation".

A successful solution should address the following mutually counterproductive aspects:

1. The transferred amount of data has to be as small as possible. This requires that parts of the application data should be processed and stored on the client side.
2. The use of local resources such as processing power and storage space should be the least possible. The client provides a presentation front end and communicates frequently with the server that process computing intensive parts of the application.

[1] The "Micro" mobility management such as handoff amongst wireless transceivers is solved by the wireless network system.

An appropriate trade-off between both aspects would overcome the most serious problems of a mobile environment, namely the narrow bandwidth and the client limited resources. In this paper we will introduce an architecture that supports adaptation at application level in regard to the available resource environment. It provides a good balance between the utilization of client resources and the required bandwidth. This allows to achieve a good quality of service within the limits of the available resources.

3 Adaptation Support Platform

In this section we present an architecture that adapts a rendering[2] application to the available resource configuration to optimize a rendering process within a mobile environment. The adaptation is achieved by the distribution of the rendering work (tasks) among the mobile client and the stationary servers using a proxy within the fixed network. The optimization will primarily focus on a fast response to the user even over a narrow bandwidth network. The main approach is to use information about the application semantic data (e.g., structure, table of contents) and the environment resources (e.g., available services) to distribute the tasks. Furthermore, user preferences (e.g., preferences between presentation quality and transfer time) will be used to control the mapping of the tasks.

The usage of a proxy between the client and the server to improve application performance is not new. They are used to reduce the bandwidth requirements [17], to split mobile applications (application partitioning) in a mobile and a static part [16], or to enable clients to negotiate the possible quality of a data representation [4]. The mobile file system Odyssey[11] provides the negotiation for resources without a proxy. However, this requires a specific server. Schill et al. [12] propose an architecture based on DCE with a station manager on each host and a domain manager to manage resource access in dynamic mobile environments. Here, the station manager mediates resource access. A key feature of our approach is a proxy using metainformation about available services within the fixed network and about the application data.

In the following, we introduce first the functional components of the adaptation architecture, discuss possible partitioning strategies for the rendering tasks, and present the establishment of the architecture within the WWW.

3.1 Defining the Scene

It is sure to say that a fixed separation of a mobile application in a client and a server component is feasible, but can overload the client or the server. If we think of the mobile application scenario mentioned above, where the mobile client retrieves some data from the stationary server and visualize it, then such a mobile application can roughly be separated into the following functional components.

1. Display service;
2. Application service (e.g., Rendering service);
3. Data management service.

[2] "Rendering" here means the generation of an image from a textual scene description of it.

This means that, for example, a rendering process is separated from the user interface and the data. This type of functional separation is already well known in the area of business applications.

A straightforward approach to map this application functionality on a multi–server environment can be to locate the display service on the client, the bulk of the application service on a server, and the data management service on a third server. This offers different partitioning possibilities, that range from a thin up to a thick client respective server. Furthermore, the display service can be further separated into two functional components. A presentation service which presents things to the user and an interaction service which receives user input. Keeping this all together, a mobile application can be defined as a composition of objects that work together as illustrated in Figure 2.

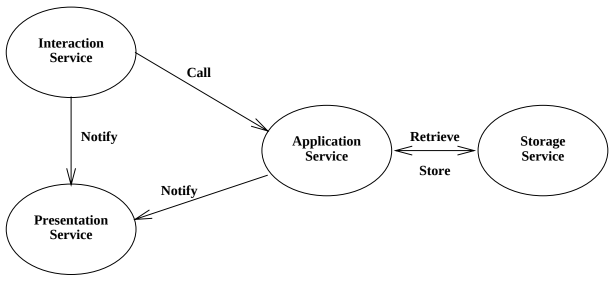

Fig. 2. Functional Decomposition of a Mobile Application.

