Detecteren, representeren en uitbuiten van de mogelijkheden van het apparaat van de eind-gebruiker

Tomas Creemers

promotor :
Prof. dr. Wim LAMOTTE

Universiteit Maastricht

Eindverhandeling voorgedragen tot het bekomen van de graad master in de informatica multimedia

universiteit hasselt

Voorwoord

Geen thesis zonder voorwoord, en dit werk is daar dan ook geen uitzondering op. De totstandkoming van deze thesis kan zonder aarzelen een moeizaam proces genoemd worden, maar het was evenzeer een leerrijke ervaring. En dat was niet mogelijk geweest zonder een hele boel mensen. Niet alleen als bladvulling, maar ook uit oprechte dankbaarheid, wens ik er dan ook enkele te vernoemen. Vooreerst mijn thesisbegeleider, Maarten Wijnants, voor zijn onvermoeibaarheid bij het telkens opnieuw nalezen van deze tekst, voor zijn grondige kennis van de Nederlandse taal die met momenten een geschenk uit de hemel leek, en voor zijn hulp bij, en kennis van, de implementatie. Natuurlijk ook mijn promotor, prof. dr. Wim Lamotte, alleen al omdat er zonder hem geen thesis was geweest. Verder ook mijn medestudenten, die het leven zowel tijdens als na de lesuren net dat tikkeltje interessanter maken. *Last, but definitely not least,* mijn ouders. Ik kan alleen maar hopen dat zij aan mij ooit het geheim achter hun onuitputtelijke voorraad steun en geduld zullen onthullen.

Inhoudsopgave

Voorwoord **i**

1 Inleiding **1**

2 Detectie **3**
2.1 Hardware . . . 4
2.1.1 Rechtstreeks . . . 4
2.1.2 Via het besturingssysteem . . . 6
2.1.3 Hardware Abstraction Layer . . . 7
2.2 Software . . . 9

3 Representatie **12**
3.1 Composite Capability/Preference Profiles . . . 12
3.1.1 World Wide Web Consortium . . . 12
3.1.2 Achtergrond . . . 13
3.1.3 Resource Description Framework . . . 14
3.1.4 Opbouw van CC/PP . . . 16
3.1.5 Voorbeeldvocabularium: UAProf . . . 17
3.1.6 CC/PP-uitwisseling via HTTP-headers . . . 18
3.2 MPEG-21 . . . 23
3.2.1 MPEG . . . 23
3.2.2 MPEG-21 . . . 25
3.2.3 MPEG-21 Part 7 (“Digital Item Adaptation”) . . . 26
3.2.4 MPEG-21: Part 7: Usage Environment Description Tools . . . 28
3.2.5 Gebruikerskarakteristieken . . . 28
3.2.6 Apparaatmogelijkheden . . . 36
3.2.7 Netwerkkarakteristieken . . . 41
3.2.8 Omgevingskarakteristieken . . . 44
3.3 Zelf-gedefinieerd formaat . . . 46
3.3.1 Binaire formaten . . . 46

3.3.2 Tekstuele formaten . . . 46
3.4 eXtensible Markup Language . . . 47
3.5 Compressie . . . 48
3.5.1 Binaire XML volgens het W3C . . . 48
3.5.2 Algemene compressie . . . 49
3.5.3 Binaire XML-formaten . . . 49
3.5.4 Schema-bewuste XML-compressie . . . 49
3.5.5 Resultaten . . . 51
3.5.6 Conclusie . . . 51
3.6 Vergelijking van representatieformaten . . . 52
3.6.1 Zelf definiëren of externe standaard . . . 52
3.6.2 Verkrijgbaarheid . . . 52
3.6.3 Uitbreidbaarheid . . . 53
3.6.4 Toepassingsgebied . . . 53

4 Uitbuiting **54**
4.1 Doelen . . . 54
4.1.1 Universal Multimedia Access . . . 54
4.2 Locatie en methode . . . 55
4.2.1 Lokaal . . . 55
4.2.2 Op afstand . . . 55
4.3 Maatstaven . . . 56
4.3.1 Quality of Service . . . 56
4.3.2 Quality of Experience . . . 57

5 Bestaande systemen **59**
5.1 "Applying CC/PP to User's Environmental Information for Web Service Customization" . . . 59
5.2 Adactus Mobilize . . . 60
5.2.1 Adactus . . . 60
5.2.2 Mobilize . . . 60
5.3 Network-integrated Multimedia Middleware . . . 61
5.3.1 Communicatie . . . 62
5.3.2 Architectuur . . . 62
5.4 NIProxy . . . 63
5.4.1 Architectuur . . . 63
5.4.2 Detectie . . . 64
5.4.3 Interne communicatie . . . 65
5.4.4 Uitbuiting . . . 65

6 Implementatie **66**
6.1 Client 67
6.1.1 Detectie ("libued") 67
6.1.2 Graphical User Interface 68
6.1.3 Connectie 72
6.2 ClientInfoManager 75
6.2.1 Ontleding 75
6.2.2 Opslag 75
6.2.3 Toegang 78
6.3 Transformer 79
6.4 Transmitter 80
6.5 Resultaten 81

7 Conclusie **82**

Lijst van figuren

3.1 Een eenvoudig voorbeeld van RDF, weergegeven in diagramvorm 15
3.2 Schematische weergave van een *profile diff* 20
3.3 Schematische weergave van een *remote profile* 21
3.4 Schematische weergave van een gecached *remote profile* 21
3.5 Schematische weergave van een profiel-bewuste proxy 23
3.6 De schematische structuur van een Digital Item Adaptation Engine 28

5.1 Schematische weergave van de NIProxy-architectuur 64

6.1 Tabblad van de NIProxy-client GUI waar "terminal capabilities" geconfigureerd kunnen worden 73
6.2 Tabblad van de NIProxy-client GUI waar connectie-informatie opgegeven kan worden en de ontvangen data weergegeven wordt 74
6.3 Excerpt uit de inauguratiespeech van Barack Obama 81
6.4 Bewerkt excerpt uit de inauguratiespeech van Barack Obama 81

Listings

2.1 Programma voor detectie van MMX-ondersteuning van de CPU 5
2.2 Programma voor bepalen van het aantal processors 6
2.3 Programma dat bij compilatie bepaalt wat het huidige besturingssysteem is . 9
2.4 Programma voor bepalen van de schermresolutie onder MS Windows 10
2.5 Programma voor bepalen van de schermresolutie van X 10
3.1 Een eenvoudig voorbeeld van RDF, weergegeven in XML 15
3.2 Een kort voorbeeld van een CC/PP-profiel 18
3.3 Syntax van de `Opt`-header 19
3.4 Syntax van de `Profile`-header 22
3.5 Syntax van de `Profile-Diff`-header 22
3.6 Een HTTP-verzoek dat een profielverwijzing en een profielstuk bevat 22
3.7 Een voorbeeld van UED-eigenschappen uit de categorie User Characteristics . 33
3.8 Een voorbeeld van UED-eigenschappen uit de categorie Terminal Capabilities 40
3.9 Een voorbeeld van UED-eigenschappen uit de categorie Network Characteristics 43
3.10 Een voorbeeld van UED-eigenschappen uit de categorie Natural Environment Characteristics 45
6.1 Prototype van functie voor het opvragen van de schermbreedte 67
6.2 Prototype van functie voor het opvragen van een compleet MPEG-21 UED-profiel 67
6.3 Code voor het met TinyXML opstellen van een MPEG-21 UED-profiel dat schermeigenschappen bevat 69
6.4 Code voor het met QXmlStreamWriter opstellen van een MPEG-21 UED-profiel dat schermeigenschappen bevat 71
6.5 MPEG-21 UED geproduceerd door de code in listing 6.4 72
6.6 Prototype van methode die profieldata in de `ClientInfoManager` invoert . . 75
6.7 Code voor opzoeken van de schermbreedte in een MPEG-21 UED-profiel met behulp van XQilla 76
6.8 Prototype van functie voor het registreren van een XPath-expressie bij de `ClientInfoManager` 76
6.9 SQL-expressie die de databasetabel aanmaakt voor client met clientID 52 . . 78
6.10 SQL-expressie die eigenschappen invoegt van client met clientID 52 78

6.11 Prototype van functie voor het opvragen van clienteigenschappen 79
6.12 SQL-expressie die een eigenschap ophaalt uit de databasetabel van client met clientID 52 79
6.13 Registratie van een XPath-expressie onder de naam “parentalrating” bij de `ClientInfoManager` 80
6.14 Opvragen van de eigenschap met naam “parentalrating” voor een bepaalde client uit de `ClientInfoManager` 80

Hoofdstuk 1

Inleiding

Het Internet zoals we dat nu kennen, is geëvolueerd uit de koppeling van verschillende kleinere netwerken tot een internetwerk, wat meteen ook de naam verklaart. De ontwikkeling ervan begon in de jaren '60 [1] en het telde twintig jaar later een tweehonderdtal aangesloten computers [2]. Weinigen hadden toen al kunnen voorspellen dat het in 2008 een wereldwijd netwerk zou worden dat de kaap van een miljard gebruikers ruim overschreden zou hebben [3, 4]. Deze gebruikers pompen momenteel gezamenlijk meer dan 300 000 000 000 000 000 bytes per dag de wereld rond. Voor de nabije toekomst verwacht men dat dit volume elk jaar meer dan anderhalve keer zo groot zal zijn [5].

De apparaten die gebruikt worden om toegang te krijgen tot het Internet zijn ook sterk toegenomen in diversiteit; ze verschillen onderling sterk in vele eigenschappen. De schermgrootte, bijvoorbeeld, kan variëren van een breedbeeldtelevisie van een vierkante meter, tot een monochroom GSM-scherm van enkele vierkante centimeters groot. Er worden per type apparaat vaak andere eisen gesteld aan het gebruik van onder andere rekenkracht, geheugen, opslagcapaciteit, stroomverbruik en vele andere bronnen. Men mag dus wel stellen dat er geen sprake is van een uniforme gebruikersterminal.

Ook de verbinding tussen deze apparaten en het Internet kan gemaakt worden met behulp van een hele hoop verschillende methodes. Elke methode heeft verschillende, soms typerende, karakteristieken. Deze kunnen vaak ook fluctueren over de tijd. Zo heeft satellietinternet traditioneel last van een grote vertraging, wegens de grote afstand die het signaal moet afleggen. Mobiele internetverbindingen hebben doorgaans een hoge kost per data-eenheid, en lopen achter qua doorvoersnelheid. Steeds nieuwere technologieën maken echter steeds degelijkere snelheden beschikbaar.

Dit alles heeft het digitale landschap ingrijpend veranderd. Niemand kijkt nog op wanneer iemand met zijn mobiele telefoon een foto maakt en doorstuurt, of streaming video kan kijken op een PDA. Multimedia is steeds wijder beschikbaar.

Zowel zakelijke als thuisgebruikers verwachten steeds meer dat ze deze multimedia kunnen bekijken op eender welke combinatie van netwerkverbinding en apparaat, volledig aangepast aan hun wensen. Dit is echter niet evident om te realiseren.

Een GSM heeft vaak een kleinere schermresolutie dan een gemiddeld werkstation, waardoor een video met grotere afmetingen naar de GSM doorsturen voor het bekijken een verspilling van zowel netwerkbronnen als rekenkracht (voor het decoderen en schaleren) inhoudt. Aan de andere kant kan een GSM beschikken over een GPS-chipset waardoor bijvoorbeeld locatiegebonden nieuwsberichten automatisch gekozen kunnen worden. Dit toont aan dat het niet enkel een kwestie is van "graceful degradation" over een absolute rangorde van apparaten, hoewel dat zeker deel uitmaakt van wat er zoal gedaan kan worden. Er dient rekening gehouden te worden met de mogelijkheden van iedere aparte klasse van apparaten en netwerken.

In hoofdstuk 2 worden verschillende methodes van aanpak voor het detecteren van apparaatmogelijkheden besproken. Voor het communiceren van de gedetecteerde data met een extern systeem moet deze data, samen met de eventuele voorkeuren van de gebruiker, voorgesteld worden in een formaat dat zowel door de zender als de ontvanger begrepen wordt. Hoofdstuk 3 behandelt enkele formaten om deze gedetecteerde mogelijkheden en gebruikersvoorkeuren te representeren. Wat er na de detectie en eventuele communicatie zoal gedaan kan worden met de verkregen informatie, wordt nagegaan in hoofdstuk 4. Een kijk op hoe enkele reeds bestaande systemen één of meerdere van de voorgaande stappen implementeren, wordt in hoofdstuk 5 gegeven. De implementatie die bij dit werk hoort, wordt besproken in hoofdstuk 6. Ten slotte bevat hoofdstuk 7 de conclusie.

Hoofdstuk 2

Detectie

De mogelijkheden van een apparaat zijn geen gegeven informatie, maar moeten achterhaald worden. Een gebruikersapparaat kan een enorme hoeveelheid eigenschappen hebben. Bij uitbreiding kan men ook geïnteresseerd zijn in informatie omtrent de huidige fysieke omgeving van het apparaat, bijvoorbeeld de positie van de gebruiker ten opzichte van het apparaat, of de hoeveelheid achtergrondgeluid.

Sommige eigenschappen zijn makkelijk te achterhalen; andere eigenschappen zijn dan weer nagenoeg onmogelijk te bepalen, zoals de maximale helderheid van een LCD-backlight na drie jaar gebruik.

Er zijn ook eigenschappen die niet noodzakelijk door het apparaat zelf achterhaald moeten worden. Wanneer het apparaat een IP-adres heeft, kan men dit traceren tot een zeer ruw geschatte locatie [6]. Ook netwerkeigenschappen zoals de vertraging, *packet loss rate* en bandbreedte kunnen in samenwerking met een ander apparaat bepaald worden. Een voorbeeld van een systeem dat dit doet is de NIProxy [7], die in sectie 5.4 besproken wordt.

Een situatie die ook kan voorkomen is wanneer het om een standaardapparaat gaat, waarvan een groot deel van de eigenschappen vast ligt. Fabrikanten van GSM's en PDA's stellen soms een profiel op voor hun apparaten, dat toegankelijk is via het Internet. Dit elimineert de noodzaak van detectie van (een deel van) de eigenschappen van het apparaat in kwestie, aangezien deze reeds aangegeven worden in het profiel. Een voorbeeld van zo'n profiel is te vinden in [8]. Helaas heeft een generieke applicatie geen manier om te bepalen of het met zulk een apparaat te maken heeft, tenzij het zelf een lijst ervan zou bijhouden. Bijkomend kan de applicatie niet weten waar het eventuele profiel zich zou bevinden.

Tot op zekere hoogte bestaat er ook een andere bron van informatie: de gebruiker. De inhoud die gepresenteerd wordt op een gebruikersapparaat dient uiteindelijk voor consumptie door de gebruiker. De informatie die de gebruiker eventueel verstrekt mag niet onderschat

worden, want het is de ervaring voor de gebruiker van de uitvoer (beeld, geluid, ...) die gemaximaliseerd moet worden. Deze ervaring is immers niet voor iedere gebruiker hetzelfde. Voor de hand liggende voorbeelden hiervan zijn hardhorigen, slechtzienden, ... Daarom kan men de eindgebruiker de bepalende factor laten zijn bij sommige eigenschappen (voornamelijk subjectieve), zoals de helderheid van het beeld en de geluidssterkte.

Helaas kan men niet verwachten dat de gebruiker alle details van de hard- en software van zijn apparaat kent. Bovendien is het niet opportuun om de gebruiker lastig te vallen met zulke vragen als de informatie ook op andere manieren vergaard kan worden.

Zonder gebruik te maken van software die dit werk zelf afhandelt, kan het achterhalen van de relevante eigenschappen doorgaans niet op één uniforme manier, onder meer doordat ze betrekking kunnen hebben op heel uiteenlopende delen van het gebruikersapparaat. Een softwarelibrary die poogt deze eigenschappen te verzamelen uit variërende bronnen en vervolgens aan applicaties aanbiedt, wordt besproken in sectie 2.1.3. Ondanks de diversiteit van de eigenschappen en de bronnen ervan, onderscheiden we twee grote hoofdcategorieën: eigenschappen met betrekking tot de software enerzijds, en eigenschappen met betrekking tot de hardware anderzijds. Dit is natuurlijk een ruwe indeling en de categorieën omvatten samen niet noodzakelijk alle mogelijke eigenschappen (zoals achtergrondgeluid, ...). Toch is dit een handige onderverdeling om detectiemethodes te bespreken. Ook komen deze categorieën terug in vele representatiemethodes (meer hierover in hoofdstuk 3).

2.1 Hardware

2.1.1 Rechtstreeks

De applicatie kan, mits het besturingssysteem dit toelaat, rechtstreeks communiceren met de hardware. De gebruikelijke interfaces van het besturingssysteem worden dan niet gebruikt; de 'communicatie' gebeurt door instructies rechtstreeks naar de hardware te sturen, of de instructies erop uit te voeren (indien het om hardware gaat die in staat is instructies uit te voeren).

Dit kan soms op een gemakkelijke manier gedetailleerde en bruikbare informatie opleveren. Als voorbeeld tonen we in listing 2.1 een miniscuul programma geschreven in C en x86 assembly, dat bepaalt of de CPU waar het op wordt uitgevoerd ondersteuning biedt voor MultiMedia eXtensions (MMX) [9, 10]. Eerst wordt waarde 1 in register `eax` geplaatst door middel van de `mov`-instructie. Deze staat daar als parameter voor de `CPUID`-instructie. Deze verwacht een getal in register eax als argument om te selecteren welke informatie over de CPU het dient terug te geven — zo staat 0 voor de naam van de fabrikant en 1 voor informatie over de CPU en een deel van de capaciteiten ervan. De gevraagde informatie wordt in de registers `eax`, `ebx`,

Listing 2.1: Programma voor detectie van MMX-ondersteuning van de CPU

```
1  #include <stdio.h>
2
3  int main( void ) {
4
5    int cpuFlags;
6
7    __asm__(
8      "mov $1, %%eax;"
9      "cpuid;"
10     : "=d" ( cpuFlags )
11     :
12     : "%eax", "%ebx", "%ecx"
13   );
14
15   /* Bit 23 geeft MMX-ondersteuning aan. */
16   if ( cpuFlags & ( 1 << 23 ) )
17     printf( "Deze CPU ondersteunt MMX.\n" );
18   else
19     printf( "Deze CPU ondersteunt geen MMX.\n" );
20
21   return 0;
22 }
```

`ecx` en `edx` geplaatst. De "=d" (cpuFlags) geeft aan de compiler aan dat de waarde in register `edx` na uitvoeren van de assembly-code in de cpuFlags-variabele geplaatst dient te worden. De regel met "%eax", "%ebx", "%ecx" informeert de compiler dat deze niet mag aannemen dat de waardes die voordien in deze registers stonden, ongewijzigd zijn (voor register `edx` wordt dit wegens eerdere code impliciet aangenomen). Vervolgens kunnen de resultaten verwerkt worden. Bijvoorbeeld, als de CPU de MMX-instructieset ondersteunt, zal bit 23 van register edx (dat gekopieerd werd naar de variabele cpuFlags) 1 zijn [11].

Deze manier van apparaatinformatie ophalen wordt onder andere door video decoders gebruikt om te bepalen welke processorinstructies hun algoritmes mogen gebruiken. Een voorbeeld hiervan is Libavcodec [12].

Een nadeel van deze aanpak is dat ze inherent hardwarespecifiek werkt: een CPU die niet gebaseerd is op de x86-instructieset zal de `CPUID`-opcode niet kennen en de code in listing 2.1 dan ook niet kunnen uitvoeren. Ook werkt de methode enkel op x86-processoren van na ongeveer 1990 (vanaf Intel's Pentium). Voor die generatie van processoren was de `CPUID`-opcode nog niet opgenomen in de x86-instructieset. Bovendien is niet alle hardware in staat gedetailleerde informatie over zijn eigen mogelijkheden prijs te geven. Exotische, toekomstige en oude hardware vereisen ook vaak een andere aanpak.

Als voorbeeld hiervan gelden applicaties zoals CPU-Z [13] en GPU-Z [14], die zich specialiseren in het opsommen van hardware-specificaties. Wanneer een fabrikant zoals Intel [15], Nvidia [16] of AMD [17] nieuwe hardware lanceert, moeten deze applicaties vaak een nieuwe versie uitbrengen om de nieuwe hardware en alle, eventueel nieuwe, mogelijkheden correct te detecteren.

2.1.2 Via het besturingssysteem

Eén van de hoofdtaken van een besturingssysteem is om toegang tot de hardware te abstraheren tot een meer algemene interface [18]. Op deze manier kunnen applicaties gebruik maken van een hele klasse van hardware (zoals bijvoorbeeld geluidskaarten) terwijl ze de details van de aansturing van het specifieke model van de hardware (bijvoorbeeld "Realtek ALC655") overlaten aan het besturingssysteem.

