A Transcoding Server for the Home Domain

François Caron and Stéphane Coulombe
Department of Software and IT Engineering
École de Technologie Supérieure
Montréal, Canada
Francois.C.Caron@gmail.com,
Stephane.Coulombe@etsmtl.ca

Tao Wu
Nokia Research Center
Cambridge, MA 02142, USA
tao.a.wu@nokia.com

***Abstract*—This paper discusses the implementation of a video transcoding service for the Home Domain. The Home Domain is composed of personal computers, consumer electronics and mobile devices that communicate with each other via an IP network. However, due to the diversity of the capabilities of these devices (formats supported, memory, processing power, etc), interoperability is a challenge. For instance, set-top boxes and DVD players support the MPEG-2 video format while mobile phones only support H.263 and/or MPEG-4. Adding a video transcoding service, acting as an intermediate between a digital media server and various digital media players, would allow transparent sharing of the media content within the network. This study has two main objectives. Firstly to insert transparently a transcoding service within the existing Home Domain architecture; and secondly to implement such a service using only open source software components. An FFmpeg-based video transcoding server for the Home Domain was implemented.**

Index Terms — Home Domain, DLNA, transcoding, mobile devices, content sharing, FFmpeg.

I. INTRODUCTION

The Home Domain, as envisioned by the Digital Living Network Alliance (DLNA), is a network of interoperable media-rich devices located in the consumer's homes [1][2]. In the Home Domain, any audiovisual content available within the home network (such as audio and video clips, movies, pictures) can be discovered and rendered on any appropriate device. For instance, a user could identify all available video content within his network (whether located on his PC, DVD player, set-top box (STB), personal video recorder (PVR) or digital video camera) and decide to play some of this content to his own choice of TV, PC, wireless phone or video player. All devices would be able to interact together under various network technologies including: Ethernet, 802.11a/b/g and Bluetooth.

The Universal Plug and Play (UPnP) [5] is a key protocol used in the Home Domain to enable peer-to-peer connectivity between the devices. It allows discovery and control of devices, event notification and presentation of content. The Home Domain is based on other open specifications such as HTTP, IPv4, XML, Ethernet and 802.11a/b/g.

Two sets of specifications have been released by the DLNA (formerly named the Digital Home Working Group (DHWG)): version 1 [3] released in 2004 and version 1.5 [4] released in March 2006.

When this project was started, only the Design Guidelines of version 1 had been released. The mandatory media formats were LPCM for audio, JPEG (resolution of 640x480 noted as small) for images and MPEG-2 for video. These formats are widely supported by consumer electronic devices such as DVD, STB, etc. However, LPCM and MPEG-2 are not supported by mobile devices and the Design Guidelines did not contain any transcoding entity to enable multimedia content interoperability between devices. Under such situations, it was problematic to include mobile devices in the Home Domain. For instance, a wireless device (either a mobile phone or a personal digital assistant) could have been part of the Home Domain using 802.11 or Bluetooth since it is possible to implement on them the required protocols such as UPnP, HTTP, etc. However, for multimedia content, it is not possible for such a device to support MPEG-2 video from a DVD or a STB, since the video resolution and frame rate would require too much bandwidth, memory and processing capability.

This paper addresses the problem of transparently adding and implementing a transcoding server for the Home Domain applicable to versions 1.0 and 1.5. The implementation uses only open source software components. Section II provides a short overview of the Home Domain version 1.0. Section III presents our implementation of a Transcoding Server for the Home Domain. Section IV illustrates how media is accessed when transcoding is performed. Finally, Section V presents how DLNA version 1.5 solves the problem of transcoding and how it compares with our solution.

II. OVERVIEW OF THE HOME DOMAIN (VERSION 1.0)

A. Architectural elements

The Home Domain, as defined in [3] and illustrated in Figure 1, defines two device classes:

- Digital Media Server (DMS): a media storage and sourcing device. It is accessed to retrieve the desired media content. Some examples of DMS include media centers, set-top boxes, PCs, etc.
- Digital Media Player (DMP): device that finds content exposed by DMSs and provides playback and rendering. It can therefore find, select, control and render media content. Some examples of DMP include TVs, stereos, some portable game consoles (with wireless connections), etc.

This project was sponsored by Nokia Research Center.

