Media Distribution in a Pervasive Computing Environment

Winfried A.H. Berkvens, Arjan Claassen, Joep P. van Gassel, Alexander Sinitsyn
Philips Research
{winfried.berkvens, arjan.claassen, joep.van.gassel, alexander.sinitsyn}@philips.com

Abstract

Distribution of media in the fast growing world of digital stored content and multimedia supporting devices with connectivity, calls for a new media distribution architecture. The user should be provided with the experience of having an overview of his full media collection, regardless of the time, the place, and the connectivity. The architecture presented in this paper, fulfils these needs and can cooperate furthermore with non-compliant devices.

1. Introduction

We believe that in the near future our homes will have distributed networks of intelligent devices that provide us with information, communication, and entertainment. Furthermore, these systems will adapt themselves to the user and even anticipate on users need. This concept, called *Ambient Intelligence (AmI)* [1], calls for attention of researchers. Noteworthy features of this new concept are ubiquitous computing, natural interaction, and intelligence. Recent developments in technology, the Internet, the consumer electronics market, and social developments indicate that this concept might become reality soon.

The *Connected Planet (CP)* [2], as a first step towards realization of AmI, is an environment that puts people at the center of technology offering greater control, convenience, freedom, and productivity. It is an environment in which devices and appliances seamlessly communicate with one another and the outside world, to enhance people's daily activities. It is becoming a reality because of the available technologies (e.g. low-cost wireless connectivity, broadband Internet, high-capacity storage, etc.) and what consumers are beginning to demand (easy access to content and to other people – at an affordable price and from many devices). The CP will be enabled by an integrated network of displays, media storage and connectivity devices. They will cooperate to allow users to access, share and store media; communicate better; control and monitor security, lighting, utilities and services from anywhere within the home or the world (via mobile devices and the Internet).

Besides, the next generation consumer electronics devices, the Personal infotainment Companion (PiC) is expecting to play a crucial role in this CP vision.

In our vision the PiC is a small personal mobile device that stores and processes digital media (audio, video, still pictures and other data), for use both at home and away. The device contains a large-capacity embedded storage and has advanced (wireless and wired) networking capabilities.

This paper highlights challenges in the media distribution area that need to be taken into account when developing middleware for the CP environment. It presents an architecture for media distribution in next generation of consumer electronics products.

2. Problem description

The proliferation in wireless networking technologies is enabling a new class of applications that allows users to access their personal data anytime and anywhere.

Developing these types of applications, however, presents challenging problems to designers. Mobile devices, like the PiC, face temporary and unannounced loss of network connectivity when they move. They have usually short connection sessions. And they need to discover surrounding devices in an ad-hoc manner. Furthermore, they have scarce resources like limited battery power, processing power and memory.

In order to cope with these limitations, many research efforts have focused on designing new middleware capable of supporting the requirements imposed by mobility. As a result of these efforts, a number of middleware systems has been produced (e.g. Sun J2ME [5], Microsoft .NET Compact [6], UPnP [7], Jini [8]). Some of these middleware systems are targeted at support for disconnected operations and data-sharing (Coda [9], Odyssey [10], Bayou [11], Xmiddle [12]). However, many issues still require research. The major one is how to provide the user with the experience of having all media he needs available at any time, in any place, regardless of connection availability in the heterogeneous environment.

Dagstuhl Seminar Proceedings 05421
Data Always and Everywhere - Management of Mobile, Ubiquitous, Pervasive, and Sensor Data
http://drops.dagstuhl.de/opus/volltexte/2006/762

3. Media distribution architecture

Scenario-based architecting [3] is the process that has been followed to come up with an architecture for media distribution. As starting point for the process a number of user scenarios has been derived from different projects in our lab and some important ones have been identified by our market experts. Based on the selected user scenarios, the requirements have been identified and an architecture has been developed.

3.1. User scenarios

A set of scenarios is used as the basis of the process. The most relevant ones for this paper, covering most of the functionality of the system, are described here:

"John and Rob meet after work in a pub. Rob would like to have the photos of last week's party. John takes his PiC and drags these photos to Rob's device, represented on Johns display. Rob directly sees the added new photos on his device even before the content is completely transferred."

"John is on a holiday trip. During the flight he starts organizing his music collection on his home server via his PiC without being connected. Returning home, the PiC propagates the changes to the home server, which carries through these changes."