This functional decomposition of a mobile application represents the basis for the distribution of the different components and their tasks among the mobile client and the available servers in the wired network. For the distribution of the tasks we propose an architecture with a so called *Resource and Task Manager* (*ResTaMan*). The role of *ResTaMan* can be compared with a mediator which is situated between consumer and provider processes. Within a rendering application *ResTaMan* distributes and controls the rendering process using metainformation. For example, it can use a filter which extracts relevant parts of a scene to distribute the rendering tasks. This is based on an additional description about the structure of the scene. Figure 3 shows the architecture with the *ResTaMan*. In principle, the *ResTaMan* receives a request including user preferences, client profile and the parameters for the operation from the mobile client. The resource information base contains the description about the resource configuration of the environment and the metainformation about the requested data. Based on this information the most suitable resources are identified and the tasks are distributed. The viability of this architecture is based on the assumption that a mobile application can be viewed as a collection of objects working together.

3.2 Approaches to Partition Rendering Tasks

In the previous section the functional components of a mobile application and the architecture to distribute tasks have been introduced. However, the decision about the distri-

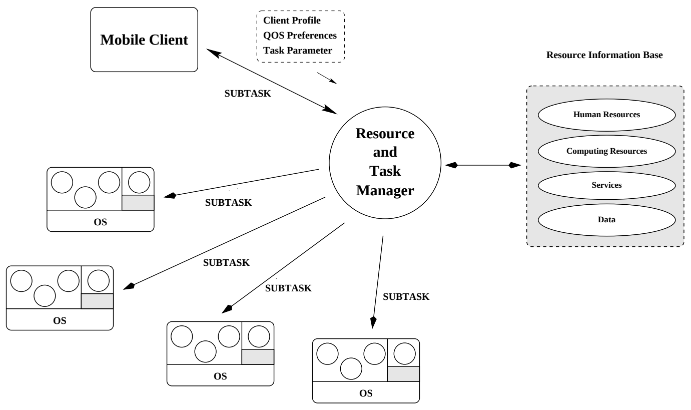

Fig. 3. Functional Components of the Adaptation Architecture.

bution of the tasks based on the application context is still an open issue. For the distribution we are focusing on the application service, that is the rendering service within a mobile rendering application. We are assuming that the presentation service to present images on the screen and the interaction service for the user input are available on the mobile client.

There exist a number of possibilities to partition the rendering work. The resulting task distribution intends to divide the work among several rendering objects to speed up the rendering process. In general, the partitioning of the rendering work can be related to the pixels (rendered image) or the scene content. In the following we introduce two possible partitioning strategies.

Quality Level. One can define different quality of service levels (QoS) that vary in the amount of work items for a rendering task. Then, the entire scene can be rendered in parallel with different qualities from several renderers. The different qualities can refer to:

1. Shading method (wireframe, flat, gouraud, phong, etc.);
2. Texture mapping;
3. Resolution of the image;
4. Level of Detail.

Scene Partitioning. In order to reduce the amount of work items for a subtask, the entire scene will be subdivided in smaller parts. These subparts are rendered in parallel and the results are merged. One obvious approach is to subdivide the pixels among the renderers. Then each of the involved renderer is responsible for a specific subarea of the entire image. If we are using the content of a scene, then we can identify following partitioning strategies.

1. Subparts of a static scene
   The scene is separated into different subscenes. Each subscene contains the objects

of a specific part. For example a scene can be separated into some background subscenes describing the surrounding panorama and one foreground scene containing the objects local to the viewpoint. This allows to render realistic panorama images on high end renderer in the fixed network and the foreground objects can locally be rendered on the mobile client. Furthermore, priorities could be assigned to different objects in the world. Then high level objects are essential to the scene and low level objects could be omitted.
2. Subparts of an animation.
The sequence of an animation can be subdivided into N subintervals. They are used by a pool of renderer in the wired network, to generate the subintervals of the animation in parallel. The generated image sequences are combined to a complete video stream and sent to the client. Certainly, one needs a video compression scheme to transmit the sequences. However, the latency time will be reduced even by high quality image sequences.

However, to enable the partitioning of rendering tasks additional knowledge about the structure of the scene is required. Within such a scene description the objects may have names, which depend on their role in the scene, or they are assigned to a specific quality level.