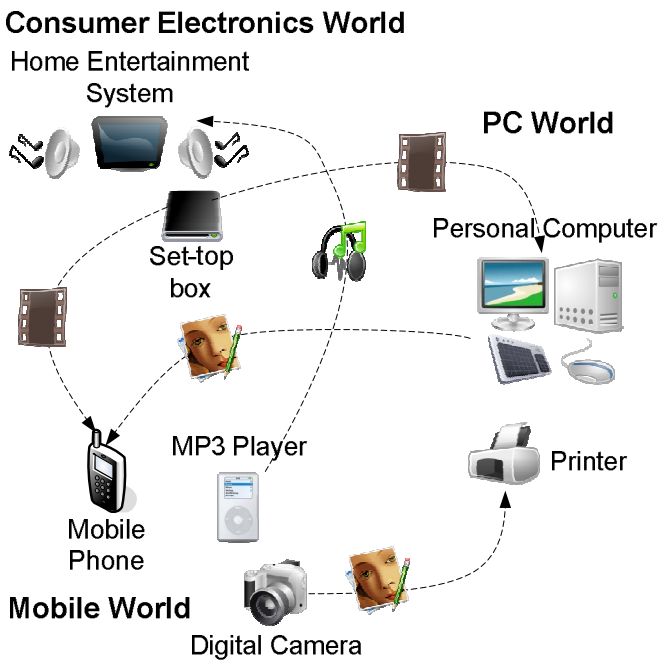

Figure 1: Example of devices within the Home Domain

B. *Media Access in the Home Domain*

Prior to accessing any media, a device will discover other devices within the network and especially the DMSs which can provide content. It may then request a list of available media files to one or many DMSs of interest in the network. Every list will contain information about each available media element: title, date, MIME type, URI where it is available, media format profile (if it matches one of the DLNA format profiles). For downloadable content, the device will issue an HTTP-GET with the URI associated with the content and render it locally. Streaming can also be performed using HTTP.

The Home Domain defines mandatory formats to facilitate interoperability while allowing the use of several optional formats. For instance a DMS could expose a video file in the MPEG-4 optional format as long as it exposes the same content in the MPEG-2 mandatory format. But not all content may be in formats supported by mobile devices and not all DMSs may have the ability to transcode content for them. A transcoding server could overcome such problems.

III. Implementation of a Transcoding Server for the Home Domain

We implemented the key elements of the Home Domain environment in addition to a new Transcoding Server (TS). The system uses only open source software such as thttpd [6], FFmpeg [7] and ezxml [8]. It runs under Linux Fedora Core 3. However, since we were only interested in media-level interoperability issues, we did not integrate the UPnP aspects of the system. The Linux SDK for UPnP Devices could have been used for this purpose [9]. Our solution is not limited to the mandatory and optional formats defined in the DLNA but includes other media formats such as H.263, AMR, etc.

We considered several fundamental requirements when designing the solution. Firstly, it had to be transparent to the DMPs since many are already deployed and can't easily be modified. Therefore we couldn't modify the UPnP protocol to add new services such as discovering and requesting the TS's services. Secondly, we had to minimize the required modifications to the DMSs and such changes had to be easily implementable and fully compatible with UPnP devices.

The proposed solution meets these requirements. It only assumes that we have control over the format of the list of available content that the DMS presents to the DMP. If the DMS is a PC, this is easily achievable since a script is often used to create the list. However, it will probably not work on devices where the control over such lists is very limited (e.g. set-top-boxes) but we present a workaround for this case.

In our solution, the TS acts as a man-in-the-middle. When a device requests the list of available media files, the DMS returns a list of specially constructed URIs pointing to the TS. Indeed, for each available media item, the URI will be specially constructed to point to the TS (rather than the origin DMS) and contain information about the original media item in the DMS (so the TS can retrieve it and adapt it). An example of such specially constructed URI is presented in step 3 of Figure 2. When a DMP requests a specific file in the list, the request is made directly to the TS, which has all the necessary information in the URI to retrieve the original media file from its origin server and to adapt it to the capabilities of the DMP. For instance, a video clip could be available in the following two formats using the following URIs :

- MPEG-2: http://hd.dms.home/movie.mpg (associated with a video/mpeg MIME type and possibly a MPEG_PS_NTSC media format profile)
- H.263: http://hd.ts.home?file=http://hd.dms.home/movie.mpg (associated with a video/3gp MIME type and no media format profile since there is no DLNA H.263 format profile)

where `hd.ts.home` and `hd.dms.home` are the TS and DMS's URIs respectively. Note that when DLNA media format profile exists for a media item, it must be provided along with its URI and MIME type in the list.

