Semantic Media Decision Taking using N3Logic

Wim Van Lancker, Davy Van Deursen, Ruben Verborgh, and Rik Van de Walle

Ghent University – IBBT, ELIS – Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
{wim.vanlancker,davy.vandeursen,ruben.verborgh,rik.vandewalle}@ugent.be
http://multimedialab.elis.ugent.be/

Abstract

In this paper, we introduce a Media Decision Taking Engine (MDTE), enabling the automatic selection and/or rating of multimedia content versions, based on the available context information. The presented approach is fully semantic-driven, which means that we not only semantically model the context information, but also the decision algorithms themselve, which are represented in N3Logic, a rule language that extends RDF. The decision rules are based on a rating function, supporting the specification of weights and affinity parameters for each environment property. Finally, we show how the MDTE is integrated in a media delivery platform, using the provisions of the existing Web infrastructure.

Keywords: Decision Taking, Media Selection, N3Logic, RDF, Rules

1 Introduction

Recent years have witnessed an increasing heterogeneity in the multimedia landscape on different fronts. First, there is a growing diversity in end-user devices that are able to consume multimedia content. In particular, these devices may vary in terms of screen size, supported media formats, and battery life. Next, network technologies, used to transport the multimedia content to the end-user may differ in terms of bandwidth, jitter, and error robustness. Furthermore, the number of multimedia coding standards has grown significantly over the last few years, especially with the introduction of new media coding formats such as H.264/AVC, Scalable Video Coding (SVC), VP8, and HD Photo. At the same time, older standards such as H.262/MPEG-2 Video and MP3 are still present. Next to coding formats, there also exists a wide variety of delivery formats and protocols [24]. For instance, video can be delivered in MP4 or WebM containers over HTTP, in RTP packets over RTSP, or using the recently introduced HTTP adaptive streaming paradigm.

The efficient delivery of multimedia content in today's world of ubiquitous multimedia consumption is an important technological challenge, due to the above described heterogeneity. Ideally, the delivery of multimedia content needs to occur in a transparent way in order to obtain Universal Multimedia Access (UMA, [26]). Today, content providers try to cope with the diversity in usage environments (i.e., devices, network technologies, users) by offering different *versions* of the same multimedia content. For instance, YouTube has about ten configurations, varying in coding format and resolution[1]. Currently, the choice of which version is delivered to the end-user is left to the user. An alternative to storing multiple versions of the same multimedia content is the use of scalable media codecs (such as SVC). However, despite numerous recent standardization efforts, scalable media codecs did not find much uptake yet. Nonetheless, we also take into account scalable media codecs in our approach and provide support for them.

In this paper, we present our Media Decision Taking Engine (MDTE), a tool for automatic selection of multimedia content versions, based on the available context information. The presented approach is fully semantic-driven, which means that we not only semantically model the context information, but also the decision algorithms themselve. Semantic in this context means that the represented information is machine-understandable [4]. For this purpose, we use the Resource Description Framework (RDF, [11]) to represent the context information. The decision algorithms are described in N3Logic [3], a rule language that extends RDF with syntax for nested graphs, quantified variables, predicates for implication, and built-in functions.

The paper is organized as follows. Sect. 2 discusses related work. In Sect. 3, the general architecture and context information flow is presented, while the models for the context are discussed in Sect. 4. We present our formalized semantic decision rules driving the MDTE in Sect. 5. Further, in Sect. 6 we elaborate on how the MDTE can be integrated into a media delivery platform that we introduced in previous work. Finally, conclusions are drawn and future work is discussed in Sect. 7.

2 Related Work

Most related work around media selection and adaptation decision taking is situated in the context of the MPEG-21 Digital Item Adaptation (MPEG-21 DIA, [10]) specification. More specifically, the Usage Environment Description (UED) tools, Universal Constraints Description (UCD) tool, and Adaptation Quality of Service (AQoS) tool provide models that can be

[1] http://en.wikipedia.org/wiki/YouTube#Quality_and_codecs

used by media selection and adaptation decision taking engines.

For instance, Mukherjee *et al.* show universal methods based on pattern search to process the information provided by the MPEG-21 DIA tools to make decisions [17]. Fully MPEG-21 DIA-based decision taking engines are also used in [8] and [9]. Further, Szwabe *et al.* describe in [20] how dynamic media adaptation based on a rate control scheme is realized through an MPEG-21 DIA-based Adaptation Decision Taking Engine (ADTE), while Köhncke *et al.* make use of preference graphs and composition algorithms for adaptation tools [12].

The approaches described by Lopez *et al.* in [14] and Prangl *et al.* in [19] come closest to the approach we describe in this paper. Lopez *et al.*, which also use preference graphs to obtain decisions, distinguish between hard constraints (i.e., imposed by the terminal and network) and soft constraints (i.e., preferences by the user). Prangl *et al.* focus on a generic semantic-based audio-visual utility model for MPEG-21 DIA that aims to enhance the multimedia experience for the user. More specifically, based on media characteristics, the perceptual quality is estimated and taken into account during the decision taking process.

Related work outside the scope of MPEG-21 DIA is for instance described by Lum *et al.* in [15]. They present a content adaptation system that decides on the optimal content version for presentation and the best strategy for deriving that version, based on CC/PP-based context information.

The work described in this paper (although our concepts are also based on MPEG-21 DIA) differs in a number of ways from the above described related work:

- we use a fully formally described context model (see Sect. 4);
- our algorithms are not hardcoded in software but formally described in terms of rules, enabling verification and proof generation of the results (see Sect. 5);
- we also distinguish between hard and soft constraints, but we do not map them onto terminal/network and user preferences respectivily (see Sect. 5);
- we elaborate on how the different parts of the context model are populated in a real server platform (see Sect. 6);
- we provide a workflow for using our MDTE in a fully Web-enabled media server environment (see Sect. 6).