Wegens hun taak is het vanzelfsprekend dat besturingssystemen veel kennis hebben over de hardware, al dan niet via stuurprogramma's. Meestal wordt deze informatie beschikbaar gesteld. Het grote nadeel van deze aanpak is natuurlijk dat ieder besturingssysteem zijn eigen manier (Application Programming Interface (API)) heeft om de informatie beschikbaar te stellen. Zo is er onder Microsoft Windows de Win32-API en biedt de Linux-kernel onder andere ondersteuning voor het "proc"-bestandssysteem waarin virtuele bestanden veel informatie bevatten. Als voorbeeld geven we een miniscuul programma (listing 2.2) geschreven in C, maar specifiek voor Microsoft Windows, om het aantal processoren in een systeem te bepalen.

Listing 2.2: Programma voor bepalen van het aantal processors

```
1  #include <windows.h>
2  #include <stdio.h>
3
4  int main( void ) {
5
6    SYSTEM_INFO siSysInfo;
7
8    GetSystemInfo( &siSysInfo );
9
10   printf( "Number of processors: %u\n",
11     siSysInfo.dwNumberOfProcessors );
12
13   return 0;
14 }
```

2.1.3 Hardware Abstraction Layer

De Hardware Abstration Layer (HAL, [19]) is een stuk software voor UNIX-desktops dat een overzicht van de hardware aangesloten op het systeem biedt aan andere applicaties. De HAL houdt ook informatie over deze apparaten bij, en kan applicaties die erom vragen op de hoogte houden van toevoegingen, verwijderingen en aanpassingen van hardware. De ontwikkeling van de software begon in 2003, na een paper [20] van Havoc Pennington die een manier voorstelde om hardware (met de nadruk op consumentenelektronica zoals digitale camera's en externe harde schijven) automatisch te detecteren, configureren en beschikbaar te stellen aan applicaties. Het werk eraan wordt nu gedragen door freedesktop.org [21].

Deze Hardware Abstraction Layer situeert zichzelf als een extra abstractielaag tussen de kernel en applicaties die de hardware willen benaderen. De HAL kan best niet verward worden met een ander soort software dat dezelfde naam draagt, maar deel is van één van de onderste lagen van de kernel zelf, en meestal niet rechtstreeks aanspreekbaar is door desktop-applicaties. Het doel van de HAL die hier besproken wordt is om *plug and play* functionaliteit van hardware mogelijk te maken op UNIX-achtige besturingssystemen. De focus ligt dus op externe, makkelijk aan en af te koppelen apparaten, maar niets verhindert de HAL om ook interne hardware te detecteren en beschrijven.

Apparaatobjecten

Hardware wordt door de HAL van freedesktop.org als een object met een uniek identificatienummer voorgesteld. Deze identificatie, *Unique Device Identifier*, wordt gegenereerd op basis van informatie van de hardware. Binnen dit object worden eigenschappen geassocieerd met het stuk hardware opgeslagen, voorgesteld door een simpele naam en de waarde van de eigenschap. Deze eigenschappen worden verzameld van verschillende bronnen. De hardware zelf kan, zoals eerder in dit hoofdstuk besproken, enige informatie bevatten. Vaak zijn dit essentiële stukken informatie, zoals de capaciteit van een harde schijf, of de status van een netwerkverbinding. De instellingen van het besturingssysteem vormen een andere bron. Dit kunnen voorkeuren zijn omtrent de acties die uitgevoerd moeten worden wanneer een specifiek apparaat of een bepaald type apparaat wordt aangesloten. Andere informatie die van het besturingssysteem kan komen is een *blacklist*, die aangeeft welke apparaten genegeerd moeten worden, bijvoorbeeld omdat het bekend is dat ze incompatibel zijn met het besturingssysteem. Ook de gebruiker kan een bron van informatie zijn. Doordat gepoogd wordt de details van de aansturing van de hardware af te schermen van de gebruiker, beperkt de informatie die die kan aandragen zich meestal tot een gekozen naam voor het apparaat of de standaard-actie (inhoud van een harde schijf tonen, foto's van een camera afladen, ...) die uitgevoerd moet worden wanneer een apparaat wordt aangesloten. Deze actie kan dus zowel door het besturingssysteem als door de gebruiker opgegeven worden. Instellingen kunnen opgeslagen

worden per specifieke instantie van het apparaat, of voor een hele klasse van apparaten. Een aantal eigenschap-namen zijn vooraf gedefinieerd door de HAL-specificatie [22].

De eigenschappen van de apparaat-objecten wordt georganiseerd in vier categorieën, die telkens onderverdeeld zijn in 'namespaces'. Deze zijn:

General properties Representeert eigenschappen die niet met fysieke of functionele eigenschappen van het apparaat te maken hebben;

Subsystem-specific properties Bevat de eigenschappen die betrekking hebben tot aanspreekbare hardware. Welke namespaces in deze categorie beschikbaar zijn, hangt af van welk (hardwarematig) subsysteem (USB, PCI, SCSI, ...) voor het apparaat gebruikt wordt. Dit wordt aangegeven in het `info.subsystem`-attribuut van de "General properties"-categorie. De eigenschappen in deze categorie zijn vooral van nut zijn voor stuurprogramma's.

Functional properties De categorie voor eigenschappen met betrekking tot de functionaliteit van apparaten. Voorbeelden hiervan zijn de partitie-indeling van een harde schijf en of er een beschrijfbare CD in een CD-schrijver zit. Deze informatie wordt meestal aangeleverd door een stuurprogramma.

Miscellaneous properties Deze categorie bestaat voor alle eigenschappen die niet in één van de vorige categorieën passen, maar bevat in de huidige specificatie van de HAL enkel de "access_control" namespace. Deze controleert welke gebruikers en gebruikersgroepen toegang krijgen tot het apparaat gerepresenteerd door het apparaat-object in kwestie.

Interfaces

De informatie in de hierboven beschreven apparaat-objecten wordt aan applicaties beschikbaar gesteld via D-Bus. D-Bus [23] is een mechanisme voor interprocescommunicatie (IPC). Het wordt net zoals HAL ontwikkeld door freedesktop.org. Zowel boodschappen met één ontvanger als een systeem dat met abonnees werkt (een *event*-systeem of "publish/suscribe communication") worden ondersteund.

Apparaatobject-eigenschappen van de HAL kunnen via D-Bus interfaces benaderd worden. Een voorbeeld van zo'n interface is de `org.freedesktop.Hal.Device.KillSwitch`-interface. Die interface kan gebruikt worden om apparaat-objecten die hardware voorstellen die als *kill switch* dienst doet, te benaderen. Een bekend voorbeeld van zulk een apparaat dat op sommige laptops wordt teruggevonden is de knop om draadloze communicatie aan en uit te zetten. De interface bevat de methode `GetPower` die 1 teruggeeft als en slechts als de kill switch uit staat (en het apparaat dat erdoor gecontroleerd wordt dus aan staat).

2.2 Software

Onder software-eigenschappen verstaan we in deze tekst informatie zoals de ondersteunde videocodecs of de versie van het besturingssysteem. Aangezien software bovenop de hardware uitgevoerd wordt, zijn software-eigenschappen vaak enkel te bepalen via een applicatie- of besturingssysteemspecifieke manier. Enkele eigenschappen kunnen al tijdens het samenstellen van een applicatie bepaald worden. Applicaties moeten vaak voor ieder besturingssysteem opnieuw gecompileerd worden, zodat men deze informatie al tijdens het compileren van de broncode kan bepalen. Listing 2.3 geeft weer hoe een applicatie die zowel onder Microsoft Windows en Linux draait, al tijdens het compileren eenvoudig kan bepalen welke situatie van toepassing is. Listing 2.4 toont een programma dat de huidige schermresolutie onder Microsoft Windows weergeeft, terwijl listing 2.5 hetzelfde doet bij X-gebruikende systemen [24]. Een schermresolutie kan ook als hardware-eigenschap gezien worden; zeker bij schermen die 'discrete' weergave van beeldelementen doen (transistors per pixel in LCD-schermen tegenover één elektronenkanon in CRT-schermen). Toch is de uiteindelijke resolutie zoals we die in listings 2.4 en 2.5 bepalen niet noodzakelijk gelijk aan de 'native' resolutie van het scherm, maar wel het aantal beeldpunten dat ter beschikking is aan de software.

Listing 2.3: Programma dat bij compilatie bepaalt wat het huidige besturingssysteem is

```
1  #include <stdio.h>
2
3  int main( void ) {
4
5    #ifdef __linux__
6      printf( "Het programma is gecompileerd onder Linux.\n" );
7    #else
8      #ifdef _WIN32
9        printf( "Het programma is gecompileerd onder Microsoft Windows.\n"
             );
10     #else
11       printf( "Het programma is niet onder MS Windows of Linux
             gecompileerd.\n" );
12     #endif
13   #endif
14
15   return 0;
16 }
```

In deze listings ziet men duidelijk dat specifieke informatie opvragen voor verschillende systemen op heel uiteenlopende manieren gebeurt. Zo kan men onder Microsoft Windows de schermresolutie te weten komen via het besturingssysteem, terwijl bij de meeste Linuxdistributies de X-server (wat een applicatie is) de schermen beheert en deze dan ook moet

Listing 2.4: Programma voor bepalen van de schermresolutie onder MS Windows

```
1  #include <windows.h>
2  #include <stdio.h>
3
4  int main( void ) {
5
6    // x = width; y = height
7    int cx = GetSystemMetrics( SM_CXSCREEN );
8    int cy = GetSystemMetrics( SM_CYSCREEN );
9
10   if ( !( cx && cy ) ) {
11
12     printf( "Kon de huidige schermresolutie niet opvragen.\n" );
13     return 1;
14   }
15
16   printf( "De huidige schermresolutie is %d x %d.\n", cx, cy );
17   return 0;
18 }
```

Listing 2.5: Programma voor bepalen van de schermresolutie van X

```
1  #include <X11/Xlib.h>
2  #include <stdio.h>
3
4  int main( void ) {
5
6    /* Verbind met de X-server */
7    Display *dpy = XOpenDisplay( "" );
8    if ( dpy ) {
9
10     printf( "De huidige schermresolutie is %d x %d.\n", DisplayWidth(
           dpy, DefaultScreen( dpy ) ), DisplayHeight( dpy, DefaultScreen(
           dpy ) ) );
11
12     XCloseDisplay( dpy );
13   }
14   else {
15
16     printf( "Kon de huidige schermresolutie niet opvragen.\n" );
17     return 1;
18   }
19
20   return 0;
21 }
```

worden aangesproken om de resolutie te achterhalen. Dit betekent dat een methode voor het detecteren van software-eigenschappen per besturingssysteem kan verschillen. Wanneer er voor een besturingssysteem verschillende applicaties bestaan die het scherm kunnen aansturen (zoals de X-server), zou het kunnen voorkomen dat voor al deze applicaties een aparte routine opgesteld dient te worden om bijvoorbeeld de huidige schermresolutie te weten te komen.

Hoofdstuk 3

Representatie

Informatie, beschouwd in zijn meest abstracte vorm, heeft altijd een representatievorm; anders is ze verloren. Letters op een papier, bits in een geheugenchip en een bepaalde configuratie van neuronen in een brein, zijn allemaal manieren om informatie voor te stellen. Niet alle vormen van representatie zijn in alle situaties evenwaardig. Zo kan een mens slechts moeizaam en traag informatie in binaire vorm opnemen, en is de manier van opslaan van herinneringen in een menselijk brein voorlopig voor niemand helemaal duidelijk.

Een representatievorm kan beschouwd worden als een vorm van transport: er is altijd een bron en meestal ook een bedoelde ontvanger van de informatie. Zo is de bron van een binair bestand meestal een computer, en is het meestal ook de bedoeling dat dit binair bestand door een computer terug ingelezen wordt. Deze bron en bedoelde ontvanger, en de manier van transport zijn drie zeer belangrijke factoren in de keuze van een representatievorm.

Zoals bij alle vormen van communicatie, is het belangrijk dat er tussen de zender en ontvanger een consensus bestaat over het 'coderen' en 'decoderen' van informatie uit de gebruikte representatie.

3.1 Composite Capability/Preference Profiles

3.1.1 World Wide Web Consortium

Het World Wide Web Consortium (W3C [25]) is een internationale organisatie die zich bezig houdt met het ontwikkelen van standaarden en richtlijnen met betrekking tot het web. Tim Berners-Lee richtte in 1994 het W3C op met overheidssteun uit zowel Europa als de Verenigde Staten. Hij fungeert sindsdien als directeur ervan. Het consortium doet zijn werk door middel van organisaties en bedrijven die er lid van worden en samen aan standaarden werken en ideeën uitwisselen. Momenteel heeft het consortium meer dan 400 leden uit meer dan 40

landen [26, 27].

Als doel stelt het W3C om het web te leiden naar volledige ontplooiing door protocollen en richtlijnen te ontwikkelen die het web op lange termijn ten goede komen [28].

De activiteiten van het W3C worden gecategoriseerd in vier groepen [29]:

- Interactie (interaction);
- Technologie & samenleving (technology & society);
- Alomtegenwoordige web (ubiquitous web);
- Webtoegankelijkheid (web accessibility).

3.1.2 Achtergrond

Het W3C erkent het probleem dat gevormd wordt door de overvloed aan verschillende eigenschappen en mogelijkheden van eindgebruikerapparaten. Het observeert dat het web door een groeiend aantal verschillende soorten apparaten kan gebruikt worden. Het consortium heeft in dat licht de toegankelijkheid van het Web door iedereen, overal, altijd, op eender welke manier en dus apparaatonafhankelijk, tot één van haar hoofddoelen gesteld [30, 31].

Om dit te faciliteren werd onder andere in 2001 de Device Independence Working Group opgericht, welke in 2007 werd opgevolgd door de Ubiquitous Web Applications Working Group [32]. Deze eerste lanceerde in 2004 Composite Capability/Preference Profiles 1.0 (CC/PP [33]) als een W3C Recommendation.

Composite Capability/Preference Profiles is een raamwerk dat toelaat profielen op te stellen die de mogelijkheden en gebruikersvoorkeuren die samenhangen met het apparaat en de eindgebruiker beschrijven. Het doel van deze profielen is om een informatiebron en richtlijn te zijn bij het adapteren van media die naar het apparaat in kwestie wordt gestuurd, zodoende de media meer geschikt te maken voor dat specifieke apparaat en zijn gebruiker.

De CC/PP-standaard is gratis verkrijgbaar en makkelijk in te kijken via de W3C-website. Er zijn vele voorbeelden van vocabularia en profielen te vinden op het web.

Sommige van de vele andere standaarden die het W3C publiceert voorzien al in beperkte mate in het overdragen van apparaateigenschappen of andere meta-informatie [33]. Zo kunnen de headers in het HyperText Transfer Protocol (HTTP [34]) de ondersteunde karaktercoderingen (HTTP-header `accept-encoding`) van het gebruikersapparaat specificeren, en de gewenste talen van het opgevraagde document (HTTP-header `accept-language`). Er bestaan nog vele andere gelijkaardige voorbeelden verspreid over de W3C-standaarden. Deze oplossingen zijn echter meestal een beperkte nagedachte, ingekapseld in een protocol of standaard die een

volledig ander doel heeft. CC/PP onderscheidt zich hiervan als een veel meer geconcentreerde en gecentraliseerde algemene oplossing voor het overbrengen van de informatie in kwestie.

In augustus 2006 werd er begonnen aan CC/PP 2.0 [35]. Deze nieuwe versie werd voornamelijk geïntroduceerd om gebruik te kunnen maken van de mogelijkheden van een nieuwere versie van de RDF-standaard. De voornaamste verbetering is de mogelijkheid om beperkingen te kunnen toepassen op de waardes die een attribuut mag aannemen, zoals "enkel een integer kleiner dan 10". Wegens deze veranderingen zullen profielen opgesteld in overeenstemming met CC/PP 1.0, geen geldige CC/PP 2.0-profielen zijn [36].

In december 2006 kwam er een eerste *working draft*; in april 2007 werd er een 'last call' *working draft* vrijgegeven waar opmerkingen over konden worden ingezonden tot half juni 2007. Hierna is of was het de bedoeling om een kandidaat W3C *recommendation* op te stellen, maar op de website is het meest actuele nieuws hierover dat dit een lage prioriteit is, en voor februari 2008 verwacht werd.

3.1.3 Resource Description Framework

Om de structuur van de profielen voor te stellen, wordt er gebruik gemaakt van het Resource Description Framework (RDF [37, 38]). RDF is één van de standaardformaten voor de uitwisseling van data op het web. Het formaat is gebaseerd op de idee om Uniform Resource Identifiers (URI's) als identificatie van resources te gebruiken, en deze te beschrijven door er eigenschappen met waardes aan toe te kennen. Eenvoudig bekeken bestaat het dus uit zogenaamde 'triples' van data: een onderwerp, een eigenschap van dat onderwerp, en de waarde van die eigenschap van dat onderwerp. Deze eigenschappen kunnen zelf op hun beurt als onderwerp fungeren, opnieuw met een eigenschap en daar een waarde voor. Met RDF is het dus mogelijk om complexe, diep geneste structuren op te bouwen. In situaties waar dit niet gewenst is, en om het dupliceren van data tegen te gaan en de informatie makkelijker te kunnen beheren, zijn er mogelijkheden (die iets verder aan bod komen) om deze structuren op te splitsen.

RDF wordt doorgaans in XML-vorm opgeslagen en gecommuniceerd, maar dit is niet noodzakelijk. XML [39] is alomtegenwoordig op het hedendaagse Internet. Een voorbeeld dat zowel in XML-vorm (listing 3.1), als 'triples' (tabel 3.1), en als schematische voorstelling (figuur 3.1) wordt gegeven, demonstreert de simpliciteit van het gebruik van RDF.

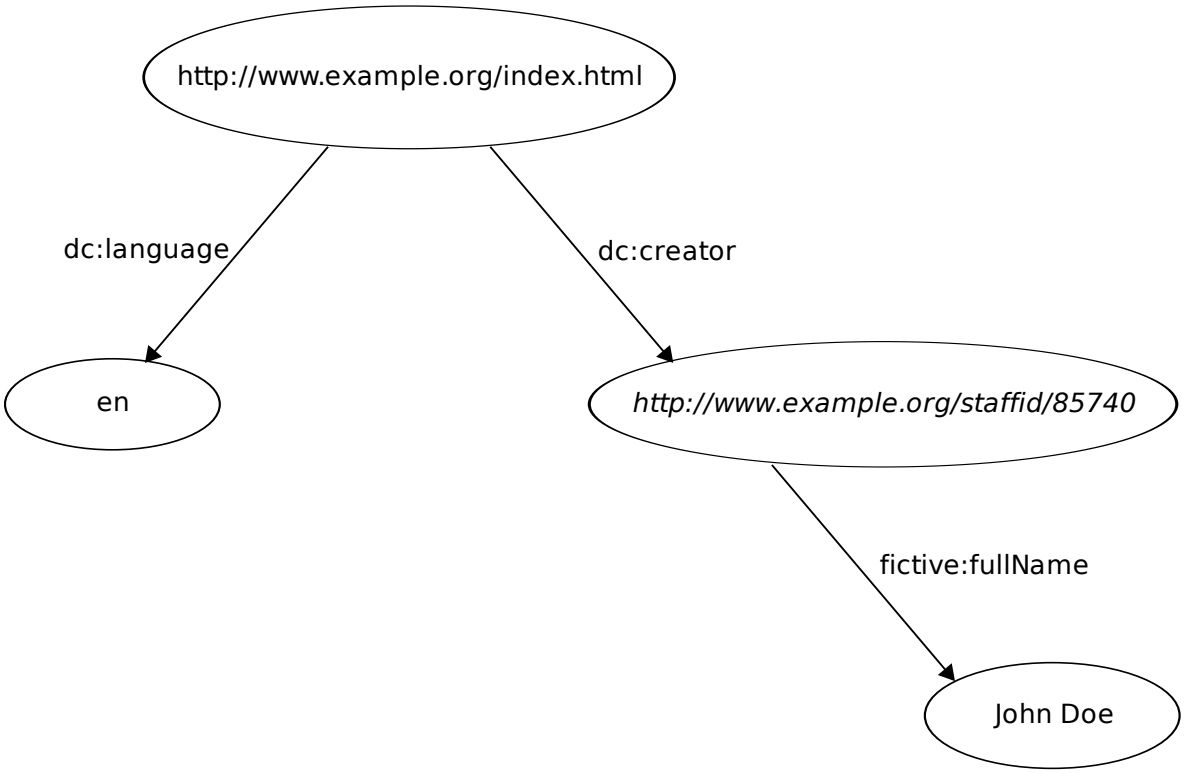
Tabel 3.1: Een eenvoudig voorbeeld van RDF, weergegeven als triples

Onderwerp	Eigenschap	Waarde
http://www.example.org/index.html	dc:language	en
http://www.example.org/index.html	dc:creator	*http://www.example.org/staffid/85740*
http://www.example.org/staffid/85740	fictive:fullName	John Doe

Listing 3.1: Een eenvoudig voorbeeld van RDF, weergegeven in XML

```
1  <?xml version="1.0"?>
2  <rdf:RDF rdf:xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:fictive="http://example.org/attributes/">
3
4    <rdf:Description rdf:about="http://www.example.org/index.html">
5      <dc:language>en</dc:language>
6      <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
7    </rdf:Description>
8
9    <rdf:Description rdf:about="http://www.example.org/staffid/85740">
10     <fictive:fullName>John Doe</fictive:fullName>
11   </rdf:Description>
12 </rdf:RDF>
```

Figuur 3.1: Een eenvoudig voorbeeld van RDF, weergegeven in diagramvorm



RDF zelf introduceert geen specifieke eigenschappen die ingevuld kunnen worden. Deze komen uit externe bronnen, zoals de relatief populaire Dublin Core-elementen [40]. Deze worden geïntroduceerd door middel van een XML namespace declaratie, zoals in listing 3.1 op regel 3. Om conflicten en verwarring bij deze manier van introduceren van eigenschap-elementen te voorkomen, worden de elementen van RDF zelf meestal ook geprefixd met hun namespace. In het voorbeeld zijn de RDF-elementen degene die beginnen met "rdf:".