3.3 Establishment of an Adaptive Rendering Facility in the WWW Environment

In order to establish the introduced architecture in the WWW environment, we propose a solution that is based on two main points: the introduction of a semantic content header for the scene description, and the application of Java based Object Request Brokers (ORB)[3]. The ORB allows to integrate the adaptive architecture into the WWW. The Virtual Reality Modeling Language (VRML) [14], which is a file format describing interactive 3D objects and Worlds in the WWW, is used for the scene descriptions. The entire pipeline of the rendering architecture will encompass at least

1. a presentation facility (e.g., located on the mobile client),
2. different renderers on the mobile client and on stationary servers,
3. VRML–analyzer and distribution manager to define and distribute the different tasks, and
4. a composing and synchronization tool for the presentation.

The VRML–analyzer reads the semantic header and the VRML description. The header contains additional information about the content of the scene. It can be regarded as metainformation supporting the definition of tasks to be distributed. Together with an ODP-Trader, which is a yellow page service of the available resources, a good utilization of the rendering resources on the mobile client and in the fixed network can be achieved. Finally, an image composer and synchronizer are necessary to display the result. Figure 4 shows a separation of a rendering process into foreground and background scenes to illustrate the dataflow between the different components. In this case the metainformation describes the foreground and background of the scene.

[3] An ORB represents an object bus enabling objects to receive or send requests to local or remote objects.

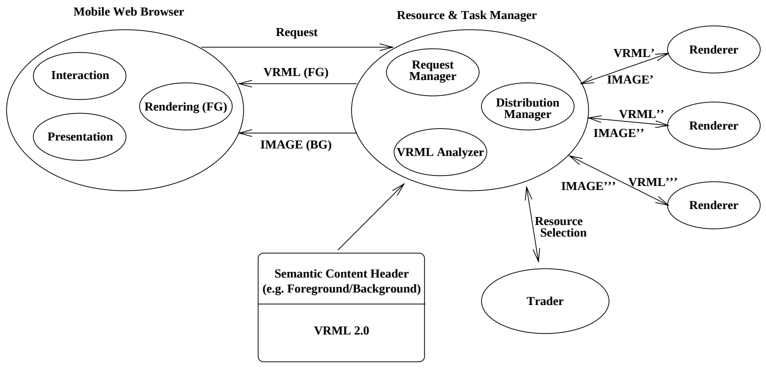

Fig. 4. Distribution of VRML Rendering.

The task for the foreground rendering is done on the mobile client and the background rendering is performed on one ore several renderers on the fixed network. The Request Manager receives the requests from the mobile client, and invokes the VRML analyzer and distribution manager. The distribution manager delegates and schedules the identified subtasks. It can be seen as a master component, that instructs various resources as workers to perform a subtask.

The usage of Java ORBs enables the Web Browser to gain access to arbitrary CORBA–based services. This allows the direct integration of computational services into the WWW. A Java applet can be downloaded to the Web client and serves as CORBA Client accessing services using the Internet Inter–ORB Protocol (IIOP). In our approach it contacts the *ResTaMan* object that represents a meta server from the viewpoint of the client. The entire rendering application consists of a set of interworking objects playing together through the ORB. For the transmission of images or image sequences we will use a stream oriented connection between the Applet and *ResTaMan* that may be established with an IP socket connection. Thus, for the bulk data transfer the stream connection is used and the control operations are transmitted by the IIOP. Figure 5 illustrates the integration of the proposed architecture within the WWW.

4 Object Extraction Method from VRML Scenes

As a first experiment for the adaptation platform presented in the previous section, we developed an application capable of selecting the elements (i.e., geometric objects, light sources, and cameras) of a remote VRML 2.0 world [9]. This application uses the *data reduction* technique to filter the data to be rendered. By presenting only the elements selected, the utilization of resources is reduced and the visualization can be done more efficiently. This filtering technique can be used to extract the relevant parts of a scene for the different renderers, as discussed in Section 3.

Figure 6 shows the strategy used in this application. It is executed in the client as a Java applet [5], downloaded with an HTML page.