3 Media Decision Taking Engine: Approach

The MDTE is fully driven by context information on the one hand and decision taking rules on the other hand. Fig. 1 provides an overview of all this information. The context information consists of three parts: information about the multimedia content, information about the media server capabilities, and information about the usage environment.

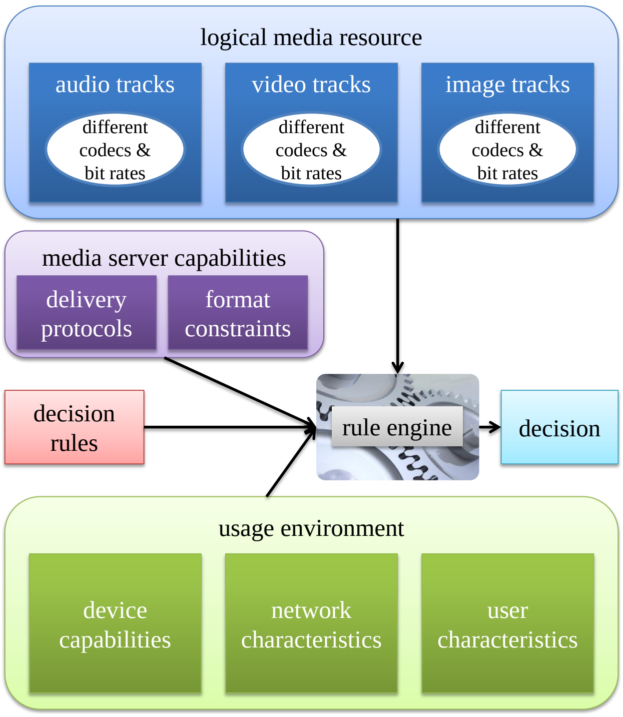

Fig. 1. Information consumed by the Media Decision Taking Engine

The multimedia content is organized in terms of logical and physical media resources. A logical media resource collects all media tracks about one media item. For example, a logical media resource can correspond to a music performance of a rock band. Examples of media tracks belonging to this logical media resource are an H.264/AVC video track having a resolution of 1920x1080, an Ogg Vorbis audio track having a bit rate of 128 kbit/s, and a PNG image track (representing a screenshot) with a resolution of 704x576. Each track contains a detailed description of its technical parameters.

The media server capabilities provides an overview of which delivery formats and protocols are currently supported by the media server. This is necessary because for some delivery protocols, server extensions (beyond HTTP) are necessary (e.g., RTSP and RTMP require a specialized media streaming server). Additionally, the supported delivery formats are also described (e.g., MP4, WebM, Ogg). Finally, a number of constraints are present linking coding formats such as H.264/AVC and AAC to delivery formats and protocols. For example, VP8 and Ogg Vorbis tracks are typically stored in a WebM container and not in an MP4 container. Although most media delivery formats do not impose restrictions to the underlying coding formats, the available media players stick to the typical combinations.

The usage environment consists of device capabilities, network characteristics, and user characteristics. Device capabilities provide information such as the screen size and the supported media formats. Network characteristics are for instance the available bandwidth, latency, and type of network (e.g., 3G, WIFI). Finally, user characteristics are particular choices made by the user beforehand. For example, the user prefers WebM as delivery format.

How the context is modeled is presented in Sect. 4. The decision rules are discussed in Sect. 5. Since we are using N3Logic to represent the rules implementing the decision taking algorithms, the rule engine must be compliant to N3Logic. In this paper we use the Euler Yap Engine (EYE, [5]) because of its performance in terms of execution speed compared to other reasoners [2, 18]. The decision itself consists of a ranked list of physical media URIs that fit the given usage environment and can be served by the media server.

4 Context Modeling

In this section, we describe which models are used for the represenation of context information. All models are implemented using the Web Ontology Language (OWL, [16]). Moreover, where possible, existing models (or ontologies) are being reused or extended. Note that we do not rely directly on the MPEG-21 DIA model (see also Sect. 2), because it is too detailed for our purposes. Nevertheless, we do reuse concepts from the MPEG-21 DIA model.

4.1 Media Characteristics

To represent media characteristics such as resolution, codec, and frame rate, we use the Ontology for Media Resources 1.0 (a.k.a., Media Annotation (MA) ontology) [13]. The ontology, currently being standardized within the W3C, is designed to facilitate cross-community data integration of information related to media in the Web. The specification defines a core set of metadata properties for media resources, along with their mappings to elements from a set of existing metadata formats. For instance, different categories supported by the ontology are identification, content description, rights, distribution, fragments, and technical properties. For this purpose, we mainly use the provisions for identification, fragments, and technical properties.

We distinguish between logical and physical media resources, as described in Sect. 3. A logical media resource contains a number of tracks. A physical media resource is then a selection of one or more tracks, combined with a container format and delivery protocol. Tracks can differ in terms of coding format, resolution, bit rate, etc.

Within the MA ontology, a track can be identified through a Media Fragment URI [21]. Note that the Media Fragment URI 1.0 specification (currently also being standardized within W3C) enables the addressing of media fragments in the Web using Uniform Resource Identifiers (URI); three different axes are supported: temporal (i.e., a time range), spatial (i.e., a spatial region), and track (i.e., a track contained in the media resource). In this paper, we use Media Fragment URIs to identify and retrieve track media fragments.