Het `about`-attribuut van het eerste RDF-`Description`-element geeft aan dat het gaat om de resource die geïdentificeerd wordt met de URI "http://www.example.org/index.html". De elementen die hierbinnen liggen, geven informatie over deze resource. Zo wordt gesteld dat de `dc:language`-eigenschap van deze resource de waarde "en" heeft. De semantiek hiervan ligt niet besloten in het weergegeven voorbeeld, maar in dit geval kunnen we aannemen dat het betekent dat het onderwerp in de Engelse taal opgesteld is.

Bij de aanduiding van de `creator` op regel 8 van listing 3.1 zien we dat men niet noodzakelijk alle informatie in één document of `Description`-instantie moet houden, maar ook kan verwijzen naar andere RDF-data binnen of buiten het huidige document. Men geeft aan dat de `creator` beschreven wordt in de resource met URI "http://www.example.org/staffid/85740", die in dit geval vlak eronder beschreven staat. In tabel 3.1 en het diagram in figuur 3.1 worden verwijzingen in schuine tekst weergegeven, en eigenlijke waardes in normale tekst. Het is dit verwijzingsmechanisme dat een hiërarchie met slechts een paar niveau's mogelijk maakt binnen RDF, door veelvuldig met URI's te werken (die opnieuw op het hoogste niveau binnen de hiërarchie kunnen beginnen). Ook zorgt het mechanisme ervoor dat data die al eens beschreven werd, niet noodzakelijk gedupliceerd moet worden in andere RDF-documenten.

3.1.4 Opbouw van CC/PP

Een CC/PP-profiel is grofweg gestructureerd als een hiërarchie van twee niveaus [41]:

- Componenten: een categorie waarin meerdere attributen vallen;
- Attributen: ieder attribuut valt onder één van de componenten en stelt een eigenschap of voorkeur van het apparaat of de gebruiker voor.

Aangezien RDF is opgesteld als een zeer generieke methode van kennisuitwisseling, is het dus uitermate geschikt voor de representatie van deze CC/PP-componenten en -attributen.

Een veelvoorkomende subset van componenten, die ook in de standaard zelf wordt vermeld, is de volgende:

- Hardwareplatform: de hardware waarop de software momenteel draait;

- Softwareplatform: het geheel van software waarop de applicatie momenteel draait (besturingssysteem, ...);
- Applicatie: deze component bevat doorgaans attributen en voorkeuren van de applicatie in kwestie (bijv. een webbrowser);
- Voorkeuren: attributen die niet vastliggen in hardware, software of de applicatie, maar doorgaans door de eindgebruiker bepaald worden.

Net zoals bij RDF, bepaalt CC/PP op zich niet welke componenten en attributen in het profiel kunnen voorkomen. Dit wordt bepaald door het vocabulary. Deze vocabularies kunnen vrij opgesteld worden door derden en worden, ook net zoals bij RDF, geïntroduceerd door middel van een XML-namespace [42]. Een profiel kan ondersteuning bieden voor het coderen van eigenschappen die gedefinieerd zijn door verschillende vocabularies, door simpelweg de namespaces ervan te declareren. Deze vocabularies kunnen opgesteld zijn door verschillende partijen.

3.1.5 Voorbeeldvocabularium: UAProf

Een voorbeeld van het gebruik van CC/PP voor het representeren van apparaateigenschappen en gebruikersvoorkeuren is UAProf [43], dat gedefinieerd werd door het Wireless Application Protocol Forum [44]. Dit vocabularium spitst zich toe op attributen die betrekking hebben tot mobiele toegang tot het world wide web.

Het UAProf-vocabularium voor CC/PP bestaat uit zes CC/PP-componenten (elk optioneel), waarin in totaal 77 attributen ondergebracht zijn:

- HardwarePlatform Component (19 attributen)
- SoftwarePlatform Component (23 attributen)
- NetworkCharacteristics Component (4 attributen)
- BrowserUA Component (12 attributen)
- WapCharacteristics Component (12 attributen)
- PushCharacteristics Component (7 attributen)

Listing 3.2 toont een kort voorbeeld van een CC/PP-profiel dat gebruik maakt van het UAProf-vocabularium. Het is een excerpt uit het profiel [8] waar een Sony Ericsson S500i mobiele telefoon naar refereert iedere keer het toestel een webpagina opvraagt.

Listing 3.2: Een kort voorbeeld van een CC/PP-profiel

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:prf="http://www.openmobilealliance.org/tech/profiles/UAPROF/
       ccppschema-20021212#">
3
4  <rdf:Description rdf:ID="Profile">
5    <prf:component>
6      <rdf:Description rdf:ID="HardwarePlatform">
7        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/
             profiles/UAPROF/ccppschema-20021212#HardwarePlatform"/>
8        <prf:Vendor>Sony Ericsson Mobile Communications</prf:Vendor>
9        <prf:Model>S500i</prf:Model>
10       <prf:ScreenSize>240x320</prf:ScreenSize>
11     </rdf:Description>
12   </prf:component>
13 </rdf:Description>
```

Het toont hoe het UAProf-vocabularium wordt geïntroduceerd door middel van de prf-namespace. Ook ziet men dat het profiel wordt opgedeeld in componenten die op hun beurt de attributen bevatten. In dit voorbeeld wordt er slechts één component, "HardwarePlatform", getoond.

Hoewel UAProf nu beschikbaar is als een vocabulary dat gebruik maakt van het CC/PP-raamwerk, is CC/PP eerst ontworpen als een soort abstractie van de reeds bestaande UAProf-standaard, waarbij er zoveel mogelijk geprobeerd werd bestaande UAProf-profielen compatibel te houden met CC/PP 1.0 [33].

3.1.6 CC/PP-uitwisseling via HTTP-headers

De CC/PP-standaard houdt zich afzijdig in verband met het transport van de profielen en specificeert enkel de RDF-gebaseerde representatie van de componenten en attributen. Wel wordt er gesuggereerd dat in bepaalde gevallen CC/PP-profielen over het HTTP-protocol getransporteerd kunnen worden [45].

Het "CC/PP exchange protocol based on HTTP Extension Framework" [46], of kortweg CCPPex, maakt gebruik van het HTTP Extension Framework [47]. Het Extension Framework is een toevoeging aan HTTP 1.1 [34], die als doel heeft uitbreidingen aan de HTTP-protocol-headers mogelijk te maken. Dit wordt gedaan op een manier die verzekert dat HTTP-applicaties (zowel clients als servers) die geen kennis hebben van het HTTP Extension Framework, correct blijven functioneren.

Een nadeel van CC/PP waar CCPPex mee te maken krijgt, is dat CC/PP profielen niet in een compact formaat staan. Het CCPPex-uitwisselingsprotocol profiteert van het feit dat CC/PP op RDF is gebaseerd. Door de in RDF ingebouwde mogelijkheid van verwijzingen (gedemonstreerd in listing 3.1) te benutten, moeten profielen niet noodzakelijk altijd volledig doorgestuurd worden. Men kan met verwijzingen en toevoegingen werken.

Het versturen van profielinformatie in HTTP-headers of het opvragen van een *remote profile* over een onbeveiligde verbinding, vormt een bijkomend nadeel. Het betekent dat er heel wat informatie over de hardware en software van de gebruiker en over de gebruiker zelf in een door derden leesbaar formaat wordt verstuurd. Ook kunnen tussenliggende routers of proxyservers de profielen verwijderen of aanpassen. Dit brengt heel wat privacy- en beveiligingszorgen met zich mee. Zowel de CC/PP-standaard als de CCPPex-standaard besteden hier weinig of geen aandacht aan, dus alle implementaties die aan deze twee standaarden voldoen, hebben niet noodzakelijk de faciliteiten aan boord om deze zorgen te adresseren. Een mogelijke oplossing is om HTTPS (Secure HTTP, [48]) in te zetten om de gehele HTTP-verbinding te beveiligen, maar dit is niet altijd een haalbare optie. Het wordt bijvoorbeeld door heel wat populaire publieke websites niet ondersteund.

HTTP-extensie declareren

De eerste stap bestaat eruit aan elk HTTP-request een `Opt`-header toe te voegen waarmee kenbaar gemaakt wordt welke HTTP-extensies er aanwezig zijn. HTTP-applicaties die CCPPex of HTTPex niet ondersteunen, zullen deze bijkomende HTTP-headers simpelweg negeren, aangezien dit zo verplicht wordt in de HTTP-standaard. Deze applicaties zullen het HTTP-verzoek zonder extensie proberen te vervullen.

De inhoud van de toegevoegde `Opt`-header bevat een lijst van HTTP-extensies, gescheiden door kommas. Ieder item in de lijst bestaat uit een *extension identifier* en een *namespace header prefix*. De syntax wordt weergegeven in listing 3.3.

Listing 3.3: Syntax van de `Opt`-header

```
1 Opt: "[extensionidentifier]" ; ns=[namespace header prefix] , ...
```

De *extension identifier* is normaalgezien een absolute URI. Deze dient voor iedere extensie uniek te zijn. Voor de CCPPex HTTP-extensie is deze URI "http://www.w3.org/1999/06/24-CCPPexchange". Deze URI wordt enkel gebruikt om de extensie uniek te identificeren; het is niet verplicht om deze URI naar een geldige resource te doen verwijzen. De *namespace header prefix* is een getal waarmee alle volgende HTTP headers afkomsting van de bijhorende HTTP-extensie beginnen. De *extension identifier* en *namespace header prefix* zijn dus sterk

vergelijkbaar met respectievelijk de namespace URI's en namespace prefixes in XML.

Door de *namespace header prefix* wordt er een soort van namespace binnen de HTTP-headers gereserveerd per HTTP-extensie. In de HTTPex-standaard wordt aangeraden voor dezelfde extensie telkens hetzelfde *namespace header prefix* te gebruiken. Voor CCPPex is dit traditioneel 19.

Profiel(en) aanduiden

De profielen die de applicatie wil aangeven worden in CCPPex in de `Profile`-header aangegeven. De header dient geprefixed te worden door de *namespace header prefix* uit de `Opt`-header.

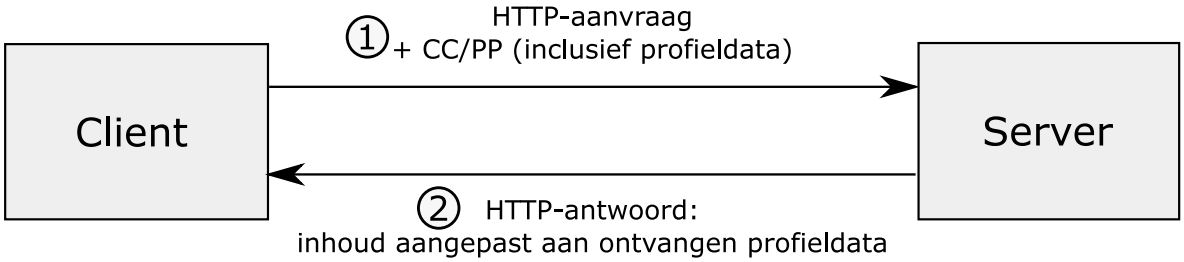
Het uitwisselingsprotocol voorziet twee manieren om CC/PP-profielen kenbaar te maken aan de serverapplicatie. De eerste mogelijkheid is om profieldata binnen de HTTP-headers zelf te plaatsen, in een zogenaamde *profile diff*. Dit kan een volledig profiel zijn, of een toevoeging of update ten opzichte van een profiel dat als *remote profile* wordt verzonden.

Wanneer de gebruiker bijvoorbeeld het geluid uitschakelt, hoeft men niet noodzakelijk het hele profiel (aangepast om te reflecteren dat het geluid uit staat) te sturen. Een *remote profile*, gecombineerd met een incrementele update (als *profile diff*) die enkel de status van het geluid specificeert, kan volstaan.

Een *profile diff* wordt in de `Profile`-header aangeduid met een volgnummer (*diff number*) en de MD5-hash van de eigenlijke profieldata (in base64-codering), gescheiden door een streep. De eigenlijke profieldata volgt in een aparte header.

Figuur 3.2 geeft een schematische weergave van deze werkwijze.

Figuur 3.2: Schematische weergave van een *profile diff*



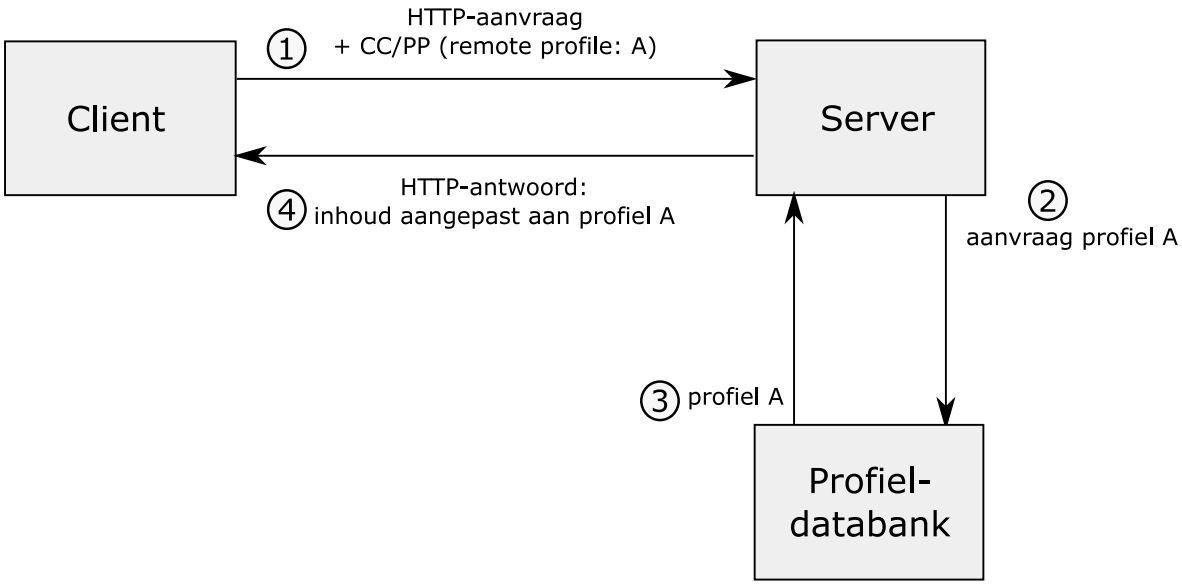
De tweede manier is een "remote profile", waarbij het profiel of de profielen in de `Profile`-header aangeduid wordt door middel van Universal Resource Identifiers (URI's). Het is dan aan de ontvangende applicatie om het eigenlijke profiel zelf te bemachtigen.

Een groot voordeel van *remote profiles* is dat ze door de ontvangende applicatie gecached kunnen worden. Een veelvoorkomend profiel (bijvoorbeeld het hardware-profiel van een veel-

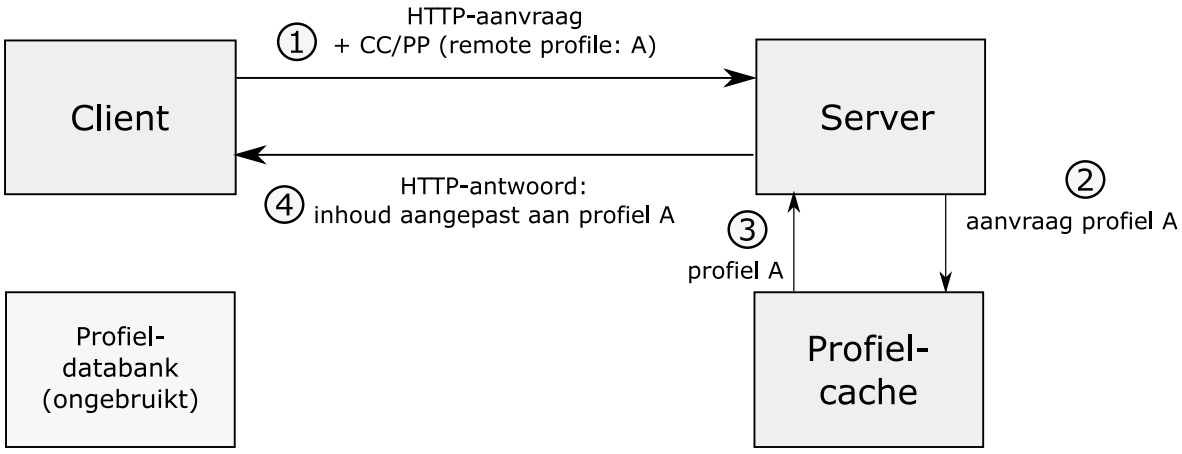
gebruikt apparaat) kan onthouden worden en hoeft dus niet iedere keer van een externe locatie verkregen te worden. Wanneer een ontvanger een verwijzing ontvangt die niet in zijn cache is terug te vinden, stuurt hij simpelweg een verzoek om het profiel naar de locatie van de verwijzing.

Een mogelijk scenario tussen een HTTP-server en HTTP-client dat enkel van deze manier gebruik maakt, wordt weergegeven in figuur 3.3. De situatie die zich voordoet wanneer het profiel al gecached is, staat in schemavorm in figuur 3.4.

Figuur 3.3: Schematische weergave van een *remote profile*



Figuur 3.4: Schematische weergave van een gecached *remote profile*



In listing 3.4 wordt de syntax van de *Profile*-header aangegeven. Het is een opsomming

van profielverwijzingen, gescheiden door komma's. Wanneer in de verschillende verwijzingen een attribuut meerdere malen voorkomt, is het het laatste profiel in de opsomming dat de uiteindelijke waarde van het attribuut bepaalt.

Listing 3.4: Syntax van de `Profile`-header

```
1 Profile: "[remote-profile-URL]", "[diff number]-[diff hash]", ...
```

Profieldata invoegen

Als er in de `Profile`-header aangegeven werd dat er *profile diff*s volgen, dienen deze in een `Profile-Diff`-header te zitten:

Listing 3.5: Syntax van de `Profile-Diff`-header

```
1 Profile-Diff-[diff number]: [profile-data]
```

Voorbeeld

In de hoop het bovenstaande concreter en duidelijker te maken volgt in listing 3.6 een voorbeeld van een HTTP-verzoek dat van deze methode gebruik maakt.

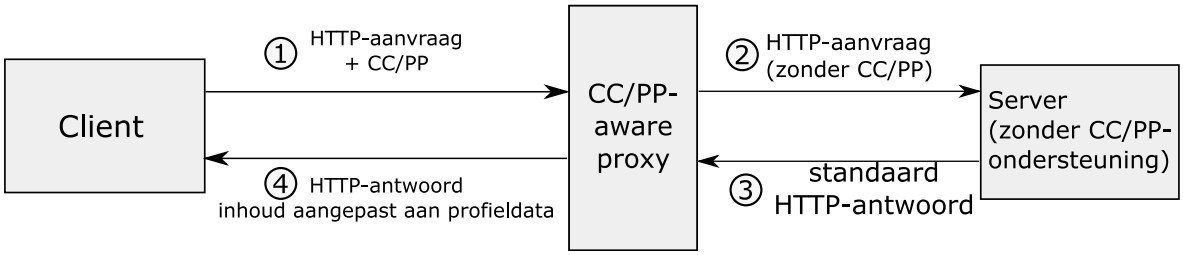
Listing 3.6: Een HTTP-verzoek dat een profielverwijzing en een profielstuk bevat