In this example, the MPEG-2 version of the content is readily available from the DMS. However the H.263 version is obtained through the TS. The extra parameter in the second URI tells the TS where to get the original content. The TS will then adapt the content to meet the capabilities of the device requesting the content. Section IV provides more details on how this is performed.

The proposed manner of adding a TS in the system is quite easy to implement. We only need to modify the DMSs to provide a list of content pointing to the TS in all the formats we believe necessary to ensure interoperability in the home network. For instance, for video, we could add URIs for MPEG-4 (VSP) and H.263 versions in addition to the mandatory MPEG-2 format. For MPEG-4 (VSP) however, since DLNA profiles exist, we would have to produce a different URI for each desirable media format profile (i.e. a different entry in the list). The creation of such a list containing specially constructed URIs can be achieved by a script at the DMS when the DMS is a PC or offers this flexibility.

A. Digital Media Server

In our implementation of the Home Domain, the DMS is an HTTP browsing server. It displays a list of media files from which the DMP can choose from. The DMP accesses the DMS using the HTTP-GET method.

The thttpd open source Web server [6] was used because it is small, compact, and easy to use. Our DMS implementation uses a Perl script to build the HTTP browsing page. It dynamically loads the available content from a multimedia database and displays it in a single column list in alphabetical order. The script creates the URIs using a special format for various media formats in addition to the original one. The newly formed URIs point to the TS and contain a URI to the original file as a parameter as described in the previous sub-section.

B. Digital Media Player

A mobile phone and a PC were used as DMPs. The PC was used to emulate different devices with various capabilities (memory, supported media formats, etc) by changing the user-agent associated with the request to the DMS.

C. Transcoding Server

The transcoding server is the key element of our system. It is composed of a Web server (WS) similar to the one used for the DMS, a transcoding application (FFmpeg), a User-Agent Profile (UAProf) database, a multimedia cache, and an intelligent controller.

- Web server (WS): The WS is used to process the HTTP requests made by the DMP (it doesn't support UPnP). A Perl script handles the incoming requests, loads the intelligent controller to manage the transcoding and provides the adapted file to the client. If the intelligent controller cannot adapt the multimedia file for the user-agent, the WS will return an error page.

- Transcoding application. FFmpeg performs all the audiovisual transcoding. The FFmpeg projet [7] is an open source project that offers transcoding between major media formats. It also offers audiovisual adaptation functionalities such as resolution reduction, frame rate reduction, cropping, etc. The transcoding application was modified to compute a bit budget allocation for the video and audio channels. If the video did not have audio, the audio budget would be reallocated to the video and vice versa.

- UAProf database (UAProf DB). A User Agent Profile (UAProf) [10] database contains UAProf files corresponding to each user-agent (device). UAProf files are XML files containing terminal capability information of the device such as screen resolution and supported audiovisual formats. When a device makes an HTTP request for content to the TS, the WS will provide the user-agent to the UAProf database to obtain the specific terminal's capabilities required to perform transcoding. We added new sections in our UAProf files to provide information about acceptable download size and maximum streaming rate (we could even have added new sections to include DLNA-related capabilities such as the set of supported DLNA media profiles). We chose UAProf because it is an extensible and widely used terminal capability description standard in the mobile world. However this approach forces us to manually obtain or create each DMP's UAProf capability description file. We could learn the DLNA capabilities by issuing a CMS:GetProtocolInfo() to the DMP and store the results in the UAProf DB; making the process semi-automatic. If a device is not in the UAProf DB, a set of minimal capabilities is assumed.

- Multimedia cache. The multimedia cache is used mainly for reducing processing requirements and improve response time of the system. When a file has been transcoded for a given user-agent, a copy of the file is kept into the multimedia cache. When another device having that same user-agent value requests the same file, the intelligent controller will not call the transcoding application but return the cached file.