Listing 1.1. Representing media characteristics.

```
@prefix ma: <http://www.w3.org/ns/ma-ont#>.
@prefix nsa: <http://multimedialab.elis.ugent.be/
     organon/ontologies/ninsuna#>.
@prefix nf: <http://multimedialab.elis.ugent.be/
     organon/ontologies/ninsuna-formats#>.
@prefix : <http://ninsuna/test#>.

<http://ninsuna/test>
  a ma:MediaResource;
  ma:hasTrack :1 , :2 , :3 , :4;
  ma:numberOfVideoTracks "2"^^xsd:int;
  ma:numberOfAudioTracks "2"^^xsd:int;
  ma:locator <http://ninsuna/test.mp4?track=1;3>;
  ma:locator <http://ninsuna/test.mp3?track=3>;
  # ...

:1  a ma:VideoTrack;
  ma:frameHeight "240"^^xsd:int;
  ma:frameWidth "320"^^xsd:int;
  ma:frameRate "25.0"^^xsd:double;
  ma:title "1";
  ma:format nf:H264;
  nsa:codingProfile nf:AVC_BASELINE;
  ma:averageBitrate "323.747"^^xsd:double;
  nsa:maxBitrate "686.421"^^xsd:double;
  ma:duration "150.0"^^xsd:double.

:2  a ma:VideoTrack;
  ma:frameHeight "480"^^xsd:int;
  ma:frameWidth "640"^^xsd:int;
  ma:frameRate "25.0"^^xsd:double;
  ma:title "2";
  ma:format nf:H264;
  nsa:codingProfile nf:AVC_MAIN;
  ma:averageBitrate "817.721"^^xsd:double;
  nsa:maxBitrate "1023.521"^^xsd:double;
  ma:duration "150.0"^^xsd:double.

:3  a ma:AudioTrack;
  ma:title "3";
  ma:format nf:MP3;
  ma:averageBitrate "81.532"^^xsd:double;
  nsa:maxBitrate "127.541"^^xsd:double;
  ma:duration "150.0"^^xsd:double.

:4  a ma:AudioTrack;
  ma:title "4";
  ma:format nf:VORBIS;
  ma:averageBitrate "95.422"^^xsd:double;
  nsa:maxBitrate "135.643"^^xsd:double;
  ma:duration "150.0"^^xsd:double.
```

Listing 1.1 shows an example RDF instance of a logical media resource containing an MP3 and a Vorbis audio track and two H.264/AVC video tracks with a different resolution and bit rate. The OWL implementation of the MA ontology[2] is used, together with a number of extensions that we have defined. More specifically, formal definitions of coding formats and coding profiles have

[2] http://dev.w3.org/2008/video/mediaann/mediaont-1.0/ma-ont.owl

Listing 1.2. Representing server capabilities.

```
@prefix nss: <http://multimedialab.elis.ugent.be/
     organon/ontologies/ninsuna-server#>.
@prefix nf: <http://multimedialab.elis.ugent.be/
     organon/ontologies/ninsuna-formats#>.

nss:HTTP_MP4 a nss:DeliveryConfiguration;
  nss:deliveryProtocol nf:HTTP;
  nss:deliveryFormat nf:MP4;
  nss:codingFormat nf:AAC;
  nss:codingFormat nf:H264;
  nss:codingFormat nf:MP4V;
  nss:codingFormat nf:MP3.

nss:HTTP_WEBM a nss:DeliveryConfiguration;
  nss:deliveryProtocol nf:HTTP;
  nss:deliveryFormat nf:WEBM;
  nss:codingFormat nf:VP8;
  nss:codingFormat nf:VORBIS.

nss:HAS_M3U8 a nss:DeliveryConfiguration;
  nss:deliveryProtocol nf:HAS;
  nss:deliveryFormat nf:M3U8;
  nss:codingFormat nf:AAC;
  nss:codingFormat nf:H264;
  nss:codingFormat nf:MP3.
```

Listing 1.3. Representing a usage environment.

```
 1  @prefix ue: <http://multimedialab.elis.ugent.be/
         organon/ontologies/adte/usage-environment#>.
    @prefix nf: <http://multimedialab.elis.ugent.be/
         organon/ontologies/ninsuna-formats#>.
    @prefix : <foo#>.

 5  :MyUE a ue:UsageEnvironment;
      ue:maxFrameHeight "320"^^xsd:int;
      ue:maxFrameWidth "480"^^xsd:int;
      ue:supportedFormat nf:MP4;
      ue:supportedFormat nf:H264;
10    ue:supportedFormat nf:AAC;
      ue:supportedProfile nf:AVC_BASELINE;
      ue:supportedProtocol nf:HTTP;
      ue:supportedProtocol nf:HAS;
      ue:deviceProfile [
15      a ue:DeviceProfile;
        ue:dimension [a ue:Dimension;
          ue:maxFrameHeight "288"^^xsd:int;
          ue:maxFrameWidth "352"^^xsd:int;
          ue:maxFrameRate "30.0"^^xsd:double];
20      ue:dimension [a ue:Dimension;
          ue:maxFrameHeight "240"^^xsd:int;
          ue:maxFrameWidth "320"^^xsd:int;
          ue:maxFrameRate "10.0"^^xsd:double];
        ue:maxBitrate "384.0"^^xsd:double;
25      ue:profile nf:AVC_BASELINE
      ];

      ue:networkConnectionType ue:WIFI;
      ue:availableBandwidth "1000.0"^^xsd:double;
30
      ue:preferredDeliveryProtocol nf:HAS.
```

been defined (e.g., `nf:H264` represents H.264/AVC)[3]. Also, we defined a number of extra properties, such as the maximum bit rate (`nsa:maxBitrate`) and the codec profile (`nsa:codingProfile`)[4] [25].