```
1 GET /webpage.html HTTP/1.1
2 Host: www.w3.org
3 Opt: "http://www.w3.org/1999/06/24-CCPPexchange" ; ns=19
4 19-Profile: "http://wap.sonyericsson.com/UAprof/S500iR201.xml",
      "1-uKhJE/AEeeMzFSejsYshHg=="
5 19-Profile-Diff-1: <?xml version="1.0"?>
6   <rdf:RDF rdf:xmlns="http://www.w3.org/TR/1999/PR-rdf-syntax-19990105#"
        xmlns:PRF="http://www.w3.org/TR/WD-profile-vocabulary#">
7   <rdf:Description rdf:ID="SoftwarePlatform" PRF:Sound="Off" />
8   </rdf:RDF>
```

De realiteit is dat weinig webapplicaties zich bewust zijn van mogelijke CC/PP-profielen in de headers van een HTTP-verzoek. Om toch geadapteerde content te kunnen aanbieden aan apparaten waar het nodig is, kan men een proxyserver of gateway tussen het apparaat en de

content server plaatsen. Als deze wel ondersteuning biedt voor CCPPex en op basis van de CC/PP profielen de inhoud die via de proxy naar de client reist, adapteert, kan de client toch beschikken over geadapteerde inhoud. Figuur 3.5 geeft deze situatie weer.

Figuur 3.5: Schematische weergave van een profiel-bewuste proxy



3.2 MPEG-21

3.2.1 MPEG

De Moving Picture Experts Group (MPEG [49]) is een werkgroep van de Internationale Organisatie voor Standaarden (ISO [50]). Ze werd opgericht in 1988 voor de codering van audio en video. Of, specifieker, voor de ontwikkeling van internationale standaarden voor compressie, decompressie, verwerking en gecodeerde representatie van bewegende beelden, geluid, en hun combinatie, voor een groot bereik van applicaties [51]. Sindsdien heeft de groep die missie uitgebreid om "ondersteunende technologieën" te incorporeren. MPEG probeert standaarden te ontwikkelen die een wijd gebruik kennen.

De motivatie voor de oprichting was de nood aan standaardisatie op de voorgenoemde vlakken vanuit de relevante private sectoren. Ze beseften dat een gemeenschappelijk formaat voor de data op verschillende merken van gegevensdragers (CD's, harde schijven, ...) dat door verschillende merken van mediaspelers (computers, DVD-spelers, "video-CD"-spelers, ...) kon afgespeeld worden, zou betekenen dat er een veel grotere, veel minder versplinterde markt gecreëerd zou worden.

Door enkel het absoluut noodzakelijke voor compatibiliteit en efficiëntie vast te leggen, probeert MPEG ervoor te zorgen dat de standaarden die het ontwikkelt langer mee kunnen gaan. Ook ontstaat er door de keuzes die opengelaten worden, concurrentie tussen verschillende implementaties van eenzelfde standaard. Zo kan het zijn dat één MPEG-2-encoder bijzonder geschikt is voor tekenfilm-achtige video, terwijl een andere gespecialiseerd is in snelle actiebeelden. Een bijkomend voordeel is dat de standaard na het vastleggen ervan nog steeds kan winnen aan efficiëntie, door steeds slimmere implementaties.

Sinds de oprichting ervan heeft de MPEG vele standaarden uitgewerkt en gepubliceerd die in wijd gebruik zijn. Tussen de oprichting van de Moving Picture Experts Group en het publiceren van MPEG-21 (waarover verder veel meer) liggen MPEG-1, MPEG-2, MPEG-4 en MPEG-7.

MPEG-1

MPEG-1 [52] werd ontwikkeld in de periode 1988–1991 en was bedoeld voor codering van audio en video. Hoewel zowel het audio- als het videogedeelte van MPEG-1 ondertussen is overgedaan in meer geavanceerde standaarden met hetzelfde doel (MPEG-2 en MPEG-4), blijft het een veelgebruikte standaard. Alle MPEG-standaarden zijn opgesplitst in delen ("parts"). Het waarschijnlijk bekendste deel van MPEG-1 is MP3 (kort voor "MPEG-1, part 3 (audio), layer III") dat op zichzelf al een revolutie ontketende op vlak van muziekdistributie.

MPEG-2

MPEG-1 was een groot succes, maar de coderingsmethodes die erin gebruikt werden, konden nog uitgebreider en meer efficiënt gemaakt worden. MPEG-2 [53] kan — kort door de bocht — gezien worden als een verbetering en uitbreiding van MPEG-1, met grotendeels dezelfde doelstellingen. MPEG-2 bood meer mogelijkheden (meer bitrates, meer resoluties, ...) voor dezelfde doelen. Het audio-gedeelte is achterwaarts compatibel met dat van MPEG-1. Na initiële publicatie werd er een tweede, meer geavanceerd, audio-gedeelte aan toegevoegd dat ondertussen bekend staat als Advanced Audio Coding (AAC). MPEG-2 is de standaardcodering voor video-DVD's, wat een gebruik door miljoenen systemen impliceert.

MPEG-4

De volgende standaard, MPEG-4 [54] (MPEG-3 werd geannuleerd), is opnieuw een veel geavanceerdere, en de meest recente, coderingsstandaard van de MPEG. Het werd echter niet als *vervanging* van MPEG-2 gezien, maar eerder een aanvulling voor applicaties die een hogere kwaliteit aankunnen of vereisen.

Populaire implementaties van de audio- en videogedeeltes van MPEG-4 zijn Divx [55], Xvid [56] en x264 [57]. Ook is het één van de ondersteunde coderingen voor Blu-ray schijven, volgens velen de opvolger van video-DVD's.

MPEG-7

De focus van MPEG-7 [58], waaraan het werk begon in 1996, lag niet op de codering van media zelf, maar op data en beschrijvingen over deze media, oftewel metadata, te coderen.

Toen er aan deze standaard begonnen werd, was er ondertussen al een gigantische hoeveelheid digitale media beschikbaar. De MPEG zag de nood om het *beheer* van deze grote hoeveelheid te vergemakkelijken en te standaardizeren. Het doel van deze standaard is dan ook om alle soorten van beheer van digitale media, al dan niet gecodeerd met één van de voorgaande MPEG-standaarden, te vergemakkelijken.

3.2.2 MPEG-21

Met de voorgaande standaarden van de MPEG had de organisatie veel noden in de context van digitale media gevuld. MPEG-1, -2 en -4 voldoen aan een ruime vraag voor coderingsmethodes van audio en video. MPEG-7 helpt bij het coderen van de beschrijvingen of metadata van deze media om deze zo beter te kunnen beheren. MPEG had dus standaarden gepubliceerd voor onder andere de codering van audio en video, de opslag daarvan en het voorstellen van informatie erover.

Het ambitieuze doel van MPEG-21 [59] (zo genoemd omdat het de eerste MPEG-standaard in de 21ste eeuw diende te worden) was om alles te zijn wat er, naast de bestaande MPEG-standaarden, nog nodig was om een volledig geïntegreerde infrastructuur voor mediadistributie en -consumptie te bouwen. Met andere woorden, MPEG-21 moest de gaten tussen de voorgaande standaarden vullen om volledig MPEG-gestandaardiseerde mediadistributie mogelijk te maken.

Toen het begin van MPEG-21 werd gemaakt in 1999 was het — veel meer dan bij de oprichting van de MPEG in 1988 — duidelijk dat er niet alleen nood was aan standaardisatie van mediacodering. Door de opkomst van digitale netwerken, waaronder zeker het Internet, kwam er een nieuw kanaal voor digitale media te verhandelen. Een video kopen staat niet langer synoniem voor een fysieke gegevensdrager kopen. Een recent voorbeeld hiervan zijn de vele Video-on-Demand (VoD) systemen die in België worden aangeboden aan de digitale consument door onder andere Belgacom (Belgacom TV [60]) en Telenet (Telenet Digital TV [61]).

De MPEG-21-standaard wordt dus samengesteld uit onder andere bijdrages van verschillende bedrijven. De MPEG heeft regelgeving die stelt dat deze bedrijven moeten verklaren dat eventuele patenten die op hun bijdrages rusten, op “Reasonable and non-discriminatory” termen (RAND-termen) gelicencieerd dienen te worden. Helaas worden noch “Reasonable”, noch de RAND-termen concreet ingevuld. Enkele delen van de standaard, zoals de XML-

schema's en sommige referentie-implementaties zijn vrij verkrijgbaar, maar alle andere delen van de standaard niet.

Het element dat in MPEG-21 als een soort basiseenheid wordt beschouwd, is het Digital Item (DI). De concrete letterlijke definitie ervan is als volgt:

> A Digital Item is a structured digital object with a standard representation, identification and metadata within the MPEG-21 framework. This entity is also the fundamental unit of distribution and transaction within this framework.

Het is eender welk digitaal 'object' samen met een eventuele identificatie en bijhorende metadata (MPEG-7). Concreet betekent het dat alle media die door het MPEG-21-systeem verwerkt wordt, in een Digital Item-container geplaatst wordt. Voorbeelden hiervan zijn een film, een muziekverzameling, of een audiosample.

De focus van MPEG-21 kan dan beschouwd worden als het faciliteren van de 'handel', zoals ruil of verkoop, van Digital Items.

De focus ligt op een heel aantal doelen [62]:

- Digital Item Declaration (DID): Digital Items zijn in MPEG-21 de basisblokken van digitale media. DID probeert een abstracte manier te zijn om DI's te declareren;
- Digital Item Identification: een manier om DI's te identificeren;
- Content Handling and Usage: Interfaces en protocollen voor het aanmaken, manipuleren, zoeken, openen, opslaan, toeleveren en (her)gebruiken van Digital Items ('content');
- Intellectual Property Management and Protection: de middelen om de rechten van Digital Items continu en betrouwbaar te beschermen en beheren;
- Terminals and Networks: de mogelijkheid om transparante toegang tot multimedia te verzekeren voor verschillende soorten netwerken en terminals;
- Content Representation: gaat over de representatie van multimedia ('content');
- Event Reporting: gaat over de interfaces en gegevens die toelaten om gebeurtenissen binnen het systeem te beheren.

3.2.3 MPEG-21 Part 7 ("Digital Item Adaptation")

Het zevende deel [63] van de MPEG-21-standaard probeert het doel van "samenwerkende, transparante toegang tot geavanceerde multimedia te bereiken en gebruikers (Users) af te schermen van het installeren, beheren, en implementeren van het netwerk en terminal" door

middel van Digital Item Adaptation (DIA). Dit doel komt sterk overeen met Universal Multimedia Access (UMA; dit concept wordt uitgebreider uitgelegd in sectie 4.1.1). Deze Digital Item Adaptation kan gebruikt worden om Digital Items te laten voldoen aan voorkeuren van de gebruiker, apparaateigenschappen, netwerkeigenschappen, omgevingseigenschappen, opslagvereisten, et cetera. Het is dus dit deel van MPEG-21 dat verdere aandacht verdient in dit werk.

De zogeheten Digital Item Adaptation tools (DIA tools) bevatten drie verschillende klasses:

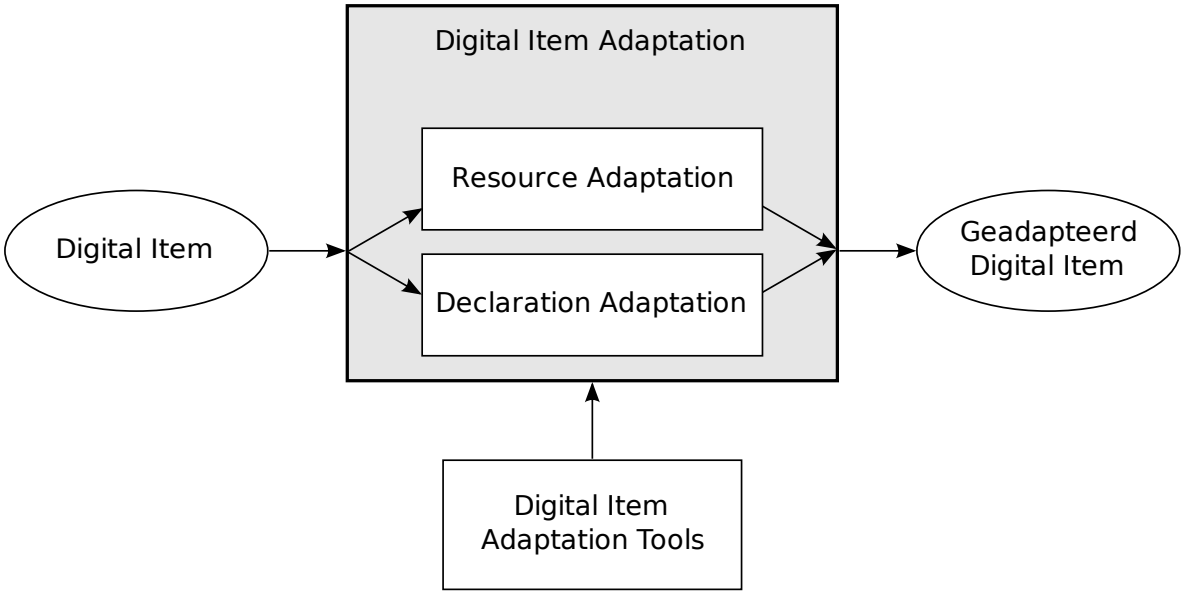
- Usage Environment Description Tools:
  - Gebruikerseigenschappen;
  - Terminaleigenschappen;
  - Netwerkeigenschappen;
  - Natuurlijke-omgeving-eigenschappen.
- Digital Item Resource Adaptation Tools:
  - Bitstream Syntax Description (BSD);
  - Terminal- en netwerk-QoS;
  - Bitstream Syntax Description Link (BSD Link);
  - Metadata-aanpasbaarheid.
- Digital Item Declaration Adaptation Tools:
  - Sessiemobiliteit;
  - DIA-configuratie.

De eerste klasse, de Usage Environment Description (UED) tools, dient om informatie die origineert van de gebruiker (of het gebruikersapparaat) voor te stellen. De tweede klasse, de Digital Item Resource Adaptation tools, dient om de binaire resources binnen een Digital Item aan te passen. Zo kan de Bitstream Syntax Description (BSD) gebruikt worden om een binaire resource op een formaat-onafhankelijke manier aan te passen. De Digital Item Declaration Adaptation Tools vormen de derde en laatste klasse, welke de gehele Digital Item Declaration aanpassen. Zo kan de huidige gehele configuratie van het DIA-systeem overgedragen worden tussen client-apparaten. Een voorbeeld van deze situatie is wanneer een Digital Item eerst op PDA wordt bekeken, en daarna overgeschakeld wordt op een televisie.

Deze DIA tools kunnen zelf ook een Digital Item zijn, of een onderdeel ervan.

Een schematische weergave van het systeem [63] wordt gegeven in figuur 3.6. De Digital Item Adaptation wordt uitgevoerd op een invoer-Digital Item. De operatie of operaties die

Figuur 3.6: De schematische structuur van een Digital Item Adaptation Engine



uitgevoerd worden, worden gespecificeerd door de DIA tools. Deze kunnen apart (zoals in het diagram) aangeleverd worden, of deel zijn van het Digital Item waar de operatie(s) op worden uitgevoerd. De operaties zijn niet beperkt tot het aanpassen van enkel 'media' in het Digital Item; ook beschrijvingen of de eventuele DIA-elementen die erin zitten, kunnen aangepast worden. Dit betekent dat wanneer je een Digital Item een tweede keer door een DIA-engine stuurt, het resultaat niet noodzakelijk hetzelfde is.

3.2.4 MPEG-21: Part 7: Usage Environment Description Tools

De eigenlijke beschrijving van de gebruiksomgeving wordt onderverdeeld in vier categorieën. Hier worden ze overlopen en de eigenschappen die eronder vallen besproken. De categorieën zijn gebruikerskarakteristieken ("User characteristics"), apparaatmogelijkheden ("Terminal capabilities"), netwerkkarakteristieken ("Network characteristics") en omgevingskarakteristieken ("Natural environment characteristics").

3.2.5 Gebruikerskarakteristieken

Dit onderdeel gaat over de eigenschappen en attributen die informatie uitdrukken over de voorkeuren van de gebruiker, of over de gebruiker zelf.

User Info

Deze eigenschap gaat over gegevens over de eigenlijke gebruiker, zoals een naam of emailadres. De concrete syntax en semantiek wordt overgenomen uit MPEG-7.

Content Preferences

Deze eigenschap gaat over de voorkeuren van de gebruiker voor bepaalde types multimedia. De concrete invulling wordt overgelaten aan MPEG-7. Er zijn twee verschillende manieren om de voorkeur over te brengen. Er kunnen in een `UserPreferences`-element concrete voorkeuren opgegeven worden. Voorbeelden hiervan zijn tijdstip, auteur, genre ("Sports", "Movies", ...), taal, formaat, ... Zo kan men bijvoorbeeld aangeven dat men items wil zien uit het genre sport, die als tijdstip de datum van de vorige dag hebben.

De andere mogelijkheid is om in een `UsageHistory`-element een geschiedenis van al 'geconsumeerde' Digital Items op te geven. De achterliggende gedachte is dat uit deze geschiedenis de voorkeur van de gebruiker afgeleid kan worden.

Onder het `UserPreferences`-element valt ook het `ParentalGuidance`-element, waarin het `ParentalRating`-element de voorkeurs-'rating' bevat van de media. Een voorbeeld van een mogelijke rating is de "R"-rating ("restricted") die uitgedeeld wordt door de Motion Picture Association of America (MPAA [64]).

Audio Presentation Preferences

De presentatievoorkeuren van de gebruiker die met audioweergave te maken hebben, worden beschreven met behulp van een `Audio`-element. Veelgebruikte attributen voor dit element zijn `SamplingFrequency` voor het aantal samples per tijdseenheid (in Hz) en `NumOfChannels` voor het aantal kanalen dat de voorkeur heeft.

Display Presentation Preferences

Het `Display`-element bestaat uit twee soorten subelementen, zijnde `ColorPreferenceType` en `StereoscopicVisionPreference`. Het eerste type gaat over de voorkeuren van de gebruiker over de kleurweergave. Hieronder vallen de kleurtemperatuur (`ColorTemperaturePreference`), de helderheid (`BrightnessPreference`), de saturatie (`SaturationPreference`) en het contrast (`ContrastPreference`). Het tweede type relateert, weinig verrassend, de gebruikersvoorkeuren omtrent de instellingen voor stereoscopie (eender welke techniek die de illusie van diepte in tweedimensionale creëert) in de elementen `From2DTo3DStereoscopic` en `From3DStereoscopicTo2D`.

Graphics Presentation Preferences

Onder dit deel vallen enkele elementen die met een relatieve waarde (tussen 0 en 1) aangeven in hoeverre de geometrie (`GeometryEmphasis`), textuur (`TextureEmphasis`) en animatie (`AnimationEmphasis`) gedegradeerd mogen worden bij compressie of hercodering van graphics. Een waarde van 1 betekent dat de gebruiker de voorkeur geeft aan het behouden van de geometrie-, textuur- of animatiedata. Hoe de adaptatie-engine de waarde dient te interpreteren, wordt niet gespecificeerd.

Modality Conversion Preferences

Het veranderen van de modaliteit van een bron wil zeggen dat het type aangepast wordt. Van een opgenomen spraakbestand kan een transcript gegenereerd worden; dit is dan een textuele modaliteit ervan. Het kan nuttig zijn wanneer een gebruikersapparaat een bepaalde modaliteit niet ondersteunt, zoals wanneer het geen manier heeft om audio weer te geven.

Aangeven van deze gebruikersvoorkeur kan op twee manieren gebeuren. Een eerste manier kan voor alle items van een bepaald type een geprefereerde andere modaliteit specificeren (`GeneralResourceConversions`). De tweede manier kan voor individuele items (die geïdentificeerd worden met een URI) de conversie opgeven die de voorkeur van de gebruiker verdient (`SpecificResourceConversions`).

Het opgeven van de modaliteiten waarvan en waarnaar geconverteerd dient te worden, gebeurt aan de hand van MPEG-7-namen (`Video`, `Text`, ...).

De conversievoorkeur op zich (zij het van een type of van een individueel item) wordt opgegeven in twee waardes. De eerste waarde is een rangorde van modaliteiten (het `order`-attribuut van het `Conversion`-element). De tweede waarde is een gewogen voorkeur per opgegeven modaliteit in die rangorde (het `weight`-attribuut van het `Conversion`-element). Men kan bijvoorbeeld de `order`-attributen gebruiken om aan te geven dat als video geconverteerd dient te worden naar een andere modaliteit, men liefst heeft dat dat audio is, en op de tweede plaats pas een transcript. Door de `Audio`-modaliteit een `weight`-waarde van 1 mee te geven en een transcript een waarde van 0.1, kan men duidelijk maken dat een conversie naar `Text` als een veel groter verlies beschouwd wordt als een conversie naar `Audio`. De adaptatie-engine kan dan bijvoorbeeld bij gebrek aan bandbreedte eerst de video omzetten naar een stream met enkel geluid. Wanneer dit niet voldoende is, wordt gekeken naar de volgende modaliteit. In dit geval is dat `Text`, maar aangezien het geassocieerd `weight`-attribuut hiervan relatief klein is, kan de adaptatie-engine bijvoorbeeld eerst proberen de bitrate van het geluid te verlagen zodat een conversie naar tekst onnodig is. Omdat de MPEG zo weinig mogelijk vastlegt in de standaard, is het specifieke gedrag van de adaptatie-engine in dit geval afhankelijk van de implementatie.

Presentation Priority Preference

Met het `PresentationPriority`-element kan het belang van de kwaliteit van multimedia aangegeven worden. Net zoals bij de Modality Conversion Preferences wordt dit opgegeven aan de hand van een relatief gewicht, zowel voor alle items (met het `GeneralResourcePriorities`-element), als voor een specifiek item (`SpecificResourcePriorities`).

Wanneer dit voor een specifiek item wordt gedaan, is een URI naar dat item vereist en wordt aan het gehele item (`Object`) een gewicht (het attribuut `priorityLevel` van het `Object`-element) toegekend.

Bij de algemene manier worden de items onderverdeeld op basis van modaliteit en genre (`ModalityPriorities` en `GenrePriorities`, respectievelijk). Men kan dan relatieve gewichten toekennen aan modaliteiten, genres, of een combinatie van die twee. Een voorbeeld hiervan kan zijn dat wanneer een video verkleind moet worden (in termen van bits per seconde), de gebruiker opgeeft dat de modaliteit `Audio` een grotere kwaliteitsvoorkeur heeft dan de modaliteit `Video`. Het gevolg hiervan zou zijn dat het geluid met minder kwaliteitsverlies gecomprimeerd zal worden dan wanneer de gebruiker deze voorkeur niet aangegeven had. Bij het waarschijnlijke geval dat de gecombineerde video stream een constante bitrate moet hebben, heeft dit als logisch gevolg dat de video met meer informatieverlies geadapteerd zal worden. Een ander scenario is dat de gebruiker aangeeft meer belang te hechten aan de kwaliteit van (delen van) items die van het genre `Sports` zijn. Beide voorgaande voorbeelden kunnen tegelijk voorkomen. Dan wordt de grootste prioriteit verleend aan audio streams over sport.

Focus Of Attention

De "Focus Of Attention", of betekenisvoller de "region of interest" (ROI), van een gebruiker is datgene binnen een multimediabron waarvoor hij meer interesse of aandacht heeft ten opzichte van de rest van die multimediabron. De regio in kwestie kan worden bepaald door onder andere een aanwijsapparaat of een *eye tracker*, of op voorhand gedefinieerd worden (bijvoorbeeld simpelweg het midden van het scherm).

Deze ROI (in het gelijknamige element) kan worden aangeduid door een ruimtelijk masker (een rechthoek gedefinieerd door vier coördinaten). Dit wordt via een MPEG-7 `Box`-element gedefinieerd.

Om een verplaatsende regio aan te geven kan men in het `ROI`-element een `updateInterval`-attribuut plaatsen, dat aangeeft om de hoeveel seconden de gedefinieerde regio opnieuw verstuurd zal worden. Dit impliceert wel dat er minstens iedere zoveel seconden een update van het UED-tool verstuurd dient te worden.

Een alternatief voor de ROI, dat het voordeel heeft dat er geen periodieke update voor moet worden verstuurd, is de "Focus of Attention" definiëren in termen van een semantisch scene-object (`SceneObjectFocusOfAttention`). Dit dient te worden gecodeerd in een MPEG-7 Media Locator.

In het geval dat het om een tekst-item gaat, hebben bovenstaande ruimtelijke bepalingen weinig nut. Voor deze items is er een `TextFocusOfAttention`-element voorzien, waarmee de gebruiker door middel van `Keyword`-elementen en een `Font`-element kan aangeven welke sleutelwoorden belangrijk zijn, en naar welk lettertype de voorkeur uitgaat. Hoe de adaptatie-engine deze informatie gebruikt om tekst-bronnen aan te passen, wordt in de UED-specificatie opengelaten.

Auditory Impairment

Het Auditory Impairment-gedeelte kan de eventuele gehoorproblemen van een gebruiker overbrengen. De adaptatie-engine kan hiervoor (mits de problemen klein zijn) compenseren. Bijkomend kan informatie over het gehoor gebruikt worden om audio efficiënter te coderen. Er zouden bijvoorbeeld heel hoge tonen weggelaten kunnen worden, als het oor van de gebruiker deze niet kan opvangen.

Het verlies van gevoeligheid wordt per oor apart opgegeven (`RightEar` en `LeftEar`). Voor verschillende frequenties (125, 250, 500, 1000, 1500, 2000, 3000, 4000, 6000 en 8000 Hertz) wordt de gehoorgrens in decibel opgegeven.

Visual Impairment

Net zoals gehoorproblemen kunnen ook eventuele zichtproblemen kenbaar gemaakt worden, binnen het `Visual`-element. Omdat slechtziendheid veel oorzaken kan hebben, wordt de specifieke aandoening niet gecodeerd, maar wel de symptomen ervan.

Men kan volledige blindheid aangeven met het `Blindness`-element, waarbij het `eyeSide`-attribuut aangeeft om welk oog het gaat (`left`, `right` of `both`).

Slechtziendheid wordt aangegeven met symptomen die als waarde de ernstigheid van het symptoom hebben. Dit kan numeriek aangegeven worden (van 0 (geen last) tot 1) of tekstueel (een van de volgende: `Severe`, `Medium`, `Mild`). De individuele symptomen zijn `LossOfFineDetail`, `LackOfContrast`, `LightSensitivity`, `NeedOfLight`, `CenterVisionLoss`, `PeripheralVisionLoss` en `Hemianopia`. Bij deze laatste kan door middel van het `side`-attribuut aangegeven worden welke helft (`left` of `right`) van het zicht erdoor aangetast wordt. ((Homonieme) hemianopia [65] is het ontbreken van een helft van het beeld door schade aan de optische zenuw, en is dus per definitie maar aan één kant van het zicht.)

Verschillende vormen van kleurenblindheid (`ColorVisionDeficiency`-element) kunnen ook worden voorgesteld. Dit wordt gedaan door aan te geven welke kleur gebrekkig gezien wordt (`DeficiencyType`-element met een van de waarden `Red-Deficiency`, `Green-Deficiency`, `Blue-Deficiency` of `CompleteColorBlindness`). Vervolgens wordt er in het `DeficiencyDegree`-element aangegeven hoe ernstig het gebrek is. Net zoals bij slechtziendheid kan dit met een tekstuele of numerieke aanduiding. De standaard bevat een tabel van kleurenblindheid-aandoeningen, en hun voorgestelde waardes voor `DeficiencyType` en `DeficiencyDegree`.

Mobility Characteristics

Dit onderdeel bevat een heel aantal elementen die te maken hebben met de fysieke geografische locatie en verplaatsing van de gebruiker. Deze kunnen gebruikt worden voor bijvoorbeeld het selecteren van gelokaliseerde multimedia zoals nieuwsberichten en reclame, of voor de bitsnelheid aan te passen aan de grootte van verplaatsing (met de achterliggende redenering dat een hogere snelheid betekent dat de ontvangst slechter is of er vaker hand-offs tussen verschillende draadloze antennes plaatsvinden).

De meest betekenisvolle van de eigenschappen die onder de “Mobility Characteristics” vallen, zijn: de relatieve verandering van richting (`Directivity`), de laatst bekende positie (`LastUpdatePoint`, in lengte- en breedtegraad) en het tijdstip daarvan (`LastUpdateTime`).

Destination

Sterk gerelateerd aan het vorige, dienen deze elementen om de bestemming van de gebruiker over te brengen. Men kan een aankomsttijdstip (`Time`), een bestemming (`Location`), de naam van de bestemming (`DestinationName`) en een bestemmingstype (`DestinationClass`) opgeven. Deze laatste kan men specificeren als vrije tekst (`FreeClass`) of via MPEG-7 (als `PlaceType`). Binnen het `Location`-element verwacht men geografische informatie over de bestemming (coördinaten), terwijl `DestinationClass` over het type bestemming gaat (bijvoorbeeld “Stad” of “Thuis”).

Voorbeeld

Listing 3.7: Een voorbeeld van UED-eigenschappen uit de categorie User Characteristics