"John is transferring a home movie from Rob's PiC. Rob has to leave, causing the movie transfer to be interrupted. When John meets Rob and Harry the next day, the movie transfer continues automatically. Because Harry's PiC also contains the home movie, John's PiC recognized the availability and starts downloading in parallel from Harry's device."

3.2. Requirements

From these user scenarios, requirements have been extracted.

One of the requirements is to support the user in handling large sets of digital assets. Digital assets are pieces of electronically stored information valuable for the end user. In general, a digital asset consists of two parts: content (e.g. music, photo, video) and metadata describing this content (e.g. creator, title, album). This paper concentrates only on multimedia-related digital assets. Other types, like bank account and agenda information, are excluded.

In order to offer the user yet bigger freedom to manage his digital collections the architecture needs to support both on-line and off-line digital asset exchange and synchronization providing the user with views of non connected devices/collections. So, the user should be able to manage assets, which may be currently unavailable due to device disconnection, as if they are available. Apart from that, changes made to the organization of assets should be reflected in the views immediately. Furthermore, the device connection and disconnection processes should be transparent for the user.

Content transfer should be possible between all types of devices such as, stationary and portable devices, large and small size devices, devices implementing the architecture (e.g. home media center, PiC) but also legacy and peripheral devices (e.g. photo camera, MP3 player) that do not support it.

In order to perform efficient, scalable and robust information management across multiple heterogeneous data sources our middleware system needs an abstraction layer from both underlying storage technologies (e.g. file systems, databases) and communication protocols. The system also should be robust, interruptible, priority-based (i.e. the most valuable asset is served first), and resource-aware.

To summarize, the media distribution system should support at least the following functions:

- Large asset collection management;
- Both online and off-line asset exchange and synchronization;
- Separation of metadata transfers from content transfers;
- Connectivity via multiple communication protocols with various types of devices from different vendors;
- Robust, multi-target, priority-based, resource-aware content transfer management;
- Configurable by the user;
- Minimal interaction with the user during asset exchange.

3.3. Architecture

Based on the requirements, as described in the previous paragraph, an architecture for media distribution can be defined. This high level architecture, depicted in Figure 1, shows that the media distribution system interacts with applications on one side and with an abstraction of a device on the other side. Typical applications that can be thought of are for instance a browser application and a synchronization application. The device abstraction provides operating system functionality, storage, and connectivity.

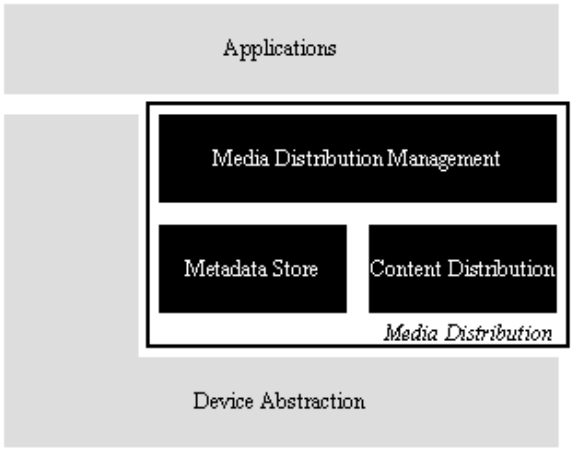

Figure 1. High level architecture for media distribution

This media distribution system provides means for transparent and consistent management of assets. The system separates the responsibilities for the handling of metadata from the handling of the content. Reason to do this, will become clear in the remainder of this section.

The next paragraphs discuss the three components of this system in more detail.

3.3.1. Media Distribution Management. The media distribution management component offers the single point of abstraction for the applications running on the local device. All distribution related operations performed by applications on assets, like queries, copies, moves, and deletions, are handled by this component. Where necessary this component creates operations for the underlying media store or content distribution components.

By providing this middleware abstraction, firstly, all applications can be kept unaware of where operations on metadata are routed and they do not necessarily need to know whether these operations are handled by a local metadata store or by a remote metadata store. At the moment an application requests for metadata, the media distribution management component addresses the convenient metadata store. So, a metadata request for a remote device can be forwarded to the remote device if it is present. If a representation of the metadata database, a so-called snapshot, of a remote device is available on the local device, such request can also be executed on this snapshot. This, because the access time to the local snapshot is expected to be shorter than accessing the remote device. When the remote device is not present, but a snapshot is available, a request can in this case also be executed on the snapshot. The decision on where the request will be handled is made by the media distribution management component. How the snapshot is kept up to date is discussed in Section 3.3.2.