4.2 Server Capabilities

For describing the capabilities of a media server, we developed our own model (by lack of existing models). The model must be able to represent which delivery formats, codecs, and protocols are currently supported by the media server. Therefore, there is one central concept, `nss:DeliveryConfiguration`, which combines a `nss:deliveryProtocol`, a `nss:deliveryFormat`, and one or more `nss:codingFormats`[5].

An example RDF instance of a server capability description is provided in Listing 1.2. In this example, three delivery configurations are supported by the media server:

– H.264/AVC, MPEG-4 Visual, MP3 and/or AAC stored in an MP4 container and sent over HTTP;
– VP8 and/or Ogg Vorbis stored in a WebM container and sent over HTTP;
– H.264/AVC, AAC, and/or MP3 described in an M3U8 manifest file and delivered using HTTP adaptive streaming.

4.3 Usage Environment

To represent the usage environment, we rely on the concepts defined within MPEG-21 DIA. More specifically, we distinguish between device capabilities (terminal capabilities in MPEG-21 DIA), network characteristics, and user characteristics. However, for each of these three categories, we only use a subset of the available properties. Also, we do not reuse the MPEG-21 DIA model directly, but rather flatten the structure (except for the three main categories) so that it fits better in an OWL ontology. An example RDF instance of a usage environment is depicted in Listing 1.3.

4.3.1 Device Capabilities Only device capabilities that are useful for media decision taking are described. For instance, the screen size of the device and the supported media formats, profiles, and protocols. Additionally, we model the concept of `ue:DeviceProfiles`. The latter defines for a particular `nf:CodingProfile` and `ue:maxBitrate` a number of possible `ue:Dimensions`. A `ue:Dimension` is a combination of `ue:maxFrameRate`, `ue:maxFrameHeight`, and `ue:maxFrameWidth`. In other words, a device profile provides, for a certain coding profile at a certain bit rate, a number of possible combinations of frame rate and resolution. Lines 6-26 in Listing 1.3 illustrate the properties for describing the device capabilities.

4.3.2 Network Characteristics For the moment, we only have a limited description of the network

[3] The ontology is available at `http://multimedialab.elis.ugent.be/organon/ontologies/ninsuna-formats`.

[4] The extensions can be found on `http://multimedialab.elis.ugent.be/organon/ontologies/ninsuna`

[5] The formalized model (implemented in OWL) can be found on `http://multimedialab.elis.ugent.be/organon/ontologies/ninsuna-server`

characteristics. More specifically, we describe the `ue:NetworkConnectionType` (e.g., WIFI, UMTS, etc.) and the `ue:availableBandwidth`, as illustrated on lines 28-29 in Listing 1.3. Network characteristics could be extended with more detailed network conditions and properties in the future.

4.3.3 User Characteristics We distinguish between user preferences and user choices. For example, a user can prefer that its media is delivered using HTTP adaptive streaming (note that this choice can also be dependent on the device or network). Line 31 in Listing 1.3 illustrates this preference. A user can also choose a certain parameter (e.g., a user wants its media to be delivered over RTSP). Note that we do not model user preferences that are related to the content of the media resources, because we are not focussing on recommending media resources based on their content in this paper.

5 Semantic Decision Rules

In this section, we explain the decision rules that drive the MDTE, by taking into account the context information. The goal of the MDTE is to provide a (ranked) list of so-called Possible Delivery Configurations (PDCs). A PDC is a combination of media tracks, a chosen delivery format, and a chosen delivery protocol (e.g., track 1 and 2, packaged in an MP4 container and delivered over HTTP). Further, a PDC corresponds to a physical media resource, which can be represented by means of a Media Fragment URI (e.g., `http://ninsuna/test.mp4?track=1;2`) (see Sect. 4.1).

Overall, the decision taking process can be split up in two main parts:

- build a list of all PDCs for a given context;
- rank the PDCs according to the estimated Quality of Experience (QoE) of each PDC.

The next subsections provide the details for those two parts. Moreover, we elaborate on how these decision rules are modeled in N3Logic. This way, they can be interpreted by a generic rule-based Semantic Web reasoner such as Eye.

5.1 Excluding Delivery Configurations

As already introduced in Sect. 2, we distinguish between hard and soft constraints imposed by the usage environment. Ignoring hard constraints will result in PDCs that are not working within the given usage environment. For example, if a device does not support the RTSP delivery protocol, sending the media resource in a delivery configuration using RTSP will fail. Therefore, all PDCs that do not meet the hard constraints are excluded. Properties imposing hard constraints are supported coding formats, supported or chosen delivery formats, and supported or chosen delivery protocols. Note that we do not consider

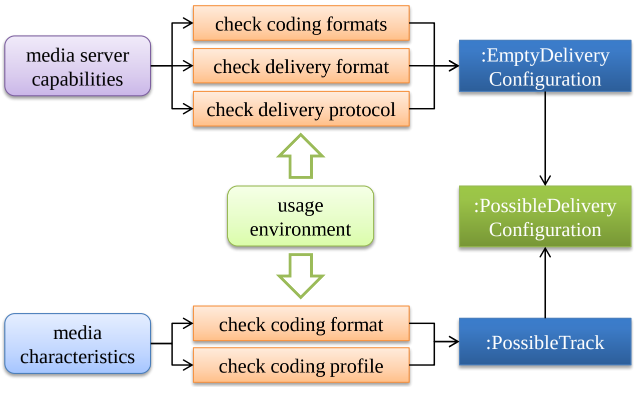

Fig. 2. Building possible delivery configurations

Listing 1.4. Checking the delivery protocol (expressed in N3Logic rules)

```
{
  ?dc a nss:DeliveryConfiguration.
  ?dc nss:deliveryProtocol ?protocol.
  ?ue a ue:UsageEnvironment.
  ?ue ue:supportedProtocol ?protocol.
}
=>
{?dc a :SupportedProtocolDeliveryConfiguration.}.

{
  ?dc a nss:DeliveryConfiguration.
  ?ue a ue:UsageEnvironment.
  ?ue ue:supportedProtocol nf:Unspecified.
}
=>
{?dc a :SupportedProtocolDeliveryConfiguration.}.
```

parameters such as the screen size as hard constraints because the screen size of a device does not necessarily corresponds with the maximum resolution the video player of that device can handle.