```
1 <?xml version="1.0"?>
2 <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
3   xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
4   <Description xsi:type="UsageEnvironmentType">
5     <UsageEnvironment xsi:type="UserCharacteristicsType">
```

```
6       <UserCharacteristics xsi:type="UserInfoType">
7         <UserInfo xsi:type="mpeg7:PersonType">
8           <mpeg7:Name>
9             <mpeg7:GivenName>John</mpeg7:GivenName>
10            <mpeg7:FamilyName>Doe</mpeg7:FamilyName>
11          </mpeg7:Name>
12        </UserInfo>
13      </UserCharacteristics>
14
15      <UserCharacteristics xsi:type="ContentPreferencesType">
16        <UserPreferences>
17          <mpeg7:FilteringAndSearchPreferences>
18            <mpeg7:ClassificationPreferences>
19              <mpeg7:Genre>
20                <mpeg7:Name>Sports</mpeg7:Name>
21              </mpeg7:Genre>
22              <mpeg7:ParentalGuidance>
23                <mpeg7:ParentalRating href=
                      "urn:mpeg:mpeg7:cs:MPAAParentalRatingCS:2001:R">
24                  <mpeg7:Name>R</mpeg7:Name>
25                </mpeg7:ParentalRating>
26              </mpeg7:ParentalGuidance>
27            </mpeg7:ClassificationPreferences>
28          </mpeg7:FilteringAndSearchPreferences>
29        </UserPreferences>
30      </UserCharacteristics>
31
32      <UserCharacteristics xsi:type="PresentationPreferencesType">
33        <ModalityConversion>
34          <GeneralResourceConversions>
35            <Conversion order="1" weight="1.0">
36              <From href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1">
37                <mpeg7:Name>Video</mpeg7:Name>
38              </From>
39
40              <To href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1">
41                <mpeg7:Name>Video</mpeg7:Name>
42              </To>
43            </Conversion>
44
45            <Conversion order="2" weight="1.0">
46              <From href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1">
47                <mpeg7:Name>Video</mpeg7:Name>
48              </From>
```

```
49
50                  <To href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:3">
51                    <mpeg7:Name>Audio</mpeg7:Name>
52                  </To>
53                </Conversion>
54
55                <Conversion order="3" weight="0.1">
56                  <From href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:1">
57                    <mpeg7:Name>Video</mpeg7:Name>
58                  </From>
59
60                  <To href="urn:mpeg:mpeg21:2003:01-DIA-ModalityCS-NS:4">
61                    <mpeg7:Name>Text</mpeg7:Name>
62                  </To>
63                </Conversion>
64              </GeneralResourceConversions>
65            </ModalityConversion>
66          </UserCharacteristics>
67
68          <UserCharacteristics xsi:type="AccessibilityCharacteristicsType">
69            <Visual>
70              <Blindness eyeSide="right" />
71            </Visual>
72          </UserCharacteristics>
73        </UsageEnvironment>
74      </Description>
75  </DIA>
```

Listing 3.7 geeft een voorbeeld van enkele Usage Environment Description-eigenschappen die in de categorie gebruikerskarakteristieken vallen. De eerste informatie die erin wordt tegengekomen is de naam van de gebruiker, in dit geval "John Doe". Die wordt gecodeerd als een MPEG-7 `PersonType`.

De volgende gebruikerseigenschap geeft aan dat John Doe een voorkeur heeft voor items met het genre "Sports". Dit staat vermeld in het subelement `UserPreferences` van het `ContentPreferencesType`. De syntax van dit subelement wordt gedefinieerd in MPEG-7.

Vervolgens zien we de modaliteitsconversievoorkeuren van de gebruiker. De informatie die hier staat is van toepassing op items die oorspronkelijk de modaliteit `Video` hebben, aangezien dat telkens de `From`-modaliteit is. De rangorde van de modaliteiten waarnaar geconverteerd kan worden is `Video`, `Audio`, en dan `Text`. Dit geeft aan dat de gebruiker liefst heeft dat de video-items gewoon video blijven. Indien de adaptatie-engine beslist dat er een conversie nodig is, bijvoorbeeld door bandbreedterestricties, toont het onderstaande voorbeeld dat de gebruiker

geen problemen heeft met een conversie naar de `Audio`-modaliteit: deze heeft rangorde 2 en `weight` 1 (net zoals `Video`). De volgende modaliteit is `Text` met rangorde 3. De gebruiker geeft aan dat conversie naar deze modaliteit een veel groter bezwaar vormt, door middel van het `weight` van 0.1. De MPEG-21 standaard geeft niet aan hoe de adaptatie-engine dit moet interpreteren, maar een optie kan bijvoorbeeld zijn om eerst nog enkele lagere audio-bitrates te proberen alvorens terug te vallen op conversie naar tekst.

Het laatste stuk informatie over de gebruiker dat in dit voorbeeld staat, is dat hij volledig blind is in zijn rechteroog. Dit wordt vrij transparant aangegeven met het `Blindness`-element.

3.2.6 Apparaatmogelijkheden

De eigenschappen en attributen in dit onderdeel stellen informatie over de terminal van de gebruiker voor.

Codec Capabilities

Voor een systeem dat, zoals MPEG-21, probeert de gehele context van multimedia-aanlevering te standaardiseren, zijn de ondersteunde codec-formaten mogelijk één van de belangrijkere stukken informatie. Deze informatie wordt in de UED opgesplitst in formaten die de terminal kan decoderen (`Decoding`-element) en encoderen (`Encoding`-element).

Dan wordt het verder opgesplitst in types; het type van de decoderings- of encoderingsondersteuning dat wordt gespecificeerd wordt met het verplichte `type`-attribuut duidelijk gemaakt. De toegelaten waarden voor dit attribuut zijn `AudioCapabilitiesType`, `GraphicsCapabilitiesType`, `ImageCapabilitiesType`, `SceneGraphCapabilitiesType`, `TransportCapabilitiesType` en `VideoCapabilitiesType`.

Per ondersteunde codec is een instantie van het `Decoding`- of `Encoding`-element vereist. Binnen een `Decoding`- of `Encoding`-element beschrijven de elementen `Format` en `CodecParameter` respectievelijk om welke codec het gaat, en welke eventuele parameters ermee geassocieerd zijn. Het `Format`-element wordt gedefinieerd in MPEG-7 (als type `mpeg7:ControlledTermUseType`). Het heeft als attribuut een URI die de codec uniek aanduidt, en als inhoud een MPEG-7 `mpeg7:Name`-element voor de naam van de codec.

Display Capabilities

Als het apparaat in staat is om iets visueel weer te geven, kunnen de specifieke mogelijkheden daartoe gerepresenteerd worden binnen één of meerdere instanties va het `Display`-element. Het element zelf kent de attributen `bitsPerPixel`, `colorCapable`, `backlightLuminance`,

`refreshRate`, `dotPitch` en `activeDisplay`.

Er kunnen meerdere instanties van het `Display`-element gebruikt worden om meerdere visuele uitvoermogelijkheden (meerdere schermen, een scherm en een projector, ...) voor te stellen.

Ieder `Display`-element mag één of meerdere van de volgende elementen bevatten, behalve `SizeChar`, wat maximaal één keer mag voorkomen.

Resolution Het `Resolution`-element valt binnen het `Display`-element en heeft als verplichte attributen `horizontal` en `vertical`, en optioneel het `activeResolution`-attribuut. Per ondersteunde resolutie dient een nieuw `Resolution`-element gebruikt te worden.

SizeChar Een andere virtuele groottemaat, naast resolutie, is het aantal karakters dat erop kan weergegeven worden. Om dit op te geven dient het `SizeChar`-element, met de attributen `horizontal` en `vertical` om de maten aan te geven.

CharacterSetCode Dit element, waarvan er vanzelfsprekend meerdere binnen hetzelfde `Display`-element kunnen voorkomen, dient om de karaktersets die weergegeven kunnen worden met dit specifieke `Display`, op te sommen. De karaktersets zelf worden opgegeven door middel van een `mpeg7:characterSetCode`.

DisplayDevice Of de `Display` in kwestie stereoscopisch of monoscopisch is, wordt aangegeven met dit element.

RenderingFormat Weten op welke manier het `Display` het beeld weergeeft, kan bijvoorbeeld van nut zijn voor sommige videocoderingsalgoritmes. Voorbeelden hiervan zijn "interlaced", "progressive" en "page-flipping".

Audio Output Capabilities

De geluidsweergavemogelijkheden van een apparaat worden met één element, `AudioOut`, gerepresenteerd. Het heeft geen inhoud, maar wel de volgende optionele attributen:

`samplingFrequency` het aantal metingen per seconde, uitgedrukt in Hertz;

`bitsPerSample` het aantal bits dat gebruikt wordt voor de waarde van één sample voor te stellen;

`lowFrequency` de ondergrens van het frequentiebereik van de audioweergave;

`highFrequency` de bovengrens van het frequentiebereik van de audioweergave;

`dynamicRange` de factor tussen de kleinste en de grootste amplitude die weergegeven kan worden;

`signalNoiseRatio` de signaal-tot-ruis-verhouding van de audioweergave in decibel (dB);

`power` het aantal Watt dat de audioweergave aan energie uitvoert in de vorm van geluidsgolven (als Root Mean Square (RMS));

`numChannels` het aantal uitvoerkanalen dat de audioweergave ondersteunt.

Er kan maar één instantie van het `AudioOut`-element voorkomen in een UED.

User Interaction Input Support

Dit onderdeel representeert, via het `UserInteractionInputSupport`-element, de verschillende manieren waarop een gebruiker kan interageren met het apparaat. Een adaptatie-engine kan deze informatie gebruiken om bijvoorbeeld een alternatieve methode van tekstinvoer te presenteren, of geen tekstinvoer te vereisen, indien er geen woorden of letters op het apparaat kunnen worden ingevoerd.

Indien er gehele woorden en zinnen ingevoerd kunnen worden, kan dit aangeduid worden met het `StringInput`-element, dat een Booleaanse waarde als inhoud moet hebben. Het `KeyInput`-element doet hetzelfde voor individuele letters, en de aanwezigheid van een microfoon wordt op dezelfde manier via het `Microphone`-element duidelijk gemaakt.

Aanwijsapparaten kennen verschillende klasses, die als subelement van het `PointingDevice`-element aan bod komen. De aanwijsapparaten die onder het `Pen`- of `Tablet`-element vallen, kunnen hun resolutie in pixels opgeven via het `resolution`-attribuut dat bij dat element hoort. Als het om een aanwijsapparaat gaat dat onder de elementen `Mouse` of `Trackball` valt, zijn er drie attributen voorhanden: `resolution` (in pixels), `buttons` en het Booleaans attribuut `scrollwheel`.

Power Characteristics

In het `Power`-element wordt informatie omtrent het energieverbruik en resterende energie ter beschikking van het apparaat opgeslagen. Dit gebeurt met de attributen `averageAmpereConsumption` (in Ampères), `batteryCapacityRemaining` (in Ampère-uur) en `batteryTimeRemaining` (in seconden).

Storage Characteristics

De eigenschappen van de opslagruimte van het apparaat krijgen een plaats in het `Storage`-element. Er is geen manier om aan te geven of dit over tijdelijk, vluchtig, permanent, of nog een ander type opslagruimte gaat. De volgende attributen zijn beschikbaar: `inputTransferRate` en `outputTransferRate` (in megabytes per seconde), `size` (in megabytes) en `writable` (als Booleaanse waarde).

Data Input/Output Characteristics

Voor meer algemene informatie omtrent data-in- en uitvoer van het apparaat wordt het `DataIO`-element voorzien. Deze informatie kan als hint dienen bij het bepalen of een bepaald systeem in staat is om zeer hoge kwaliteit multimedia (met een hoge bitrate) te verwerken. De eigenschappen die voorgesteld kunnen worden zijn de breedte van de invoer/uitvoerbus in bits (`busWidth`-attribuut), de doorvoersnelheid van de bus in megabytes per seconde (`transferSpeed`-attribuut), het aantal apparaten dat maximaal op de bus aangesloten kan worden (`maxDevices`) en het huidig aantal apparaten aangesloten op de bus (`numDevices`).

Voorbeeld

Een voorbeeld van een Usage Environment Description die enkele eigenschappen uit de categorie apparaatmogelijkheden gebruikt, wordt gegeven in listing 3.8. Het eerste deel in deze listing gaat over de Codec Capabilities van het apparaat. De ondersteunde formaten (in de `Format`-elementen) worden aangeduid met URI's die in MPEG-7 gedefinieerd worden, en een naam. De formaten worden ook gegroepeerd per type, zoals in het onderstaand voorbeeld de groepen audio en video voorkomen. Deze UED geeft aan dat het apparaat het audioformaat dat geïdentificeerd wordt door de URI `urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4` kan decoderen, dat als alternatieve naam "MP3" heeft. Ook het videoformaat met URI `urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2` en alternatieve naam "MPEG-4 Visual Simple Profile @ Level 1" wordt ondersteund.

Vervolgens staat er informatie omtrent de weergavemogelijkheden van het apparaat. De eerste twee elementen in dit stuk (de twee `Display`-elementen) geven aan dat er twee schermen beschikbaar zijn op dit apparaat (voor ieder scherm één element). Het eerste scherm ondersteunt twee resoluties, aangegeven door de twee `Resolution`-elementen binnen het `Display`-element. Deze zijn 1920×1200 (door het `activeResolution`-attribuut aangeduid als de momenteel actieve resolutie) en 960×600. De kleurdiepte van het scherm wordt door het `bitsPerPixel`-attribuut aangegeven als 24. Het tweede scherm heeft een blijkbaar veel kleinere resolutie (300×50), heeft een kleurdiepte van slechts 1 bit per pixel (wat betekent dat het een monochroom scherm is), en is niet in staat om kleur weer te geven (aangeduid door

Listing 3.8: Een voorbeeld van UED-eigenschappen uit de categorie Terminal Capabilities