Secondly, all operations on assets are interpreted by the media distribution management component, which has the capability to decide whether an operation results in a request for metadata change, a request for content transfer, or a request for both or none of them. Operations requested by applications based on user actions performed on assets, provide the inputs for deciding whether changes in the content distribution are required. If changes in the content distribution are required, a content distribution component is informed. This is done by means of so-called content distribution actions of which the following types are currently supported: copy content, move content, and delete content. Which content distribution component is informed, depends on the types of devices involved. If the destination device is compliant with the proposed architecture, the content distribution subsystem in this compliant device will control the transfer. If the destination device is a non-compliant device, e.g. legacy or peripheral device, the content transfer is controlled by the compliant source device. The reason for controlling the actions primarily from the destination device is to enable the destination device to have control over its own resources and to allow it to initiate the transfer of content from multiple sources simultaneously.

3.3.2. Metadata Store. The metadata store component deals with storing of metadata and making this metadata available to local or remote applications via the media distribution management component.

One option to store this metadata is to tag each content file of an asset with its metadata (e.g. ID3-tag [4]) and query this metadata when requested by an application. Because this is very time consuming, due to the need to read these tags from the files, it has been chosen to separate metadata from the content. A Database Management System (DBMS) is being used for storing the metadata. Using a database furthermore allows the system to perform more complex queries.

On top of the DBMS a metadata abstraction layer is included, to provide access to the metadata via an asset oriented interface. Furthermore, the abstraction makes it possible to implement whatever DBMS underneath without the need to change the interface.

Allowing that the content storage is not mapped one-to-one on the metadata organization, enables that a user can be presented with a user view that does not match the physical organization of the content. So, such a user view can even be provided if the content is not available on the device. Furthermore, assets need to be stored only once on a device but could be available at the same time in multiple places of a view. It even helps in providing the user with a direct response on organizational changes, giving the user feedback that an action is understood by the system, but allowing that the content redistribution which is possibly required is performed at a more convenient time.

Because content independent metadata storage is used, it is easy to include the use of snapshots in the system. By using snapshots, the information about assets of a device that is currently not available is still available to the user. This makes it possible to perform actions on these assets even when the device is not available. To make sure that the changes performed during an offline period are carried through on the original metadata database, the snapshots needs to be synchronized with the original database.

3.3.3. Content Distribution. The content distribution component takes care of handling the distribution of the content that belongs to the digital assets. It performs operations on content, based on the content distribution actions obtained from the media distribution management component. The different modules that are part of this component are shown in Figure 2.

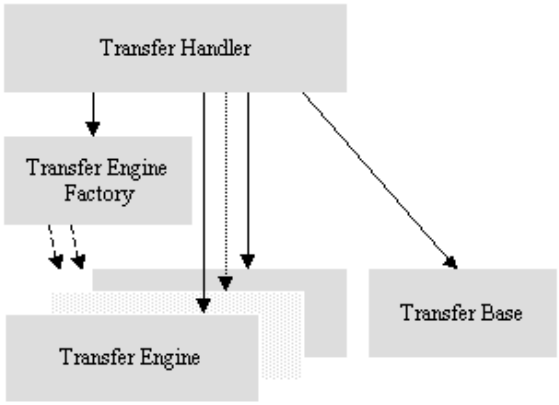

Figure 2. Detailed architecture of the content distribution component

Each content distribution action received is stored in a transfer base, which is a pool of pending content distribution actions, and is carried out by the content distribution component at an appropriate time. Whether it is the appropriate time is decided by the transfer handler based on a number of conditions.

The first condition that is used, is the availability of the device from which the content has to be retrieved and possible other resource constraints. The availability is checked via information stored in a device list manager, which receives this information via some service discovery mechanism (for instance UPnP Service Discovery). If the source device is accessible and sufficient resources are free a second condition, the priority of an action, can be used to select the appropriate content distribution action. The priority can be set based on the importance of the action which can depend on the (type of) application that generated the operation resulting in the content distribution action. For instance, it could be chosen that a user performed operation will result in a higher priority action than an automatically generated synchronization action.