In order to build the list of PDCs, a number of steps are executed; these are also visualized in Fig. 2. In the first step, we compare the given usage environment (i.e., Listing 1.3) with the server capabilities (i.e., Listing 1.2). Only the `nss:DeliveryConfigurations` fullfilling the following conditions are kept for further processing:

- the usage environment supports or chooses coding formats specified in the delivery configuration, or has no specified coding formats;
- the usage environment supports or chooses the delivery format specified in the delivery configuration, or has no specified delivery format;
- the usage environment supports or chooses the delivery protocol specified in the delivery configuration, or has no specified delivery protocol.

An example of how these conditions can be checked using formalized rules is given in Listing 1.4. In the first rule of this example, we check if the usage environment supports the delivery protocol specified in the server delivery configuration. The second rule is triggered in case the usage environment does not specify its supported delivery protocols. In the latter case, we regard every `nss:DeliveryConfiguration` as a valid candidate. Further, the listing shows how valid candidate delivery configurations are promoted to an intermediate class, i.e., `:SupportedProtocolDeliveryConfiguration`. We have constructed intermediate classes for each of the above described conditions. When all these conditions hold, i.e., the delivery configuration is an instance of all these intermediate classes, we promote the delivery configuration to the `:EmptyDeliveryConfiguration` class. In our example, we can see that `nss:HTTP_MP4` and `nss:HAS_M3U8` are valid delivery configuration candidates and that `nss:HTTP_WEBM` is excluded due to the fact that `nf:WEBM` is not supported as delivery format in the given usage environment.

The second step compares the characteristics of the available tracks of the requested media resource (i.e., Listing 1.1) with the given usage environment. A track is a valid candidate if both of the following conditions are met:

– the coding format of the track is supported by the usage environment, or the usage environment has not specified a coding format;
– the coding profile of the track is supported by the usage environment, or the usage environment has not specified a coding profile.

If the above conditions are met for a particular track, we promote the track to the `:PossibleTrack` class. Applying these rules to our example result in three possible tracks (i.e., track 1, 2, and 3). Track 4 is not a valid track due to the fact that `nf:VORBIS` is not supported in the given usage environment (Listing 1.3).

In the last step, the `:PossibleTracks` are linked with the `:EmptyDeliveryConfigurations` based on supported coding formats. This allows us to create a list of `:PossibleDeliveryConfigurations`. Each PDC will contain either both a video and an audio track or just one audio track. In our example, we can construct 6 PDCs:

(1) `nss:HTTP_MP4` with track 1 and 3;
(2) `nss:HTTP_MP4` with track 2 and 3;
(3) `nss:HTTP_MP4` with track 3;
(4) `nss:HAS_M3U8` with track 1 and 3;
(5) `nss:HAS_M3U8` with track 2 and 3;
(6) `nss:HAS_M3U8` with track 3.

5.2 Ranking Possible Delivery Configurations

Given a list of PDCs, we now want to rank this list in such a way that the highest ranked PDC offers the best (estimated) QoE in the given usage environment. Therefore, we consider the remaining properties provided by the usage environment as soft constraints (i.e., everything except information regarding media formats and protocols). In order to estimate the QoE of a certain

Listing 1.5. Passing properties and parameters to the rating calculation

```
{
  ?pdc a :PossibleDeliveryConfiguration.
  ?pdc :PDCBitrate ?br.
  ?context a ue:UsageEnvironment.
  ?context ue:availableBandwidth ?bw.
  _:l :maxBitrateWeight ?weight.
  _:l :maxBitrateAffinity1 ?a1.
  _:l :maxBitrateAffinity2 ?a2.
}
=>
{?pdc :calcRating (?br ?bw ?weight ?a1 ?a2).}.
```

Listing 1.6. Rating function expressed in N3Logic (for α_1)

```
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
{
  ?t :calcRating1 (?a ?b ?weight ?a1).
  (?a ?b) math:difference ?diff1.
  ?diff1 math:absoluteValue ?diff.
  (?diff ?a1) math:exponentiation ?exp.
  (-1.0 ?exp) math:product ?prod.
  (2.71828183 ?prod) math:exponentiation ?unwrating.
  (?unwrating ?weight) math:product ?rating.
}
=>
{?t :rating ?rating.}.
```

PDC, we calculate the distance between a PDC and the usage environment. More specifically, we introduce a rating function for each combination of two corresponding properties (e.g., screen size vs. video resolution, bit rate vs. bandwidth, etc.). Given a and b two corresponding properties, the rating $r_{a,b}$ for properties a and b corresponds to the following formula:

$$r_{a,b} = e^{-|a-b|^{\alpha_1}} \text{ if } a \geq b$$
$$r_{a,b} = e^{-|a-b|^{\alpha_2}} \text{ if } a < b$$

$r_{a,b}$ is a value between 0 and 1 with lower values for greater distances between a and b.

We introduce two affinity parameters in the rating function, enabling the specification of the affinity of the differences in both directions. More specifically, the parameters α_1 and α_2 denote to which extent $r_{a,b}$ drops when the difference between a and b becomes higher. The value of α_i lies between 0 and ∞; the higher the value for α_i, the faster the $r_{a,b}$ drops. Note that we use two αs to indicate the affinity towards the direction of the difference. If α_1 is greater than α_2, $r_{a,b}$ will drop less faster for $a < b$ than for $a \geq b$. Equal values for α_1 and α_2 denote no preference towards either direction.