```
1  <?xml version="1.0"?>
2  <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
3    xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
4    <Description xsi:type="UsageEnvironmentType">
5      <UsageEnvironment xsi:type="TerminalCapabilitiesType">
6        <TerminalCapabilities xsi:type="CodecCapabilitiesType">
7          <Decoding xsi:type="AudioCapabilitiesType">
8            <Format href="urn:mpeg:mpeg7:cs:AudioCodingFormatCS:2001:4.4">
9              <mpeg7:Name xml:lang="en">MP3</mpeg7:Name>
10           </Format>
11         </Decoding>
12
13         <Decoding xsi:type="VideoCapabilitiesType">
14           <Format
                 href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:3.1.2">
15             <mpeg7:Name xml:lang="en">MPEG-4 Visual Simple Profile @
                   Level 1</mpeg7:Name>
16           </Format>
17         </Decoding>
18       </TerminalCapabilities>
19
20       <TerminalCapabilities xsi:type="InputOutputCapabilitiesType">
21         <Display bitsPerPixel="24">
22           <Resolution horizontal="1920" vertical="1200"
                 activeResolution="true" />
23
24           <Resolution horizontal="960" vertical="600" />
25         </Display>
26
27         <Display bitsPerPixel="1" colorCapable="false">
28           <Resolution horizontal="300" vertical="50"
                 activeResolution="true" />
29         </Display>
30
31         <AudioOut bitsPerSample="16" numChannels="2" />
32
33         <UserInteractionInputSupport>
34           <Microphone>true</Microphone>
35
36           <PointingDevice>
37             <Mouse buttons="3" scrollwheel="true" />
38           </PointingDevice>
39         </UserInteractionInputSupport>
40       </TerminalCapabilities>
41
42       <TerminalCapabilities xsi:type="DevicePropertyType">
43         <Power batteryTimeRemaining="3600" />
44       </TerminalCapabilities>
45     </UsageEnvironment>
46   </Description>
47 </DIA>
```

de `false`-waarde van het `colorCapable`-attribuut). Een mogelijk voorbeeld hiervan is een laptop met een groot scherm, en aan de buitenkant een klein schermpje om een melding bij nieuwe berichten op weer te geven.

De audioweergavemogelijkheden van het apparaat in kwestie worden opgesomd in de attributen van één element, `AudioOut`. De audiohardware heeft een sample-'resolutie' van 16 bits (dit wil zeggen, de amplitude van één meting van de geluidsgolf kan weergegeven worden met een maximum-'nauwkeurigheid' van 16 bits). De audioweergave gebeurt door middel van 2 audiokanalen. Dit zou een simpele stereoluidsprekers-opstelling kunnen zijn.

In het volgende deel worden er enkele eigenschappen met betrekking tot gebruikersinvoer vermeld. Het `Microphone`-element bevat de Booleaanse waarde `true` wat aangeeft dat het apparaat over een microfoon beschikt. In het `PointingDevice`-element wordt één aanwijsapparaat aangegeven. Door het `Mouse`-element te gebruiken geeft het document aan dat het om een muis-achtige gaat, en waardes van de attributen `buttons` en `scrollwheel` geven aan dat het aanwijsapparaat in kwestie 3 knoppen en een scrollwheel heeft. Of het om een eigenlijke computermuis gaat of een vervanger ervan zoals een touchpad (met drie werkelijke of geëmuleerde knoppen en een werkelijk of geëmuleerd scrollwheel), kan in een UED-document niet worden aangeduid.

Als laatste komen we een deel tegen met de weinigzeggende typenaam `DevicePropertiesType`. Het `Power`-element erin met het attribuut `batteryTimeRemaining` spreekt meer voor zichzelf. De waarde van het attribuut wordt uitgedrukt in seconden, wat aangeeft dat er geschat wordt dat er nog 3600 seconden oftewel één uur aan batterijstroom voorhanden is.

3.2.7 Netwerkkarakteristieken

De explosieve groei van het Internet was één van de motivaties achter MPEG-21. Informatie over de netwerkverbinding van het apparaat mag dan natuurlijk niet ontbreken. Deze kennis kan van groot belang zijn voor bijvoorbeeld streaming media servers.

Dit onderdeel van de UED is veel kleiner dan de twee voorgaanden. Er is één hoofdelement, `NetworkCharacteristics`, dat zowel gebruikt wordt om de `NetworkCapabilityType`-attributen een plaats te geven, als om de `NetworkConditionType`-elementen in te plaatsen. Dit hoofdelement kan meerdere keren voorkomen, bijvoorbeeld een eerste keer met `NetworkCapabilityType`-attributen, en een tweede keer met `NetworkConditionType`-elementen die erbinnen vallen. (Deze kunnen niet samen in één hoofdelement gezet worden.)

De `NetworkCapabilityType`-attributen handelen over de statische eigenschappen van het netwerk; bijvoorbeeld of pakketten altijd in de zendvolgorde zullen worden afgeleverd. De `NetworkConditionType`-elementen, daarentegen, gaan over de huidige toestand van het netwerk, die van interval tot interval kan fluctueren.

NetworkCapabilityType

Dit type definieert attributen die in een instantie van het `NetworkCharacteristics`-element thuishoren. De attributen zijn de onderstaande:

`maxCapacity` De maximumcapaciteit (bandbreedte) van de verbinding, uitgedrukt in bits per seconde;

`minGuaranteed` De minimale bandbreedte van de netwerkverbinding die altijd beschikbaar is, uitgedrukt in bits per seconde;

`inSequenceDelivery` Geeft aan of de netwerkverbinding data-eenheden in volgorde aflevert. Er wordt in de standaard niet vermeld of dit als 'waar' gemarkeerd mag worden indien de client enkel van zijn lokale verbinding weet dat het geldt (bijvoorbeeld als het eerste stuk van de netwerkverbinding deze eigenschap wel heeft, maar er dan een router volgt waarna de eigenschap onbekend is);

`errorDelivery` Een Booleaanse waarde die 'waar' is als data-eenheden met fouten in afgeleverd worden;

`errorCorrection` Als data-eenheden met fouten in gecorrigeerd worden door het netwerk, dient deze Booleaanse waarde 'waar' te zijn.

NetworkConditionType

Net zoals het bovenstaande, horen de elementen die gedefinieerd worden door dit type in een `NetworkCharacteristics`-element thuis. Het type definieert drie elementen.

`AvailableBandwidth` is een leeg element dat vier attributen heeft:

`minimum` Minimum aantal bits per seconde beschikbaar in het opgegeven interval;

`maximum` Maximum aantal bits per seconde beschikbaar in het opgegeven interval;

`average` Gemiddeld aantal bits per seconde beschikbaar in het opgegeven interval;

`interval` Het opgegeven interval gebruikt voor de berekening van de bovenstaande drie attributen, in milliseconden.

Om de vertraging van data-eenheden op het netwerk aan te geven, wordt het `Delay`-element aangereikt, met de volgende drie attributen:

`packetTwoWay` Beschrijft de heen-en-terug vertraging (*round trip delay*) van een data-eenheid op de netwerkconnectie, in milliseconden;

`packetOneWay` Heeft als waarde de vertraging in milliseconden voor een data-eenheid in één richting;

`delayVariation` Dit is het verschil in vertraging in enkele richting (in tegenstelling tot heen-en-terug) van twee opeenvolgende data-eenheden binnen dezelfde data stream.

Het laatste element, `Error`, gaat over de foutkarakteristieken van de netwerkverbinding en kent als attributen `packetLossRate` (een getal tussen 0 en 1 dat het percentage verloren data-eenheden weergeeft), en `bitErrorRate`. Dit laatste is de ratio van foute bits uitgedrukt als de waarde van n in de formule 10^{-n}.

Voorbeeld

Listing 3.9: Een voorbeeld van UED-eigenschappen uit de categorie Network Characteristics

```
1  <?xml version="1.0"?>
2  <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
3     xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
4    <Description xsi:type="UsageEnvironmentType">
5      <UsageEnvironment xsi:type="NetworkCharacteristicsType">
6        <NetworkCharacteristics xsi:type="NetworkCapabilityType"
             minGuaranteed="0" inSequenceDelivery="false" />
7
8        <NetworkCharacteristics xsi:type="NetworkConditionType">
9          <Delay packetOneWay="100" delayVariation="40" />
10
11         <Error packetLossRate="0.0003" />
12       </NetworkCharacteristics>
13     </UsageEnvironment>
14   </Description
15 </DIA>
```

Het document weergegeven in listing 3.9 is een Usage Environment Description-document met enkele netwerkkarakteristieken. Zoals uitgelegd in sectie 3.2.7 bestaat dit uit twee delen: netwerkmogelijkheden (type `NetworkCapabilityType`) en de netwerktoestand (type `NetworkConditionType`). Beide types worden ondergebracht in (verschillende instanties van) het element `NetworkCharacteristics`. De eerste maal dat we het element tegenkomen, is er sprake van het eerste type. De attributen van het element geven aan dat er op dit netwerk geen gegarandeerde minimumdoorvoersnelheid bestaat (minGuaranteed="0"), en dat pakketten niet altijd in de zendvolgorde worden ontvangen (inSequenceDelivery="false").

De tweede maal dat het `NetworkCharacteristics`-element in het voorbeeld voorkomt, is het van type `NetworkConditionType`. Het eerste subelement ervan, `Delay`, specificeert met het

attribuut `packetOneWay` dat de tijd die een pakket nodig heeft om van de zender naar de ontvanger te geraken, 100 milliseconden bedraagt. Het `delayVariation`-attribuut van dit `Delay`-element geeft aan dat er een variatie (*jitter*) van 40 milliseconden ten opzichte van de `packetOneWay`-waarde bestaat. Ten slotte geeft het `packetLossRate`-attribuut van het `Error`-element aan dat er een kans van 0.03% bestaat dat een verzonden pakket verloren gaat.

3.2.8 Omgevingskarakteristieken

Het laatste onderdeel binnen de UED gaat over eigenschappen van de natuurlijke of fysieke omgeving van de gebruiker. Net zoals het vorig onderdeel van de UED is dit stuk vrij klein. Er is opnieuw één hoofdelement, `NaturalEnvironmentCharacteristics`, dat als basis dient voor `LocationType`, `TimeType`, `AudioEnvironmentType` en `IlluminationCharacteristicsType`.

Het eerste type, `LocationType`, definieert een `Location`-element, dat gedefinieerd wordt als een `mpeg7:PlaceType`. Het tweede type, `TimeType`, definieert een `Time`-element, dat ingevuld wordt als een `mpeg7:TimeType`. Het `AudioEnvironmentType`-type bevat twee optionele subelementen. `NoiseLevel` geeft het achtergrondgeluid van de omgeving in decibels (dB). `NoiseFrequencySpectrum` bevat een meer gedetailleerde analyse, opgedeeld in 33 frequentiebanden, van het achtergrondgeluid, allemaal opnieuw opgegeven in dB. De visuele achtergrondomstandigheden worden gecodeerd in het `IlluminationCharacteristicsType` met behulp van twee elementen. Het eerste, `TypeOfIllumination`, geeft het type van de globale belichting. Dit kan op één van de twee volgende manieren: als kleurtemperatuur (`ColorTemperature`; het bepalen van deze waarde is vastgelegd in MPEG-7) of als chromaticiteit (`Chromaticity`). Het tweede element, `Illuminance`, beschrijft de globale illuminantie, in Lux.

Voorbeeld

Het voorbeeld van omgevingskarakteristieken, gegeven in listing 3.10, is vrij duidelijk. Het eerste stuk informatie zit vervat in het `Location`-element en geeft de geografische positie en regio weer als zijnde respectievelijk $50.93\,^{\circ}N, 5.34\,^{\circ}E$ en België (aangeduid met de code `BE`).

Het volgende deel van het voorbeeld geeft de tijd aan. Deze staat in een `mpeg7:TimePoint`-element binnen het `Time`-element, maar het formaat van de waarde zelf is welbekend als het uitgebreide datum- en tijdsrepresentatieformaat uit zowel RFC 3339 [66] als ISO 8601 [67].

Listing 3.10: Een voorbeeld van UED-eigenschappen uit de categorie Natural Environment Characteristics