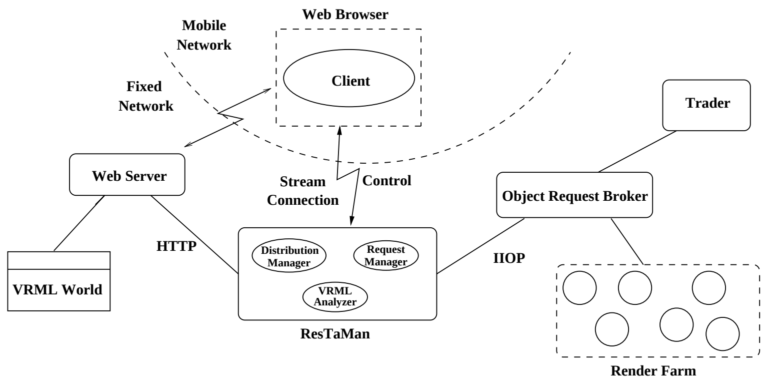

Fig. 5. Integration of the adaptive rendering architecture in the WWW.

The process begins with the establishment of a connection between the client and the application server. Then the URL of a VRML 2.0 world is send to the application server (arrow 1 in Figure 6). The application server will use Java resources to connect the server of the VRML file, read and parse it (arrow 2 in Figure 6). The first output of the application server is a small scene graph document, representing the hierarchy of the elements of the VRML world, which will be sent to the client (arrow 3).

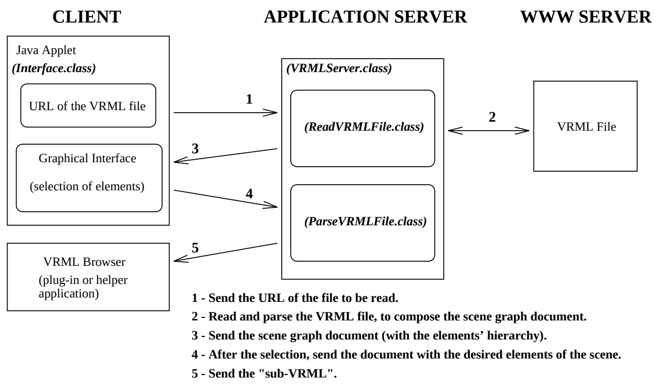
CLIENT
APPLICATION SERVER
WWW SERVER
Java Applet
(Interface.class)
URL of the VRML file
Graphical Interface
(selection of elements)
(VRMLServer.class)
(ReadVRMLFile.class)
(ParseVRMLFile.class)
VRML File
VRML Browser
(plug-in or helper application)
1
2
3
4
5
1 - Send the URL of the file to be read.
2 - Read and parse the VRML file, to compose the scene graph document.
3 - Send the scene graph document (with the elements' hierarchy).
4 - After the selection, send the document with the desired elements of the scene.
5 - Send the "sub-VRML".

Fig. 6. Strategy to visualize only a subset of a VRML 2.0 world.

Based on the received document, the applet creates an interface adapted to the hierarchical structure of the VRML world. This interface allows the user to select the desired elements of the world. After this selection, a new document is sent to the application server, describing the selected elements (arrow 4 in Figure 6). Using this new document, the application server can create a new VRML 2.0 world from the original one, by extracting only the desired parts of it. This final "sub-VRML" world is then sent to the client (arrow 5), that visualizes the results using any VRML browser.

5 Conclusions and Future

With the development of enabling technologies such as wireless networking and various types of affordable mobile devices, mobile computing has begun to be widely accepted. This leads to a need for applications capable of interactively accessing, manipulating, and visualizing multimedia data in a mobile environment. They are mainly confronted with problems caused by the limited resources on the mobile devices and the narrow bandwidth of the wireless link.

In this paper we have introduced an architecture aiming to improve the visualization in a mobile environment. Furthermore the integration of such an architecture within the WWW has been presented. We defined a general architecture that supports the adaptation of a mobile application in regard to the available resource configuration. This will avoid overloaded mobile clients and allows to achieve the best quality of service within the limits of the system. The main approach is to use application semantic information, information about environment resources and user preferences to divide the tasks to be done.

In order to use this architecture for a mobile rendering architecture we presented different possible partitioning strategies which are related to the quality or the content of a scene. It depends on the goal of the user, which strategy might be used. However, to identify different quality levels or specific subparts of the entire scene additional information about the content of the scene is required. Therefore, as a first step, we have presented an application that is able to select parts of a VRML world and to create sub-VRML worlds. This sub-VRML worlds could be rendered on the mobile client and the stationary server. It demonstrates how application semantic information can be used to identify and handle different subparts of a VRML world. Currently, the selection is done by the user who should have some knowledge about VRML, but our plan is to use semantic headers which automize the extraction process. This header can describe different quality levels, give names to subscenes or assign priority levels to different objects.