Thus, for each combination of two corresponding properties, two affinity parameters can be specified (one for each direction). The overal rating of the PDC is then the weighted sum of the ratings of the individual property combinations. Note that the rating for each PDC seperatly has no actual meaning on its own but is only used for sorting the PDCs according to rating values.

Listing 1.5 illustrates how the property combination `:PDCBitrate` and `ue:availableBandwidth` is passed to the rating calculation algorithm, using

N3Logic. Next to these two properties, a weight factor for this property combination is provided, together with the two affinity parameters (i.e., α_1 and α_2). Additionally, Listing 1.6 shows how the first part of the rating function (i.e., for the case $a < b$) is implemented in N3Logic.

The weight factor denotes the weight of this property in the overal ranking. The higher the weight, the more this property will have an influence in the total score of the PDC. The affinity parameters are used to indicate the penalty of moving further away from the target value. For example, we can allow the bit rate to differ more from its target value (i.e., the available bandwidth) than the frame rate by setting α_i for bit rate lower than α_i for frame rate. Within one property combination (e.g., bit rate), we can specify that lower bit rates than the target bit rate are penalized less than the case when the bit rate is higher than the target bit rate. This is done by setting α_1 higher than α_2. Additionally, if we do not allow that a certain property is higher than its target value (i.e., simulating a hard constraint), α_1 must be much higher than α_2.

Using the weight and affinity parameters, we can tweak the MDTE, based on the target application. In our implementation (see also Sect. 6), we have chosen a fixed weight of 100 for all properties and α_i values between 0.1 and 0.001. However, future work could consist of introducing a self-learning system that tweaks these parameters, based on the received feedback from the usage environment.

Applying our rating calculation algorithm to the list of PDCs that we obtained in Sect. 5.1, we can calculate the estimated QoE for each PDC. As an example, we consider the `ue:maxFrameWidth` and `ma:frameWidth` property combination. Suppose we have $\alpha_1 = 0.1$ and $\alpha_2 = 0.01$ for this parameter combination, which denotes that we do not like to go over the targeted frame width, the following ratings are obtained for each PDC (for our parameter combination only):

– PDCs (1) and (4): 0.3492;
– PDCs (2) and (5): 0.1899;
– PDCs (3) and (6): 0.

Note that PDCs (3) and (6) do not get a rating since the `ue:maxFrameWidth` property is not applicable for those PDCs. If we apply the same rating calculation algorithm for all properties and subsequently take a weighted sum of these property ratings, PDC (4) will get the highest rating.

6 Implementation in a Media Delivery Platform

We integrated the MDTE as described in the previous sections in NinSuna[6], our fully integrated media adaptation and delivery platform. At its core, format-independent modules for temporal selection and packaging of media content are present. A tight coupling exists between these core modules and a model for describing structural, semantic, and scalability information of media resources. Media resources are ingested into the platform and mapped to this model. The adaptation and selection operations are based on this model and are thus independent of the underlying media formats, making NinSuna a format-independent media delivery platform. Furthermore, it enables enhanced support for media on the Web based on the following features:

– popular media formats (MP4, WebM, Ogg, MPEG-2 TS) and delivery protocols (RTSP, RTMP, HTTP progressive download, HTTP adaptive streaming) on the Web are supported;
– the Media Fragments URI 1.0 protocol [21] is implemented, which enables the delivery of media fragments (i.e., temporal or track fragments) in a standardized way;
– media metadata is published according to the Ontology for Media Resources 1.0 [13], a standardized way to represent basic information of media resources on the Web.

A detailed explanation of NinSuna and its core technologies can be found in [22] and [23].

Next, we elaborate on how the context information is collected. Subsequently, the workflow is described for the delivery of a media resource on the NinSuna platform using the MDTE.

6.1 Context Collection

Similarly to the description of the models for context information (Sect. 4), we distinguish between server capabilities, media characteristics, and usage environment.

6.1.1 Server Capabilities The server capabilities are static for one specific media server, in our case NinSuna. Hence, we only need to describe the server capabilities once and adjust them if the server is extended with a new protocol or media format for example. Currently, NinSuna supports HTTP, RTSP, RTMP, and HTTP Live Streaming (which is a form of HTTP adaptive streaming) as media delivery protocols; MP4 (with H.264/AVC, MPEG-4 Visual, AAC, MP3), WebM (with VP8, Ogg Vorbis), MPEG-2 TS (with H.264/AVC, AAC), JPG, and PNG are supported media format combinations.

6.1.2 Media Characteristics Before a media resource becomes available within the NinSuna platform, it needs to be ingested. During this ingest process, information regarding the media characteristics is collected. For instance, based on the available headers present in the source media that is ingested, technical properties

[6] `http://ninsuna.elis.ugent.be`

Listing 1.7. Representing characteristics of scalable media resources

```
<http://ninsuna/svc-test>
  a ma:MediaResource;
  ma:hasTrack :1 , :2;
  ma:numberOfVideoTracks "2"^^xsd:int;
  ma:locator <http://ninsuna/svc-test.mp4?track=1>;
  ma:locator <http://ninsuna/svc-test.mp4?track=2>.

:1  a ma:VideoTrack;
  ma:frameHeight "480"^^xsd:int;
  ma:frameWidth "640"^^xsd:int;
  ma:frameRate "25.0"^^xsd:double;
  ma:title "1";
  ma:format nf:H264;
  nsa:codingProfile nf:AVC_BASELINE;
  ma:averageBitrate "817.7"^^xsd:double;
  ma:duration "15.0"^^xsd:double.

:2  a ma:VideoTrack;
  ma:frameHeight "480"^^xsd:int;
  ma:frameWidth "640"^^xsd:int;
  ma:frameRate "12.5"^^xsd:double;
  ma:title "2";
  ma:format nf:H264;
  nsa:codingProfile nf:AVC_BASELINE;
  ma:averageBitrate "628.1"^^xsd:double;
  ma:duration "15.0"^^xsd:double.
```

such as codec, resolution, and frame rate can be extracted. Moreover, the media is organized in terms of physical and logical media resources, as described in Sect. 4.1. Therefore, instances of the media characteristics are automatically generated, compliant to the Ontology for Media Resources.