```
1  <?xml version="1.0"?>
2  <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"
3     xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
4     <Description xsi:type="UsageEnvironmentType">
5       <UsageEnvironment xsi:type="NaturalEnvironmentCharacteristicsType">
6         <NaturalEnvironmentCharacteristics xsi:type="LocationType">
7           <Location>
8             <mpeg7:GeographicPosition>
9               <mpeg7:Point latitude="50.93" longitude="5.34" />
10            </mpeg7:GeographicPosition>
11
12            <mpeg7:Region>BE</mpeg7:Region>
13          </Location>
14        </NaturalEnvironmentCharacteristics>
15
16        <NaturalEnvironmentCharacteristics xsi:type="TimeType">
17          <Time>
18            <mpeg7:TimePoint>2008-08-21T05:10+02:00</mpeg7:TimePoint>
19          </Time>
20        </NaturalEnvironmentCharacteristics>
21      </UsageEnvironment>
22    </Description>
23 </DIA>
```

3.3 Zelf-gedefinieerd formaat

Naast de bovenstaande, door externe organisaties gedefinieerde formaten, is het ook mogelijk om zelf een formaat op te stellen om de informatie over te brengen. Een zelf-gedefinieerd formaat kan vele vormen aannemen, waarin twee hoofdcategorieën te onderscheiden zijn: binaire formaten en tekstuele formaten. Hoewel het niet de bedoeling is een allesomvattende vergelijking tussen alle vormen en mogelijkheden van diverse zelf-gedefinieerde formaten te maken, worden enkele relevante punten met betrekking tot de twee hoofdcategorieën opgesomd.

3.3.1 Binaire formaten

Aan de ene kant zijn er binaire formaten, waarin informatie geplaatst wordt in de vorm die ze aanneemt in het geheugen van de computer. Een voordeel van formaten in deze categorie is vaak compactheid in vergelijking met tekstuele formaten. Mits de ontvangende applicatie het formaat kent, kan het inlezen van de informatie uit een binair formaat zeer snel gebeuren. Dit heeft als oorzaak het feit dat er geen vertaling hoeft te gebeuren tussen het opslag-formaat en het formaat dat het geheugen van de applicatie gebruikt.

Er hangen echter ook grote nadelen vast aan binaire formaten. Het representatieformaat van informatie in het geheugen van een computer is niet voor alle computers gelijk. Een bekend probleem is de "Endianness": of de meest significante bit danwel de minst significante bit van een getal eerst wordt geplaatst ("Big Endian" en "Little Endian", respectievelijk). Een ander probleem is de grootte van eenzelfde stuk informatie op verschillende computersystemen. Sommige systemen gebruiken voor een getal te representeren standaard een geheugenvak van 32 bits lang, terwijl andere systemen daar 16, 64, of nog andere lengtes voor gebruiken.

3.3.2 Tekstuele formaten

Aan de andere kant zijn er tekstuele formaten, waarin een tekstuele representatie van de informatie geplaatst wordt. Een onmiddellijk voordeel is dat voor menselijke lezers de informatie in haar opgeslagen vorm, hoewel niet noodzakelijk meteen compleet zinnig, meer betekenis heeft dan diezelfde informatie in binaire vorm.

Een nadeel van tekstuele representatie is de grootte. Omdat de tekstuele representatie op zich weer een onderliggende binaire vorm heeft, kan deze nooit minder bits gebruiken dan de informatie in binaire vorm. Zo heeft het getal 1431655765 op de meeste computersystemen een grootte van 32 bits in het geheugen, maar het tekstequivalent "1431655765" heeft doorgaans acht bits per karakter nodig; in dit voorbeeld dus 80 bits.

3.4 eXtensible Markup Language

Zoals de standaarden die eerder in dit hoofdstuk besproken werden, kunnen doen vermoeden, is XML een populaire basis voor informatie-uitwisselingsformaten. Hoewel in sectie 3.1.3 over RDF al geraakt werd aan XML, is het de bedoeling hier enkele concrete eigenschappen en voor- en nadelen ervan uit de doeken te doen. Technisch gezien valt XML in de categorie van tekstuele formaten, maar het biedt enkele interessante voordelen ten opzichte van een arbitrair, zelf-gedefinieerd tekstueel formaat. XML is strikt gezien een ontwerp voor een boomstructuur, welke in tekstvorm omgezet (oftewel "geserialiseerd") kan worden. Zo kan de structuur van een XML-document vastgelegd worden door middel van een Document Type Definition (DTD, vastgelegd in dezelfde standaard [39] als XML) of een XML Schema [68]. Hierin kan vastgelegd worden welke types data (XML-elementen, XML-attributen, cijfers, tekst, ...) waar in het document mogen voorkomen. Er bestaan twee 'gradaties' van controle op een XML-document. Als een eerste stap kan er geverifieerd worden dat een XML-document *well formed* is. Dit wil zeggen dat alle XML-elementen die geopend worden ook gesloten worden, attribuut-waardes beginnen en eindigen met een "-karakter, enzovoort. Wanneer zowel het schema en het document voorhanden zijn, kan een ontvangende applicatie het XML-document *valideren*, wat betekent dat gecontroleerd wordt dat het document het schema in kwestie respecteert. Dit betekent dat er nergens tekst staat waar enkel cijfers toegelaten zijn, enzovoort. Er is allerhande software voorhanden die XML-validatie kan uitvoeren, zoals de gratis XML-parser Xerces-C++ [69]. Zowel controle op wellformedness en complete validatie zorgen ervoor dat de ontvanger veel meer zekerheid heeft over de ingevoerde documenten.

Een ander voordeel is dat er allerhande technologieën ontwikkeld zijn om het gebruik van XML te vergemakkelijken. Zo is de XPath-taal [70] ontwikkeld om specifieke informatie uit een XML-document te halen. De XPath-taal laat in principe toe om de boomstructuur van het XML-document te navigeren.

Een vaak herhaald nadeel aan XML is dat het al snel grote documenten oplevert, maar door het feit dat het structuren uitdrukt, kan het vaak sterk gecomprimeerd worden, waarover in sectie 3.5 meer.

Als testament van de uitgebreide mogelijkheden en het wijde gebruik en ondersteuning van XML, kan gekeken worden naar onder andere software die XML gebruikt als het opslagmechanisme van databases (zoals eXist [71] en Xindice [72]). Ook meer traditionele databasesystemen bieden steeds vaker specifieke XML-functionaliteit, zoals het XML-datatype [73] van PostgreSQL [74] en de bijhorende XML-functionaliteit [75] (die onder andere XPath-expressies ondersteunt), en de pureXML-module [76] van DB2 [77].

3.5 Compressie

Multimediabronnen zoals audio en video worden voor transport over netwerken zoals het Internet meestal in een sterk gecomprimeerd formaat, zoals JPEG, MPEG-4 Video of MP3, geplaatst. Vaak gebeurt dit met bandbreedtebeperkingen in het achterhoofd. Deze efficiënte formaten staan in sterk contrast met het tekstuele XML-formaat, wat meestal zeer verboos is, en dus de bandbreedtevereisten (relatief sterk) doet stijgen.

Het leesbare aspect van XML heeft voordelen, bijvoorbeeld bij het manueel zoeken naar fouten. Volgens de doelstellingen uitgelijnd in de XML-standaard [39], was en is bondigheid bij het ontwikkelen van XML slechts van minimaal belang. Maar bij verzending over een netwerk wordt er meestal meer waarde gehecht aan bondigheid. Door de XML-documenten te verkleinen, worden ze ook meer geschikt voor mobiele apparaten die hiermee opslagcapaciteit en batterijkracht (voor het verzenden en ontvangen, bijvoorbeeld) kunnen besparen. Ook grote verzamelingen van XML-documenten, zoals sommige databases, zouden voordeel hebben bij kleinere XML-documenten.

Zowel de CC/PP-profielen als de Usage Environment Description van MPEG-21 die in dit hoofdstuk werden besproken, worden uitgedrukt in XML-formaat. Het kan dus lonen om methodes te bekijken die de transmissietijd of grootte (in bytes bekeken) van XML-documenten verkleinen.

3.5.1 Binaire XML volgens het W3C

Aangezien XML een standaard is van het W3C, is hun kijk op een binair formaat ervoor interessant. In 2004 richtte het W3C de "XML Binary Characterization Working Group" [78] op om te achterhalen of er nood was aan een binaire XML-representatie. Na een jaar kwam deze werkgroep tot de volgende conclusies:

- Binaire XML is nodig: er werden use cases [79] gevonden die de nood aantoonden;
- Binaire XML is haalbaar: er werd een document [80] opgesteld dat de verlangde eigenschappen identificeert;
- Binaire XML moet gestandaardiseerd worden door het W3C: er bestonden al vele verschillende, 'concurrerende' formaten voor binaire XML. Het idee achter de XML-standaard om uitwisseling van data tussen verschillende systemen te vergemakkelijken, wordt teniet gedaan indien een systeem de binaire representatie van een binnenkomend XML-document niet kent.
- Binaire XML moet integreren met XML: er mogen geen veranderingen binnen XML zelf nodig zijn.

Daarop werd in 2005 de "Efficient XML Interchange Working Group" [81] opgericht om een efficiënt uitwisselingsformaat voor XML te ontwikkelen. Deze werkgroep heeft ondertussen enkele drafts [82] geproduceerd, maar een definitieve standaard wordt niet voor 2009 verwacht.

3.5.2 Algemene compressie

Een eerste optie is om generieke compressiemethodes toe te passen. XML is, zoals hierboven al aangehaald, een tekstformaat, en kan met algemene compressiemethodes al veel kleiner gemaakt worden. Dit soort methodes kan ingedeeld worden in twee soorten: aritmetische en woordenboek-gebaseerde.

Aritmetische hebben vaak relatief veel geheugen en rekentijd nodig. Ze steunen op statistische methodes. Voorbeelden van dit soort compressiemethodes zijn CACM3, PAQ en PPM.

Woordenboekgebaseerde methodes worden het meest wijd gebruikt. Ze zijn vaak gebaseerd op het Lempel-Ziv-algoritme. Dit werkt door een stuk data dat hetzelfde is als een ander stuk dat ervoor komt, te vervangen door een verwijzing naar het eerdere stuk. Voorbeelden van woordenboekgebaseerde compressors zijn PK-Zip, Gzip en Bzip.

Een interessant feit is dat GZip een stream-compressiemethode is. Afhankelijk van de venstergrootte (verwijzingen zijn enkel binnen het venster mogelijk) die bij het comprimeren gebruikt werd, kan men het begin van gecomprimeerde data al decomprimeren zonder over het eind te moeten beschikken.

3.5.3 Binaire XML-formaten

Technisch gezien zijn de vorige methodes ook binair, maar de formaten die in deze sectie vallen, zijn eerder binair gestructureerde representaties van de XML-documenten. Ze ondersteunen bijvoorbeeld soms ook XML-gerelateerde operaties zonder de data terug te moeten omzetten naar tekstvorm.

"Wireless Application Forum (WAP) Binary XML" (WBXML, [83]) werd ontwikkeld door de Open Mobile Alliance. Het wordt momenteel optioneel gebruikt in onder andere SyncML [84], een XML-gebaseerde standaard voor gegevensuitwisseling van mobiele apparaten, ontwikkeld door het SyncML Initiative, dat ondertussen deel uitmaakt van de Open Mobile Alliance.

3.5.4 Schema-bewuste XML-compressie

Deze categorie is, zoals de naam al zegt, bij compressie bewust van het schema waar het XML-document in kwestie aan voldoet. Door via het XML-schema informatie te hebben over de structuur van het document, hopen deze methodes een efficiëntere vorm van binaire

representatie van XML te bereiken. Een vereiste is wel dat het schema beschikbaar is bij zowel het comprimeren als het decomprimeren.

Kennis van het schema laat toe redundante informatie zoals voluit geschreven elementnamen weg te laten. Als voorbeeld: wanneer een element `a` enkel elementen `b` en `c` en attributen `d` en `e` kan bevatten volgens het schema, hoeft de binaire representatie niet de naam van het element of attribuut op te slaan. Een verwijzing naar om welk van de mogelijke opties het gaat (die bijvoorbeeld ge-enumereerd zijn), is voldoende.

Wanneer het XML-schema het datatype (integer, string, ...) van een waarde aangeeft, kan deze vorm van compressie hiermee rekening houden om de waarde efficiënter voor te stellen. Een integer kan dan bijvoorbeeld als binair getal opgeslagen worden, wat minder plaats inneemt dan de string-representatie van dat getal.

MPEG Binary Format for Metadata (BiM)

Een voorbeeld van een schema-bewuste compressiemethode is MPEG BiM. Dit formaat werd door de Moving Picture Expert Group (MPEG) ontwikkeld als deel van MPEG-7 [58] en wordt hergebruikt in MPEG-21 [59]. Het kan toegepast worden op XML-documenten, zoals Digital Item Declarations, metadata ingesloten in Digital Items en ook de Usage Environment Description (UED) tools. Het laat toe om in de binaire representatie van een XML-document bepaalde onderdelen op te vragen en daar bewerkingen op uit te voeren, zonder het document eerst terug naar tekstuele vorm om te zetten.

Eén van de mogelijkheden van MPEG BiM is om een XML-document te 'streamen'. Het document wordt ingedeeld in zogenaamde Access Units (AU's) welke op hun beurt ingedeeld zijn in Fragment Update Units (FUU's). Iedere AU kan los van de andere AU's gedecodeerd of gemanipuleerd worden. Iedere FUU bevat het volgende:

FU-commando dit is één van de volgende: `add`, `delete`, `replace` of `reset`. Dit geeft aan wat deze Fragment Update dient te doen met de vorige reeds ontvangen Updates;

FU-context dient om de plaats van deze Update in het XML-document te bepalen;

FU-payload bevat de eigenlijke data die aangeduid wordt door de FU-context en volgens het FU-commando verwerkt dient te worden.

Dit kan bijvoorbeeld gebruikt worden om ondertitels van een video, die in één groot XML-document staan, in kleine incrementele updates (het FU-commando `add`) met een video stream mee te sturen. Een ander voorbeeld is wanneer een klein deel of één waarde van een XML-document wordt aangepast. Dan hoeft enkel een FUU met het FU-commando `replace` doorgestuurd te worden, en niet het hele XML-document.

3.5.5 Resultaten

In de paper waarin XMill [85] wordt voorgesteld, wordt meteen ook een vergelijking gemaakt tussen XMill en de algemene gzip-compressiemethode op vlak van compressietijd, decompressietijd en resultaatgrootte. XMill is een hybride aanpak; het XML-document wordt ontleed waarna de waardes van de elementen en attributen worden samengezet per datatype in 'streams'. Deze streams worden dan individueel met behulp van gzip gecomprimeerd. De compressietijd varieert weinig tussen de twee vergeleken compressiemethodes. Interessant om op te merken is dat een complexer XML-document een snellere compressie met XMill tot gevolg heeft. De verklaring die hiervoor gegeven wordt is dat het ontleden van de XML in dit geval meer voordeel oplevert, en proportioneel kleinere 'streams' oplevert. Hierdoor is de algemene-compressie-stap veel korter. Ook de decompressietijd varieert weinig tussen de twee methodes.

Uit de data in deze paper blijkt dat voor kleine bestanden de algemene compressiemethode (gzip) beter geschikt is, en er een kantelpunt in XML-documentgrootte ergens tussen 10 en 100 kilobyte is waarna XMill beter presteert. De behaalde compressieratio's van XMill zijn in het beste geval (een dataset van meer dan 100 megabyte) dubbel zo goed als die van gzip (2.5% en 5% van de originele grootte, respectievelijk).

De studie "XML Sizing and Compression Study For Military Wireless Data" [86] voerde een vergelijking van compressieratio's uit tussen WinZip [87] (als algemene compressiemethode), hun eigen WBXML-implementatie, MPEG BiM en XMill. Verder werd er ook de combinatie van WinZip en MPEG BiM-compressie uitgeprobeerd. Een opmerkelijke vaststelling van WinZip toepassen op de uitvoer van MPEG BiM-compressie is dat dit nooit grotere bestanden als resultaat had, maar soms tot 200% verkleining opleverde. Verder bleek (hun implementatie van) WBXML zeer slecht te presteren, met een resultaat dat meestal meer dan 40% van de grootte van de originele XML bedroeg. De algemene conclusie in deze studie luidt dat voor 'kleine' XML-documenten een XML-specifieke compressie de beste resultaten oplevert, terwijl voor grote documenten een algemene compressiemethode het best is. Het kantelpunt ligt, afhankelijk van de implementaties, ergens tussen 12 en 100 kilobyte. Een uitzondering op deze algemene conclusie wordt gemaakt voor XMill, wat de resultaten van de XMill-paper bevestigt.

3.5.6 Conclusie

In een overzichtspaper over dit onderwerp, "An analysis of XML compression efficiency" [88], luidt de conclusie dat het aangeraden is een algemene compressiemethode te gebruiken indien er geen speciale mogelijkheden vereist zijn, zoals manipulatie van het XML-document in zijn binaire vorm of streaming van het XML-document. Deze conclusie lijkt ook overeen te komen

met de resultaten van de papers besproken in sectie 3.5.5

3.6 Vergelijking van representatieformaten

Er zijn duidelijk een heel aantal mogelijkheden voor het representeren van gebruikersvoorkeuren en apparaateigenschappen. Elk formaat heeft zijn eigen karakteristieken en het is duidelijk dat verschillende formaten met verschillende toepassingsgebieden als uitgangspunt werden ontworpen. Het is dan ook moeilijk om de vergelijking van verschillende formaten uit te voeren zonder een concrete applicatie voor ogen. Toch kan elk formaat aan enkele cruciale vlakken getoetst worden.

3.6.1 Zelf definiëren of externe standaard

Zelfgedefinieerde formaten hebben het voordeel dat ze (mits doordacht ontwerp) perfect op maat gemaakt zijn voor de specifieke applicatie in kwestie, welke dat ook moge zijn. Wanneer iets niet aanwezig is in het formaat, of er een benodigde structuur moeilijk mee uit te drukken valt, kan het formaat zelf aangepast worden aan de noden. Een gerelateerd voordeel is dat ze niets bevatten dat niet nodig is; er is geen sprake van onnodige ballast, zoals de mogelijkheid om eigenschappen uit te drukken die compleet irrelevant zijn.

Externe standaarden (zowel de facto standaarden als die door een standaardenorgaan gedefinieerd) hebben meestal het voordeel dat ze brede ondersteuning genieten. Zo kunnen er al applicaties en gereedschappen bestaan die gebruik maken van de standaard in kwestie, wat het aannemelijk maakt dat de standaard niet compléét onwerkbaar is. Ook kunnnen bestaande applicaties van nut zijn bij het ontwikkelen van een eigen applicatie. Een voorbeeld hiervan is de DELI-library [89] voor het opvragen van CC/PP-profielen. Een ander concreet voorbeeld is dat MPEG-21 UED-profielen gebruikt kunnen worden door een MPEG-21 Digital Item Adaptation-engine, die dus onmiddellijk inzetbaar is wanneer UED als standaard gekozen wordt. Nog een voordeel van een externe standaard is dat deze door meerdere mensen ineen gestoken werd, en misschien al in de praktijk getest werd voordat deze uitgebracht werd, en dus minder snel fouten zal bevatten.

3.6.2 Verkrijgbaarheid

Hoewel het geen rechtstreekse eigenschap van de standaarden zelf is, maakt in sommige situaties de verkrijgbaarheid van de standaard (en eventuele bijhorende documentatie) een groot verschil. Zo is de CC/PP-standaard van het W3C open en gratis, terwijl er voor MPEG-21, afgezien van enkele XML Schema's die voor validatie vereist zijn, betaald dient te worden. Over de verkrijgbaarheid van zelfgedefinieerde formaten valt weinig te zeggen, behalve misschien

dat er zorg gedragen moet worden dat de documentatie met betrekking tot de standaard 'verkrijgbaar' dient te blijven om ten alle tijde het formaat te kunnen interpreteren.

3.6.3 Uitbreidbaarheid

Aan profielen die opgesteld zijn aan de hand van de CC/PP-standaard kan nieuwe, voorheen onbekende, informatie toegevoegd worden door een additioneel vocabularium toe te voegen. In dat vocabularium kunnen de nieuwe eigenschappen ondergebracht worden, terwijl de oude vocabularia van kracht blijven. Op diezelfde manier is het mogelijk een combinatie van bestaande vocabularia te gebruiken om zo tot een adequaat geheel te komen, zonder zelf een vocabularium te definiëren. MPEG-21 UED-profielen maken intensief gebruik van XML Schema, wat als voordeel heeft dat profielen zeer accuraat gevalideerd kunnen worden, maar ook betekent dat de standaard niet uitbreidbaar is [90]. Zelfgedefinieerde formaten zijn in principe uitbreidbaar, hoewel afhankelijk van het ontwerp van het formaat de compatibiliteit met eerdere versies van het formaat verloren kan gaan.

3.6.4 Toepassingsgebied

De achtergrond van CC/PP maakt het duidelijk dat deze standaard, gepubliceerd door het World Wide Web Consortium en gegroeid uit User Agent Profiles, ontworpen is met het www in het achterhoofd. Ook het geassocieerde transportmechanisme, CC/PP over HTTP, laat hier weinig twijfel over bestaan. MPEG-21, aan de andere kant, werd ontwikkeld om een volledig geïntegreerde infrastructuur voor mediadistributie en -consumptie te bouwen. Uit de bespreking van de UED van MPEG-21 (in sectie 3.2.4) valt duidelijk op te maken dat multimedia de focus is van deze standaard. Het toepassingsgebied van zelfgedefinieerde formaten valt doorgaans samen met dat van de applicatie waarvoor de standaard ontworpen werd.

Hoofdstuk 4

Uitbuiting

Wanneer we er vanuit gaan dat gegevens over de gebruiker en het gebruikersapparaat gekend zijn, volgt de vraag op welke manieren men deze informatie wil en kan benutten.

4.1 Doelen

Gebruikers zouden graag willen en hebben meer en meer de verwachting dat ze met de relatief nieuwere en mobielere klasses van apparaten zoals GSM's en PDA's hetzelfde kunnen bereiken als met hun oudere maar vaak krachtigere tegenhangers, zoals desktop-PC's. De capaciteiten van deze apparaten verschillen sterk, en veranderen (verbeteren) continu. Het doel is dus om hetgene dat met één type apparaat kan uitgevoerd worden (zoals film kijken, communiceren, teksten schrijven, ...), mogelijk te maken op zoveel mogelijk verschillende types apparaten. Dit doel wordt in de literatuur vaak omschreven met de termen *ubiquitous computing* en *pervasive computing* [91].

4.1.1 Universal Multimedia Access