- Intelligent controller (IC). The IC is a C program created to make transcoding decisions in order to obtain the best possible quality for each user-agent. It uses the UAProf database to obtain the necessary information on the user-agents and to select or compute the best options used when calling the FFmpeg transcoding application. Alternatively, the IC could use the desired media format profile, provided as an extra parameter in the URI, to determine the appropriate FFmpeg parameters. The UAProf file is parsed using ezxml [8]. Before calling the transcoder, the IC checks the multimedia cache for an existing version of the file for the user-agent.

IV. MEDIA CONTENT ACCESS WITH TRANSCODING

The detailed media content access with transcoding steps are illustrated in Figure 2 for an example of MPEG-4 AVC to H.263 conversion (note that a fully DLNA compliant system would use UPnP for steps 1 and 2).

1. The DMP client accesses to the DMS using HTTP. It requests the main page of the DMS.
2. The DMS responds to the client's request and sends the index.html page for the client to browse the available media files. The index.html file contains a list of several specially constructed URIs referring to various versions of media content (The index.html page was created by the Perl script handling the incoming request. This ensures that the media content list is always up to date.)
3. The DMP client selects a media file to view. The DMP issues an HTTP-GET method with the specially constructed URI corresponding to the selected file. Note that the URI points to the TS but contains the origin media file name as a parameter. Although not illustrated, the DMP provides its user-agent information in the HTTP request (the user-agent header information is normally provided in any HTTP transaction to identify the user-agent or browser making the request).
4. The WS part of the TS requests the original file to the DMS using a conventional HTTP-GET method.
5. The DMS provides the requested file in the HTTP response.

6. The WS loads the IC. It passes the user-agent and the media file requested and waits for the IC to finish.
7. The IC accesses the UAProf DB to retrieve information about the user-agent. If the user-agent profile cannot be found, the default user-agent profile XML file will be used.
8. The UAProf DB returns a list of terminal capabilities in an XML format.
9. The IC calls the transcoding application with appropriately selected and computed parameters. The IC waits for the transcoder to complete its task.
10. The transcoding application completes the media adaptation and returns the control to the IC.
11. The IC looks at the return value from the transcoding application and returns the control to the WS.
12. The WS part of the TS sends the adapted file to the client (the HTTP response).

V. TRANSCODING IN DLNA VERSION 1.5

After this project had been completed, the DLNA released its Networked Device Interoperability Guidelines, version 1.5 [4]. This new specification includes several new device categories and classes. For the Home Network Domain (HND) devices, the mandatory formats are still: JPEG (small), LPCM and MPEG-2 video. For Mobile Handheld Devices (MHD), the mandatory formats are: JPEG (small), MPEG-4 AAC, MP3 and MPEG-4 AVC video.

More relevant for the scope of this work, the specifications introduce the Home Infrastructure Device (HID) category which includes the following classes:

- Mobile Network Connectivity Function (M-NCF): provides bridging between MHD and HND networks.
- Media Interoperability Unit (MIU): provides content transformation (transcoding) between required media formats for the HND and the MHD Device Categories.

The role of the MIU is to convert content between HND and MHD mandatory formats to ensure interoperability between the two device categories. The MIU performs this operation by virtualizing other M-DMSs or DMSs, offering their content in other formats and transforming it on the fly when requested. Note that if an M-DMP supports a format in which a DMS provides content, it can contact it directly without passing through the MIU. The same applies between a DMP and an M-DMS.

Our solution works for the Design Guidelines version 1 in which no transcoding solution was provided. This is already a significant benefit. The solution offered by DLNA v1.5 provides similar benefits to our solution in that it offers existing content in a wider variety of formats. Their solution is appealing from the perspective that a MIU behaves just like another DMS or M-DMS in the network. Our solution requires that at least one DMS in the home network be flexible enough to support, through a script, the creation of specially constructed URIs. This device, normally a PC, could add to its own list of content, a list of content offered by other DMSs but in a wider set of formats (such URIs would point to the TS and refer to other DMSs for original content). Alternatively, we could integrate our transcoding system as part of the MIU.