When different versions of a media resource correspond to different layers/views of a scalable/multiview media resource, these versions cannot be described by the Ontology for Media Resources (at first sight). However, it should be noted that scalability layers and alternative views are very similar to tracks; the only difference is that the former can be dependent on other layers/views while this is not the case for the latter. Thus, if these layers/views are identifyable, track fragments could be used to address them. Listing 1.7 illustrates the description of the characteristics of a media resource containing two temporal scalability layers (i.e., 25 fps and 12.5 fps). Each scalability layer corresponds to a track. Of course, this only works if the server knows the mapping between tracks and scalability layers, which is the case with NinSuna.

6.1.3 Usage Environment The usage environment can be divided into three categories: device capabilities, network characteristics, and user characteristics, as described in Sect. 4.3. It should be noted that we collect the information of the usage environment at the server, hereby allowing every existing media player. This means that collecting information of the usage environment solely relies on what you can get from a normal HTTP request. Therefore, we currently have no provisions for transparantly collecting information about network and user characteristics. To overcome this, we introduced an ad-hoc solution in the sense that we allow URI query parameters and map them to the properties of the usage environment (e.g., `http://foo.com/media?maxBandwidth=1000` is mapped to the `ue:availableBandwidth` property). This way, the responsibility for providing this context information is handed over to the user agent or a proxy in the network.

In contrast to network and usage characteristics, device capabilities can be easily detected at the server, based on a normal HTTP request. Device detection tools such as Wireless Universal Resource FiLe (WURLF, [1]) or the User Agent Profile (UAProf, [27]) are capable of recognizing devices based on HTTP headers. In this paper, we use WURFL as device detection tool on our media delivery platform. WURFL relies on the HTTP User-Agent request header [6] to determine the device. Based on a database that maps user agent strings to device capabilities, we are able to detect properties such as screen size, supported media formats, and supported codec profiles. This way, we are able to populate our model for device capabilities. An example of a User Agent string for a Google Nexus S phone is depicted on line 4 of Listing 1.8.

Currently, WURFL provides the database in the form of an XML document, following a proprietary structure. We created a converter taking as input this XML document and producing an RDF document describing the devices and their capabilities according to the model presented in Sect. 4.3.1. This way, we obtain a filtered version of WURFL's database, only containing the properties that we are interested in. Additionally, we enhance our database with more detailed and up-to-date information regarding supported media formats, profiles, and protocols. These updates are represented in the form of SPARQL 1.1 UPDATE statements [7], enabling a flexibel update mechanism when a new version of the WURFL database is released. In an ideal scenario, a formalized data set of device capabilities would be available in the Linked Open Data cloud[7], where everyone can contribute device information.

6.2 Integration into the Web Infrastructure

The integration of our MDTE within NinSuna can be tested at `http://ninsuna.elis.ugent.be/Media`, which is the media repository of NinSuna. In this subsection, we illustrate how the NinSuna platform is steered by the MDTE, by means of describing a workflow of different scenarios user agents could follow.

Accessing the metadata[8] of a NinSuna media resource is already steered by the MDTE. More specifically, based on the hard constraints applying to the usage environment that is trying to access the media resource, only the PDCs are shown to the user. For instance, accessing the metadata page of a media resource on NinSuna with

[7] `http://linkeddata.org/`

[8] By writing text/html, application/rdf+xml, or text/turtle in the HTTP Accept request header.

Listing 1.8. Redirecting devices to the best fitted version

```
GET /Media/test HTTP/1.1
Host: ninsuna.elis.ugent.be
Accept: video/*
User-Agent: Mozilla/5.0 (Linux; U; Android 2.3.4; de-
     de; Nexus S Build/GRJ22) AppleWebKit/533.1 (
     KHTML, like Gecko) Version/4.0 Mobile Safari
     /533.1

HTTP/1.1 307 Temporary Redirect
Server: NinSuna/2.0
Location: /Media/test.mp4?track=1;3
Content-Length: 0
Content-Type: video/mp4
```

an iPhone will not show any RTSP or RTMP links, because these protocols are not supported on the iPhone. The rating for each PDC is also shown and is calculated according to the algorithm explained in Sect. 5.2.

Directly accessing the best fitted version of a media resource[9] is also possible. An example of such a request is shown in Listing 1.8. The MDTE, deployed on the NinSuna platform, picks the highest ranked PDC and redirects the user agent to that PDC. In this example, the user agent is redirected to an MP4 containing tracks 1 and 3 and delivered over HTTP.

Note that for HTTP adaptive streaming protocols (such as Apple's HTTP Live Streaming), the MDTE does not pick the highest ranked PDC. Instead, all PDCs are described within the manifest, according to the specifications of the HTTP adaptive streaming protocol. This way, a user agent can seamlessly switch between the various PDCs, according to the varying usage environment conditions.