At the moment the action is selected, the transfer engine factory is requested to create a suitable transfer engine. This engine executes the actual requested content distribution. If the content can be transferred from multiple sources simultaneously, for each source device a transfer engine is created which transfers a specified segment of the content. When the transfer is finished, or when the transfer is stopped due to disconnection of one of the devices involved, the metadata for this asset is updated in the DBMS. In case of a transfer interruption, storing the transfer progress information makes it possible to resume the transfer later-on from the point where it was interrupted. Furthermore, by doing this, the user can be provided with feedback about the availability level of the content.

4. Implementation

The presented architecture was validated through a working prototype. The demonstrator code was written in C++ and runs under Linux on PC and Transmeta Crusoe platforms.

The metadata store provides an object-oriented interface implemented on top of the relational database system MySQL [13]. Device and service discovery as well as remote action invocation was implemented using the UPnP framework [7]. The media distribution management component was build and positioned as an UPnP CDS extension service. To support media distribution with peripheral devices a device storage abstraction layer was build on top of USB Mass Storage [14], FireWire [15] and Bluetooth [16]. A transfer engine supporting HTTP based content transfers was build.

5. Future work

A basic architecture for media distribution, complying with the given requirements, has been presented. This architecture can however be extended with extra functionalities that will enhance the user's capabilities in handling large sets of digital assets (e.g. [17]). User interaction models which improve the experience will provide other inputs for future work.

To increase flexibility for programmers and enabling them to tune the proposed architecture the following policies have been envisioned: interface selection if a connected device has more than one; picking a suitable network technology and matching transfer protocol; scheduling and deletion of transfer actions in the Transfer Base.

The proposed architecture does not explicitly deals with assets protected by digital rights management. The handles required for this will be added to the current

architecture. Eventually, the core parts of the architecture can be formalized and standardized, thereby enabling it to be added to future middleware systems.

6. Conclusions

Transparent distributed data management is crucial to Ambient Intelligent applications. The proposed media distribution architecture offers a possible solution. It provides the user with the experience of having all his media collections available at any time, in any place, and managing them regardless of connection availability in the heterogeneous environment. This experience is enabled in our system by the separation of metadata and content handling. Other features are efficient handling of snapshots, usage of various database technologies, and leveraging device and service discovery mechanisms.

7. References

[1] Philips Ambient Intelligence vision, http://www.philips.com/research/ami

[2] Philips Connected Planet vision, http://www.philips.com/connectedplanet/

[3] H. Obbink, P. America. Towards Evergreen Architectures: On the usage of scenarios in system architecting. In *Proceedings of the Int. Conference on Software Maintenance,* pages 298-303, September 2003.

[4] ID3 informal standards, http://www.id3.org/, 2003.

[5] Sun Microsystems, Inc. Java Micro Edition. http://java.sun.com/products/j2me/, 2001.

[6] Microsoft. .NET Compact Framework. http://msdn.microsoft.com/vstudio/device/compactfx.asp, 2002.

[7] UPnP Forum. Universal Plug and Play. http://www.upnp.org, 1998.

[8] K. Arnold, B. O'Sullivan, R.W. Scheifler, J. Waldo, and A. Wollrath. *The Jini[tm] Specification.* Addison-Wesley, 1999.

[9] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, and D. Steere. Coda: A Highly Available File System for a Distributed Workstation Environment. *IEEE Transactions on Computers*, 39(4):447–459, Apr. 1990.

[10] M. Satyanarayanan. Mobile Information Access. *IEEE Personal Communications*, 3(1):26–33, Feb. 1996.

[11] D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, and C. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP-15)*, pages 172–183, Cooper Mountain, Colorado, Aug. 1995.

[12] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich. XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *Int. Journal on Personal and Wireless Communications*, April 2002.

[13] MySQL, http://www.mysql.com

[14] Universal Serial Bus, http://www.usb.org

[15] FireWire, http://www.apple.com/firewire/

[16] Bluetooth, http://www.bluetooth.com/

[17] A. Sinitsyn. A Synchronization Framework for Personal Mobile Servers. *Second IEEE International Conference on Pervasive Computing and Communications,* pages 208-212, March 2004.