Our TS is able to perform very custom transcoding without having to generate a very long list of media versions when the media types are not part of the existing DLNA media format profiles. For instance, in the DLNA solution, there are close to 100 different MPEG-4 AVC profile IDs taking into account various audio, video, and system characteristics. Creating an extensive list of all possible content variations in all possible formats could be bandwidth inefficient and overwhelming for limited devices such as mobile phones. With our solution, there would be a single URI for an H.263 version; the TS would create the best version based on the specific user-agent characteristics. Of course, to be DLNA-compliant, we would have to provide an extensive list of versions for content formats for which a DLNA media format profile exists. Therefore formats not covered by DLNA permit to reduce the amount of versions to announce while using those covered by DLNA allow the devices more control on the characteristics of the content they receive (for instance they may support high-resolution video but only want the smaller version).

Our implementation also expands the possible formats outside the mandatory formats of HND and MHD. Our solution supports H.263, MPEG-2 and MPEG-4 video decoding and encoding. The main test cases of interest include MPEG-2 to MPEG-4 (VSP) and H.263 (for STB and DVD movies to mobile), and H.263 to MPEG-2 (for watching a mobile video clip on a TV) but it is also possible to perform transcoding from MPEG-4 (VSP and AVP) to MPEG-2, H.263 to MPEG-4 (VSP) and MPEG-4 (VSP and AVP) to H.263.

The AAC (using the faac and faad2 libraries [11]), AMR-NB and AMR-WB (using external AMR libraries [12]), LPCM and MP2 are the main audio/speech formats targeted by our tests. The use cases of interest include LPCM to AAC, AMR-WB and AMR-NB (in decreasing order of preference) to make content, stored in mandatory DLNA format, available to mobile devices. We also were interested in conversions from AMR-NB, AMR-WB and AAC to LPCM to enable playback of mobile content to DLNA-compliant devices.

It is possible to obtain a more extensive list of the supported audio and video formats from the FFmpeg project [7] page. Some formats are not readily supported by FFmpeg but may be added through the use of external libraries.

One drawback we found with our solution is with the transfer of long video clips. In such a case, the DMP has to wait a long time for the TS to retrieve the original video clip, transcode it and transfer the response to the DMP. This may cause timeouts at the DMP or user's impatience. Another drawback which has already been mentioned is that we must have some control over at least one DMP to generate the specially formed URIs. Finally, we have to configure manually the UAProf DB. But we consider the drawbacks to be manageable considering to the numerous advantages: offering a transcoding service for devices compliant with Design Guidelines v1 and DLNA v1.5, more customized transcoding and wider set of formats supported compared to DLNA v1.5 transcoding solution, and fully transparent to all DMPs and most DMSs.

VI. Conclusions

This project implemented of a video transcoding service for the Home Domain especially designed for the Design Guidelines v1 but which could be used with DLNA v1.5. The system is based on open source software components. Another advantage of the solution is that it can provide custom transcoded content to any device often without having to create an extensive list of versions. Finally is supports an extensive set of media format conversions.

Disclaimer

This article does not necessarily represent Nokia's position nor should be used to predict the features of future Nokia products.

References

[1] Digital Living Network Alliance, "DLNA Overview and Vision WhitePaper 2006," 2006, http://www.dlna.org/industry/about/dlna_white_paper_2006.pdf.

[2] Digital Living Network Alliance, "DLNA Overview and Vision WhitePaper 2004," 2004, http://www.dlna.org/industry//about/DLNA_Overview.pdf.

[3] Digital Home Working Group, "Digital Home Networking Group Design Guidelines," version 1, June 2004.

[4] Digital Living Network Alliance, "DLNA Networked Device Interoperability Guidelines," version 1.5, March 2006.

[5] UPnP Device Architecture Version 1.0, UPnP Device Architecture Version 1.0, UPnP Forum, June 13, 2000.

[6] ACME Laboratories, "tiny/turbo/throttling HTTP server," version 2.25b, 2006, http://www.acme.com/software/thttpd/.

[7] FFmpeg Project, 2006, http://ffmpeg.mplayerhq.hu/.

[8] Aaron Voisine , "ezXML - XML Parsing C Library," version 0.8.6, http://ezxml.sourceforge.net/.

[9] SourceForge.net, "Linux SDK for UPnP Devices (libupnp), " version 1.3.1, March 2006, http://upnp.sourceforge.net/.

[10] WAP-248-UAProf, "Wireless Application Protocol, User Agent Profile," WAP Forum, May 2001, http://www.openmobilealliance.org.