7 Conclusions and Future Work

In order to anticipate the huge diversity in usage environments on the one hand and the multiple possibilities for media delivery on the other hand, we introduced a formalized, semantic media decision taking engine. The latter is capable of ranking possible media delivery configurations, based on context information consisting of characteristics of the requested media, server capabilities, and the usage environment. The MDTE is fully driven by formalized rules, expressed in N3Logic, applying hard constraints to exclude PDCs and soft constraints to rank PDCs. The ranking algorithm is based on a ranking function with support for weights and affinity parameters for each property combination. Finally, we illustrated how the MDTE can be deployed inside NinSuna, a media delivery server targeted at media on the Web.

Future work consists of investigating other places within the network to deploy the MDTE. For instance, deploying the MDTE in a user agent would result in an easier and more detailed collection of context information about the usage environment. The MDTE could be deployed within user agents supporting HTTP adaptive streaming in order to decide when to switch from one quality version to another. Other future work consists of introducing a self-learning component inside the MDTE. More specifically, the MDTE must be able to learn the weight and affinity parameters by itself, based on feedback from the user agent. This way, its decisions would gradually be improved.

[9] By writing video/* in the HTTP Accept request header.

8 Acknowledgments

The research activities as described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

1. http://wurfl.sourceforge.net/
2. Eye deep taxonomy benchmark results, http://eulersharp.sourceforge.net/2003/03swap/dtb-2010.txt
3. Berners-lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3Logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming 8(3), 249–269 (2008)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34 (2001)
5. De Roo, J.: Euler proof mechanism, http://eulersharp.sourceforge.net/
6. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: RFC 2616: "Hypertext Transfer Protocol – HTTP/1.1," Available on http://www.w3.org/Protocols/rfc2616/rfc2616.html.
7. Gearon, P., Passant, A., Polleres, A. (eds.): SPARQL 1.1 Update. W3C Working Draft, World Wide Web Consortium (May 2011)
8. Herranz, L.: Integrating semantic analysis and scalable video coding for efficient content-based adaptation. Multimedia Systems 13, 103–118 (2007)
9. Hutter, A., Amon, P., Panis, G., Delfosse, E., Ransburg, M., Hellwagner, H.: Automatic adaptation of streaming multimedia content in a dynamic and distributed environment. In: Proceedings of IEEE International Conference on Image Processing (ICIP 2005) (September 2005)
10. ISO/IEC: 21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation (October 2004)
11. Klyne, G., Carrol, J.J.: Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation (Feb 2004), http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
12. Köhncke, B., Balke, W.T.: Preference-driven personalization for flexible digital item adaptation. Multimedia Systems 13, 119–130 (2007)
13. Lee, W., Bailer, W., Bürger, T., Malaisé, V., Michel, T., Sasaki, F., Söderberg, J., Stegmaier, F., Strassner, J. (eds.): Ontology for Media Resources 1.0. W3C Working Draft, World Wide Web Consortium (March 2011)

14. López, F., Martínez, J.M., García, N.: A model for preference-driven multimedia adaptation decision-making in the MPEG-21 framework. Multimedia Tools and Applications 53, 181–211 (May 2011)
15. Lum, W.Y., Lau, F.C.M.: A Context-Aware Decision Engine for Content Adaptation. IEEE Pervasive Computing 1, 41–49 (July 2002)
16. McGuinness, D., van Harmelen, F. (eds.): OWL Web Ontology Language: Overview. W3C Recommendation, World Wide Web Consortium (February 2004), http://www.w3.org/TR/owl-features/
17. Mukherjee, D., Delfosse, E., Kim, J.G., Wang, Y.: Optimal Adaptation Decision-taking for Terminal and Network Quality-of-service. IEEE Transactions on Multimedia 7(3), 454–462 (June 2005)
18. Osmun, T.: Euler Eye installation, demo, and deep taxonomy benchmark, http://ruleml.org/WellnessRules/files/WellnessRulesN3-2009-11-10.pdf
19. Prangl, M., Szkaliczki, T., Hellwagner, H.: A Framework for Utility-based Multimedia Adaptation. IEEE Transactions on Circuits and Systems for Video Technology 17(6), 719–728 (June 2007)
20. Szwabe, A., Schorr, A., Hauck, J., Kassler, A.: Dynamic Multimedia Stream Adaptation and Rate Control for Heterogeneous Networks. Journal of Zhejiang University SCIENCE A 7, 63–69 (2006)
21. Troncy, R., Mannens, E., Pfeiffer, S., Van Deursen, D. (eds.): Media Fragments URI 1.0. W3C Working Draft, World Wide Web Consortium (March 2011)
22. Van Deursen, D., Van Lancker, W., De Bruyne, S., De Neve, W., Mannens, E., Van de Walle, R.: Format-independent and Metadata-driven Media Resource Adaptation using Semantic Web Technologies. Multimedia Systems 16(2), 85–104 (2010)
23. Van Deursen, D., Van Lancker, W., De Neve, W., Paridaens, T., Mannens, E., Van de Walle, R.: NinSuna: a Fully Integrated Platform for Format-independent Multimedia Content Adaptation and Delivery based on Semantic Web Technologies. Multimedia Tools and Applications – Special Issue on Data Semantics for Multimedia Systems 46(2-3), 371–398 (January 2010)
24. Van Deursen, D., Van Lancker, W., Van de Walle, R.: On Media Delivery Protocols in the Web. In: Proceedings of the IEEE International Conference on Multimedia and Expo 2010. pp. 1028–1033. Singapore (July 2010)
25. Van Lancker, W., Van Deursen, D., Mannens, E., Van de Walle, R.: Harmonizing Media Annotations and Media Fragments, submitted to the 2011 Workshop on Multimedia on the Web
26. Vetro, A., Christopoulos, C., Ebrahimi, T.: Universal Multimedia Access. IEEE Signal Processing Magazine 20(2), 16 (March 2003)
27. Wireless Application Protocol Forum: UAProf User Agent Profiling Specification (2001)