At the moment we have achieved a simple adaptation level, that allows to render only parts of interest or up to a specific priority level on the mobile client. Due to the reduced resource requirements this improves the visualization process.

Obviously many issues are still remain open. Currently, implementations of further components to validate the proposed architecture are in progress. This mainly address the Resource Information Base and the task–oriented interface for ResTaMan, allowing to include preferences within a request.

Acknowledgements

This paper is a result of ProBrAl 002/94, a cooperation project between UNICAMP (State University of Campinas) and the Technical University of Darmstadt, sponsored by CAPES (Brazil) and DAAD (Germany). We deeply appreciate the support granted by these institutions. We would like to thank FAPESP and CNPq (Brazil) for the sponsorship of some authors. Special thanks to Dr. Léo P. Magalhães, Dr. Ivan L. M. Ricarte, José M. De Martino, Dr. Rüdiger Strack and Jian Zhang for helpful pointers and clarifying discussions.

References

1. A. Bakre and B.R. Badrinath. I–TCP: Indirect TCP for Mobile Hosts. In *Proc. 15th International Conference on Distributed Computing Systems (ICDCS)*, pages 136–143, Vancouver, British Columbia, May 1995.
2. G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(4):38–47, April 1994.
3. A. Fox and E. A. Brewer. Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation. In *Fifth International World-Wide Web Conference*, Paris, France, May 1996. (http://www5conf.inria.fr/fich_html/papers/P48/Overview.html).
4. A. Fox, S. D. Gribble, E. A. Brewer, and E. Ami. Adapting to Network and Client Variation via On-Demand, Dynamic Distillation. *In Proc. ASPLOS-VII*, October 1996.
5. J. Gosling, B. Joy, and G. Steele. *The Java Language Specification – Version 1.0*. Java Series. Addison–Wesley, 1996. (http://java.sun.com/doc/language_specification/index.html).
6. IBM, WT Mobile Data Communication – ARTe, Heidelberg, Germany. *IBM ARTour – Technical Overview*, 1995.
7. L. Neumann, J. Zhang, and R. Strack. Evaluation of the Existing and Forthcoming Mobile Infrastructure. MOMID Deliverable 2, Computer Graphics Center, March 1996.
8. Object Management Group (OMG). CORBAservices: Common object services specification. Revised Edition 95-3-31, March 1995.
9. A. B. Raposo, L. Neumann, L. P. Magalhães, and I. L. M. Ricarte. Efficient Visualization in a Mobile WWW Environment. Technical Report DCA 001/97, DCA – FECC – UNICAMP, 1997. (ftp://ftp.dca.fee.unicamp.br/pub/docs/techrep/1997/index.html).
10. M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *15th. ACM Symposium on Principles of Distributed Computing*, Philadelphia, PA, May 1996. (http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-coda.html).
11. M. Satyanarayanan. Mobile Information Access. *IEEE Personal Communications*, 3(1), February 1996. (http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-coda.html).
12. A. Schill, B. Bellmann, W. Böhmak, and S Kümmel. System Support for Mobile Distributed Applications . *IEEE Workshop on Services in Distributed and Networked Environments (SDNE)*, pages 124–131, June 1995.
13. J. Solomon. Applicability Statement for IP Mobility Support. RFC 2005, Internet Network Working Group, October 1996.
14. *The Virtual Reality Modeling Language Specification – Version 2.0, ISO/IEC DIS 14772–1*, April 1997. (http://vrml.sgi.com/moving-worlds).
15. T. Watson. Application Design for Wireless Computing. In *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, December 1994. (http://snapple.cs.washington.edu:600/wit/presentations.html).
16. T. Watson, B. Bershad, and H. Levy. Using application data semantics to guide system network policies. In *SOSP'95 WIP Session*, 1995.
17. B. Zenel and D. Duchamp. Intelligent Communication Filtering for Limited Bandwidth Environments. In *Proc. 5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, Washington, U.S., May 1995. IEE Computer Society Press.