[11] Audiocoding.com, 2006, http://www.audiocoding.com/

[12] 3GPP, "3GPP Specification series", 2006, http://www.3gpp.org/ftp/Specs/html-info/26-series.htm

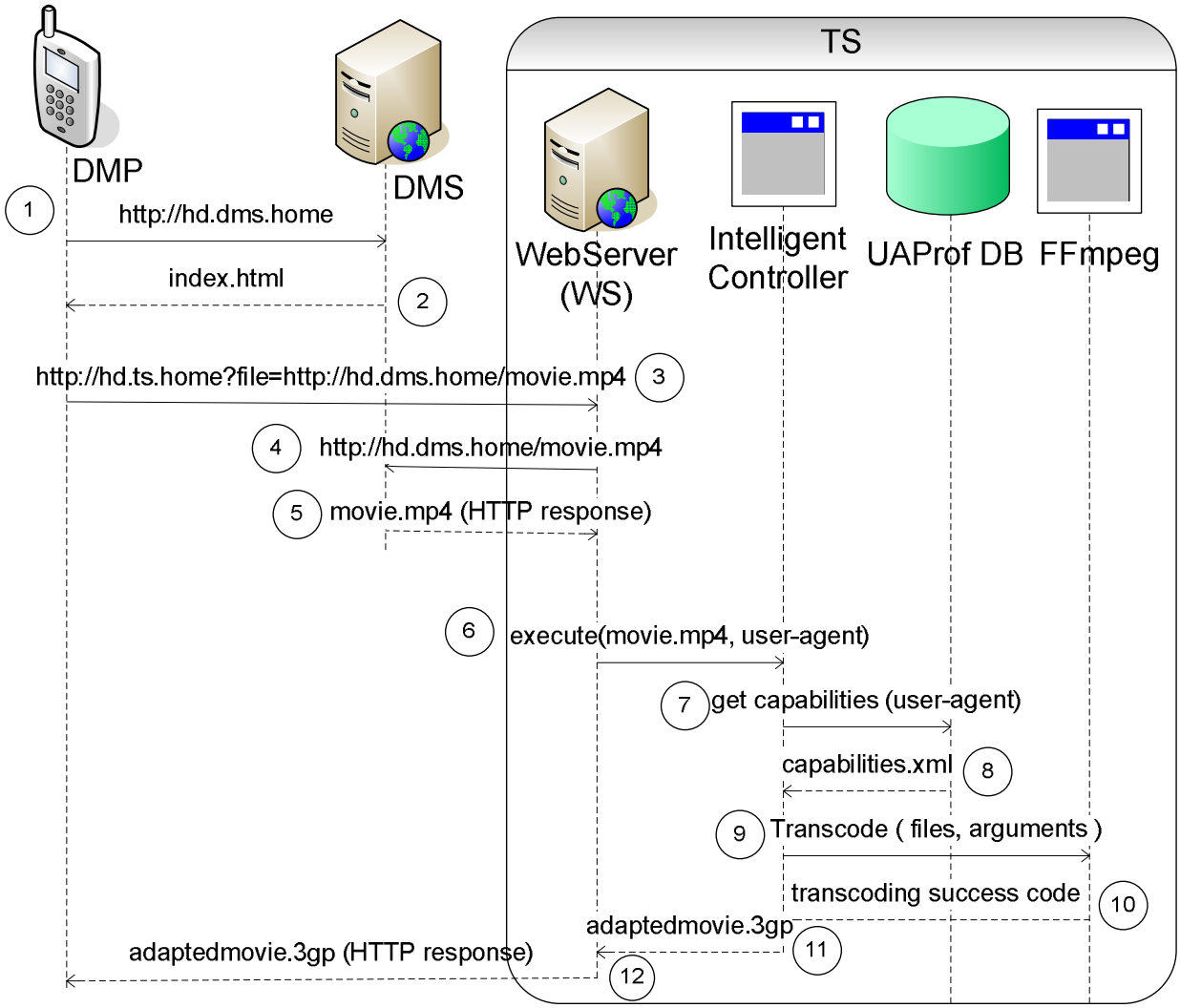

Figure 2: Detailed transaction flow for multimedia content access with transcoding.