Niet zozeer equivalent aan *ubiquitous computing*, maar zeker een onderdeel ervan, is *universal multimedia access* (UMA, [92]). UMA is een concept dat staat voor naadloze, 'alomtegenwoordige' toegang tot afbeeldingen, video, audio en andere multimediabronnen. Dit moet mogelijk zijn zonder beperkingen en via elk soort apparaat en elk soort netwerkverbinding. Dit betekent dat de multimediabronnen geadapteerd en eventueel gepersonaliseerd moeten worden om een divers bereik aan apparaten, netwerken en gebruikersvoorkeuren te accommoderen. Een grote drijfveer achter UMA is om mobiele of andere apparaten met beperkte rekenkracht en netwerkmogelijkheden in staat te stellen multimediabronnen te bekijken, te gebruiken of te consumeren [93, 94].

4.2 Locatie en methode

4.2.1 Lokaal

De benutting moet niet noodzakelijk op een extern systeem plaatsvinden. De informatie die in listing 2.1 in hoofdstuk 2 verkregen wordt, namelijk of de processor van het apparaat in kwestie MMX [10] ondersteunt, is bijvoorbeeld vooral op het apparaat zelf van nut. Wanneer een bepaald algoritme bestaat in een versie met en een versie zonder MMX-instructies, is deze informatie over het apparaat van groot nut bij het bepalen welke versie van het algoritme uitgevoerd kan worden.

Als een applicatie weet welke snelheid de netwerkverbinding van het apparaat heeft, kan dit een factor zijn bij de keuze tussen verschillende versies van een netwerkresource met verschillende *bit rates*. Wanneer de vertraging (*delay*) en *jitter* van de netwerkverbinding gekend is, kan er een geïnformeerde beslissing gemaakt worden over een adequate buffergrootte voor het afspelen van de netwerkresource. Een voorbeeld hiervan is wanneer een gebruiker via een website filmtrailers wil bekijken. Vaak worden deze in verschillende versies met verschillende bit rates aangeboden. Wanneer de browser beschikt over de hiervoor opgesomde informatie, eventueel gecombineerd met de voorkeur van de gebruiker omtrent de kwaliteit van de filmtrailer, kan de browser automatisch de meest geschikte versie kiezen en een adequate buffergrootte bepalen voor het afspelen ervan.

4.2.2 Op afstand

De in het bovenstaand voorbeeld beschreven aanpak gaat er van uit dat er verschillende versies van dezelfde netwerkresource beschikbaar zijn, en dat er daar minstens één en liefst meerdere binnen de capaciteiten van het apparaat vallen. Om voor alle mogelijke combinaties van eigenschappen van apparaten een versie van de resource te genereren die daar perfect (of redelijk) bij aansluit, zou een grote hoeveelheid 'instanties' van eenzelfde resource vereisen. Dit stelt grotere opslageisen aan de netwerkbron en vereist meer moeite van de *content provider*.

Indien de voorkeuren van de gebruiker en de eigenschappen van het gebruikersapparaat gecommuniceerd worden met de bron van de resource die opgevraagd wordt, kan deze bron zelf *on the fly* een versie van de resource genereren (of een bestaande versie aanpassen). De 'intelligentie' die ingezet wordt voor het kiezen of aanmaken van een geschikte versie wordt verplaatst van het gebruikersapparaat naar een extern systeem. In het ideale geval kan dit externe systeem een instantie van de resource genereren die binnen de capaciteiten van het gebruikersapparaat valt, voldoet aan de voorkeuren van de gebruiker, en gebruik maakt van eventuele extra mogelijkheden van het gebruikersapparaat. Deze aanpak brengt enkele bijko-

mende voordelen met zich mee. Wanneer een GSM een video aanvraagt in zeer lage resolutie, kan de bron de oorspronkelijke versie schaleren om hieraan te voldoen. De originele, grotere versie hoeft niet verstuurd te worden, wat betekent dat de transmissiegrootte en -duur verkleint, en het gebruikersapparaat wordt met minder zware decoderings- en schaleringsactiviteiten (als deze laatste al niet weg valt) belast.

Deze adaptatierol kan eventueel ook vervuld worden door een proxyserver die zich in het netwerk tussen de bron en het gebruikersapparaat bevindt. Een voordeel van die situatie zou zijn dat de bron zelf niet in staat hoeft te zijn om de resource aan te passen. Wanneer bijvoorbeeld een GSM alle video-aanvragen via zulk een proxyserver zou laten verlopen, zijn alle video's die het apparaat aankrijgt in een geschikt formaat. De gebruiker hoeft zich dus niet te beperken tot bronnen die zelf adaptatie ondersteunen. Een ander voordeel van een proxyserver is dat, wanneer de proxyserver betrouwbaar is en de verbinding veilig, de gegevens in verband met de gebruiker en het gebruikersapparaat niet naar iedere bron gestuurd hoeven te worden, wat een aantal privacyzorgen opheft. Een nadeel van deze manier van werken is dat de proxyserver één versie van de resource ontvangt. De proxy heeft geen toegang tot de 'masterkopie' of de gegevens waarmee de resource oorspronkelijk is opgesteld. De proxyserver kan dus bijvoorbeeld geen versie van hogere kwaliteit van een film genereren, dan dat de bron bereid is te verzenden.

4.3 Maatstaven

Het doel van de adaptaties en keuzes die op basis van de gebruikersvoorkeuren en apparaateigenschappen worden gedaan, is doorgaans om aan een set van beperkingen (maximumbandbreedte, schermresolutie, ...) te voldoen, en om de ervaring van de gebruiker te optimaliseren. Sommige van deze doelstellingen kunnen uitgedrukt worden als een beperking waarvan bepaald kan worden hoe goed eraan voldaan wordt door bepaalde keuzes of adaptaties. Andere zijn moeilijker om rechtstreeks te kwantificeren. Om deze prestaties en resultaten toch te kunnen meten, worden er metrieken opgesteld. Men kan deze verdelen in objectieve en subjectieve criteria. Een verzameling van objectieve criteria is de zogenaamde *Quality of Service* (QoS). De subjectieve criteria worden meestal aangeduid met *Quality of Experience* (QoE).

4.3.1 Quality of Service

Zoals de naam al aangeeft, is Quality of Service (QoS) een begrip dat bepaald wordt door de kwaliteit van de dienstverlening ('service'). Hoewel het op alles kan worden toegepast dat als een dienstverlening gezien kan worden, wordt het begrip vaak vooral geassocieerd met netwerken (bijvoorbeeld zoals in [95]). In de context van netwerkverbindingen zijn enkele van de criteria die objectief gemeten of vastgesteld kunnen worden de vertraging (*delay*), de

jitter, de minimale gegarandeerde doorvoersnelheid, de verhouding van verloren pakketten tot succesvol verstuurde pakketten (*packet loss rate*), en de ratio van 'omgevallen' (foute) bits tot juiste bits (*bit error rate*).

Om deze criteria zo goed mogelijk te behalen, zijn er een aantal mogelijkheden. Klanten van Internet Service Providers (ISP's) kunnen een contract afsluiten dat waardes stipuleert voor de minimumsnelheid, maximumdelay en maximumjitter van de netwerkverbinding. Deze contracten worden vaak Service Level Agreements (SLA's, [96]) genoemd, aangezien ze het niveau (oftewel de kwaliteit) van de dienstverlening (oftewel de service) vastleggen. Deze contracten worden meestal vergezeld van een hoog prijskaartje, en zijn niet weggelegd voor de gemiddelde thuisgebruiker.

Een andere mogelijkheid om tot op zekere hoogte aan deze criteria te voldoen, is het netwerkverkeer in verschillende klasses in te delen. Men kan in de pakketten aanduiden om welk type verkeer het gaat, en routing-apparatuur gebruiken die routeringsalgoritmes inzetten die verschillende klasses van verkeer op verschillende manieren behandelen [97]. Men zou klasses kunnen specificeren gebaseerd op de eisen die het verkeer erbinnen stelt aan de delay, jitter, packet loss, et cetera. Streaming media, zoals vooraf opgenomen filmpjes, hebben weinig last van delay, en kunnen in beperkte mate jitter opvangen door ontvangen data te bufferen. Packet loss is een iets groter probleem, afhankelijk van de codecs gebruikt voor de audio en video in de stream. Een audiogesprek over een netwerkverbinding heeft grotere vereisten. Zo zorgt een grote delay of jitter op de verbinding al snel voor communicatiemoeilijkheden tussen de betrokken sprekers. Packet loss is ook onwenselijk, maar men kan bijvoorbeeld stellen dat een pakket dat na een bepaald aantal milliseconden nog steeds onderweg is, niet meer afgeleverd hoeft te worden.

4.3.2 Quality of Experience

Naast de objectieve, meetbare criteria van QoS bestaan er ook subjectievere overwegingen, gegroepeerd onder de noemer Quality of Experience (QoE). Men kan het ook wel "'Perceived Quality of Service" noemen, omdat het om de waargenomen (of ervaren) kwaliteit van dienstverlening gaat. De grens tussen QoS en QoE is niet altijd even duidelijk. Zo kan men stellen dat het niveau van irritatie van een gebruiker (bijvoorbeeld bij een slechte telefoonverbinding) een subjectieve eigenschap is, maar wanneer men de stem of gelaatsuitdrukking zou kunnen analyseren om deze irritatie te detecteren, wordt het (zeker gedeeltelijk) een meetbare eigenschap.

Bij het voorbeeld van filmtrailers eerder in dit hoofdstuk, kan een QoS-bewuste browser, indien er een goede netwerkverbinding voorhanden is, voor een filmtrailer van hoge kwaliteit kiezen. Maar dit impliceert niet noodzakelijk een positieve ervaring door de gebruiker. De gebruiker kan bijvoorbeeld een limiet op de hoeveelheid netwerkverkeer naderen, en op dat moment

veel liever de versie van lagere kwaliteit zien als dat betekent dat er in totaal ook minder netwerkverkeer plaatsvindt. Men kan dit counteren door te stellen dat de netwerkverkeerlimiet of de huidige voorkeur van de gebruiker voor kleinere filmtrailers een factor had moeten zijn bij de oorspronkelijke keuze.

Men kan QoS-criteria als ondergeschikt beschouwen ten opzichte van QoE-criteria, omdat het doorgaans de ervaring voor de gebruiker van de uitvoer (beeld, geluid, ...) is die gemaximaliseerd moet worden. In het onwaarschijnlijke geval dat QoS- en QoE-criteria mekaar tegenspreken, mogen QoS-criteria wegens het voorgaande opgeofferd worden om een betere Quality of Experience te verkrijgen.

Hoofdstuk 5

Bestaande systemen

5.1 "Applying CC/PP to User's Environmental Information for Web Service Customization"

De implementatie beschreven in [98] presenteert een experimenteel framework dat gebruik maakt van CC/PP (beschreven in sectie 3.1) en de extensie voor transport van CC/PP over HTTP-headers (kortweg CCPPex, beschreven in sectie 3.1.6) om data omtrent de gebruikersomgeving naar geïnteresseerde web services te transporteren.

De architectuur van de implementatie is opgesplitst in twee delen in de client en twee delen in de server. Het eerste clientgedeelte is een softwarebibliotheek die in staat is verscheidene omgevingskarakteristieken vast te stellen (zoals de fysieke geografische locatie met gehulp van een GPS-ontvanger), en deze als CC/PP-profieldata uitvoert. Het tweede clientgedeelte is een kleine webbrowser (een uitbreiding van een implementatie uit een vorig onderzoek op basis van libwww [99]) die de hiervoor beschreven softwarebibliotheek gebruikt om CC/PP-profielen op te stellen en met behulp van CCPPex met de HTTP-headers mee te sturen.

Aan de kant van de server zorgt een Apache-webserver [100] voor de ontvangst van de HTTP-verzoeken. Het eerste deel van de implementatie aan de serverkant is een Apache-module die eventuele CCPPex-headers onderschept. De CC/PP-profielen worden door deze module uit de CCPPex-headers gehaald en afhankelijk van de configuratie van de module, doorgegeven aan een opgegeven CGI-script op de server. Het CGI-script, wat als het tweede deel van de implementatie aan de serverkant wordt beschouwd, is dan vrij om aanpassingen aan de opgevraagde resource te doen met de verkregen informatie over de gebruikersomgeving. De in de paper gepresenteerde implementatie gebruikt de locatie (verkregen door GPS) om de locatie van de gebruiker aan te duiden op een kaart (een afbeelding) van zijn omgeving.

5.2 Adactus Mobilize

5.2.1 Adactus

Adactus [101] is een Noors softwarebedrijf dat zich specialiseert in *content delivery*, vooral (maar niet enkel) naar mobiele apparaten. Ze richten hun producten op TV-zenders en andere grote soorten mediakanalen. Het bedrijf ontwerpt producten voor de aanlevering, adaptatie en consumptie van multimedia waarbij ze een maximale QoS en QoE proberen te bereiken door rekening te houden met de eindgebruiker en de eigenschappen van het gebruikersapparaat. Ze maken hiervoor gebruik van onder andere de MPEG-21-standaard. De mogelijkheden van hun producten zijn onder andere Video-on-Demand-services (VoD) en live streaming.

5.2.2 Mobilize

Mobilize is een content-delivery-system gebaseerd op onder andere MPEG-21. Het wordt geadverteerd als een manier voor *content providers* om nieuwe (mobiele) klanten te bereiken op nieuwe manieren. De meest technische details van de werking van Mobilize zijn niet gepubliceerd, maar op de website van het bedrijf is een document [102] beschikbaar dat een algemeen overzicht geeft van de verschillende componenten.

Aanlevering

Wanneer de beheerder van het systeem een nieuw stuk of een nieuwe verzameling multimedia wil publiceren, wordt dit in een MPEG-21 Digital Item verpakt. De "Control Server" brengt geregistreerde clients vervolgens op de hoogte van de beschikbaarheid van een nieuwe resource. De gebruikers van deze clients kunnen dan kiezen om het Digital Item op te vragen (af te spelen). Dan wordt ook de Usage Environment Description met details over onder andere het apparaat van de gebruiker naar de Control Server gestuurd. Welke eigenschappen precies verstuurd worden, wordt niet meegedeeld, maar door voorbeelden die gegeven worden staat vast dat de locatie (indien bekend), de audio- en videoformaten die het apparaat kan decoderen, de geluidsweergave (zoals het aantal audiokanalen) en beeldtype en -grootte zeker verzonden worden. Het Digital Item wordt dan, eventueel na conversie door de "Video Transcoder Server", naar de client gestuurd, samen met details over hoe het door de client weergegeven dient te worden. Een specifiek voorbeeld van dit laatste dat gegeven wordt is *branding*. Branding, het promoten van een product door het aan een specifiek merk te koppelen, houdt concreet in dat een reeds bestaand product of een bestaande resource of service wordt aangepast om een bepaald merk te vertonen. Bekende voorbeelden van branding zijn het aanpassen van de firmware van een mobiele telefoon zodat deze de naam, het logo en het kleurenschema van de GSM-operator (prominent) weergeeft, of het aanpassen

van de titelbalktekst van Microsoft Internet Explorer van “Microsoft Internet Explorer” naar “Microsoft Internet Explorer aangeboden door Telenet Internet”. In het huidige voorbeeld kan men een generiek XML-document met presentatievoorkeuren opstellen en er dan door middel van XML-transformaties een specifiek merk en merkspecifieke presentatievoorkeuren in verwerken alvorens het XML-document naar de client te verzenden. Een ander voorbeeld dat gegeven wordt is het aanpassen of selecteren van reclame gebaseerd op de fysieke locatie van de gebruiker.

De software ondersteunt een groot aantal formaten van bestanden, audio en video als invoer in het systeem (om Digital Items van te maken) en als uitvoer van het systeem (voor verzending naar de clients):

- Bestandsformaten: .avi, .dv, .flv, .gxf, .h263, .h264, .m4v, .m4a, .mov, .mpa, .mpg, .mpeg, .mxf, .rm, .ts, .wmv, .wma, .vob;
- Audioformaten: AAC, AMR-NB, AMR-WB, DV Audio, MPEG-1 (mp2 en mp3), Windows Media Audio, Real Audio;
- Videoformaten: AVID DNxHD, Cinepack, DV Video, DVCPRO, Flash Video, H.263, H.264, HuffYUV, M-JPEG, MPEG-1, MPEG-2, MPEG-4, Microsoft MPEG-4, On2 (VP5, VP6), VC-1, Windows Media Video (7, 8, 9), XVID.

Afspelen

Specifieke mobiele apparaten die ondersteund worden zijn onder andere de Apple Ipod, Microsoft Zune en Sony Playstation Portable. Op PC’s wordt onder andere Apple Quicktime en de VLC mediaspeler ondersteund. Daarnaast heeft het bedrijf een clientapplicatie gemaakt voor het J2ME-platform (Java 2 Platform, Micro Edition [103]) dat door een groot aantal moderne mobiele apparaten (inclusief mobiele telefoons) wordt ondersteund. Daarnaast kan de “Mobilize Reporter”-applicatie gebruikt worden om audio- en videoregistratiemogelijkheden van het (mobiele) apparaat te benutten. De hiermee gemaakte opnames kunnen naar de Control Server gestuurd worden en eventueel onmiddellijk ter beschikking gesteld worden als een nieuw Digital Item.

5.3 Network-integrated Multimedia Middleware

Network-integrated Multimedia Middleware (NMM, [104]) is een framework dat zich opwerpt als “multimedia middleware”; een laag tussen de applicatie enerzijds en de client(s), server(s) en het netwerk anderzijds die de applicatie toegang geeft tot een soort platform dat de harde grens tussen client en server probeert te vervagen. Naar eigen zeggen richten ze hun product

vooral op genetwerkte home entertainment systemen. De software is geschreven in C++ en functioneert op Windows, Linux, Mac OS X en de Playstation 3 van Sony.

5.3.1 Communicatie

Het versturen van data doorheen het NMM-systeem gebeurt via een boodschappen-systeem dat twee soorten boodschappen kent.

- "Buffers": bevatten de eigenlijke multimediadata zoals bijvoorbeeld videoframes;
- "Events": ter communicatie van opdrachten, aanvragen, etc. Bijvoorbeeld een aanvraag om het volume te verminderen. Deze events kunnen parameters bevatten van verschillende datatypes.

Er kunnen in de software 'listeners' geregistreerd worden om specifieke events op te vangen en te verwerken.

5.3.2 Architectuur

De verwerking van multimediadata door het systeem gebeurt aan de hand van een keten van componenten ("nodes"). Iedere node heeft één of meerdere in- of uitvoer-"jacks" waaraan andere nodes gekoppeld kunnen worden. Iedere jack aanvaardt enkel een specifiek formaat van data (bijvoorbeeld "audio/mp3") en een aantal parameters. Er bestaan zes verschillende types nodes.

- Source: een bron of producent van data; deze heeft enkel één uitvoerjack;
- Sink: een 'consument' van data; deze heeft enkel één invoerjack;
- Filter: voert een bewerking op data uit zonder het formaat of andere invoerparameters te veranderen; deze heeft één invoer- en één uitvoerjack;
- Converter: hetzelfde als een filter, met het verschil dat deze nodes wel aan het formaat of andere invoerparameters mogen raken;
- Multiplexer: een node met meerdere invoerjacks en één uitvoerjack;
- Demultiplexer: een node met één invoerjack en meerdere uitvoerjacks.

Deze nodes worden in de vorm van plugins meegeleverd. Er zijn al meer dan 60 plugins ontwikkeld, onder andere voor de ondersteuning van XML, RTP, TCP, UDP, het decoderen en coderen van verschillende audio- en videoformaten, het capturen en afspelen van audio

en video, enzovoort. Er is een lijst [105] beschikbaar op de website. Ook een applicatie kan tijdens uitvoeren een lijst met de momenteel beschikbare nodes opvragen uit het "NMM registry". Deze databank houdt van elke plugin bij wat voor type het is en welke in- of uitvoerformaten ondersteund worden.

5.4 NIProxy

De Network Intelligence Proxy (NIProxy, [7]) is een netwerkapplicatie die meer 'intelligentie' in het netwerk plaatst zodat multimediabronnen op een door de client beheerde manier deze client bereiken. Door rekening te houden met zowel de netwerktoestand en de voorkeuren van de clientapplicatie, verhoogt de proxy de QoE voor de gebruiker.

5.4.1 Architectuur

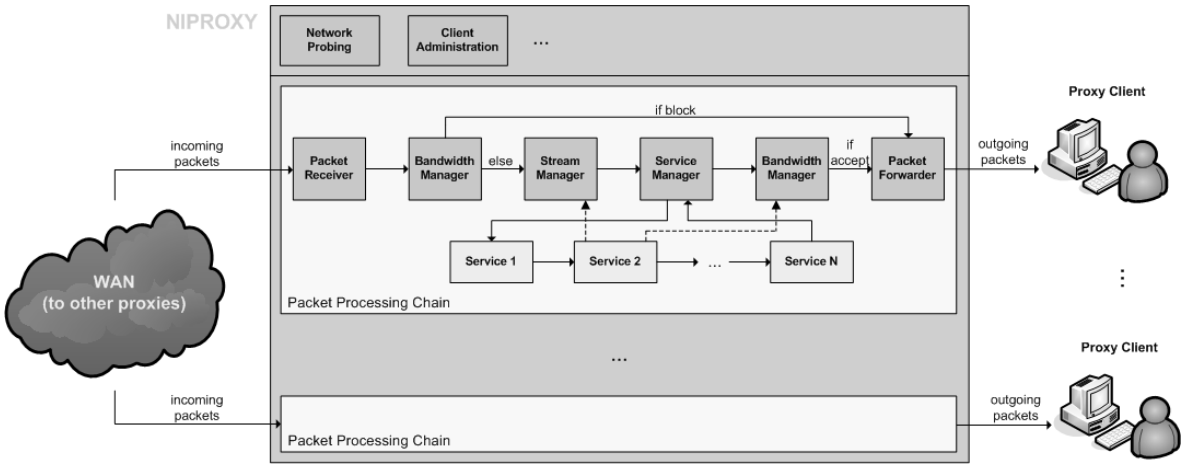
Om gebruik te kunnen maken van de mogelijkheden van de NIProxy, en om metingen van de netwerktoestand tussen proxy en client zo accuraat mogelijk te houden, is het best dat alle (of zoveel mogelijk) netwerkconnecties van de client via de NIProxy verlopen. Voor iedere verbonden client bestaat er een pakketverwerkingsketen in de proxy. Alle pakketten die voor een client aankomen bij de proxyserver, dienen eerst de keten geassocieerd met die client te doorlopen alvorens ze (eventueel) doorgestuurd worden naar de juiste client. De belangrijkste componenten binnen de NIProxy-server, weergegeven op figuur 5.1, zijn:

- Packet Receiver: ontvangt inkomende pakketten en levert ze af aan de pakketverwerkingsketen van de juiste client;
- Bandwidth Manager: gebruikt een boomstructuur opgesteld met informatie aangeleverd door de client (meer hierover in sectie 5.4.2) om de gemeten bandbreedte van die client te verdelen over de pakketten die de selectieprocedure van de keten overleven en naar die client op te sturen;
- Stream Manager: analyseert de inkomende pakketten om de bandbreedtevereisten van individuele network streams te bepalen;
- Service Manager: staat in voor het toepassen van door de client geselecteerde services (meer hierover in sectie 5.4.4) op binnengekomen pakketten;
- Packet Forwarder: verstuurt pakketten die de selectieprocedure van de keten overleefd hebben naar de juiste client.

Wanneer er een voorlopig onbekende verbinding voor (niet van) een client aankomt bij een NIProxy-server, wordt er een omschrijving van de verbinding naar de client gestuurd. In

afwachting van een beslissing van de client omtrent de gewenste afhandeling van de nieuwe verbinding, worden de pakketten ervan geblokkeerd.

Figuur 5.1: Schematische weergave van de NIProxy-architectuur



5.4.2 Detectie

Een deel van de informatie waarover de NIProxy beschikt, wordt door de clientapplicatie aangeleverd. Deze dient de proxy te informeren over network streams waarin ze geïnteresseerd is. Dit wordt bewerkstelligd door de clientapplicatie een *NILayer*-library aan te bieden die toelaat een boom met als bladeren de network streams op te stellen, waarbij intern in de boom drie mogelijke soorten knopen bestaan die aangeven hoe de verdeling van bandbreedte over de streams dient te verlopen:

- Mutex: meerdere streams die mekaar uitsluiten. Bijvoorbeeld een video stream in hoge en lage kwaliteit;
- Priority: kan een rangorde onder streams maken. Knopen die eronder vallen krijgen een prioriteitscijfer toegewezen. Bandbreedte wordt eerst aan de knoop met de hoogste prioriteit gegeven. Eventuele overblijvende bandbreedte gaat naar de knoop met de tweede prioriteit, enz.;
- Weight: verdeelt bandbreedte onder de nodes rechtevenredig met hun geassocieerde weight-value. Bijvoorbeeld om aan te geven dat een video dubbel zo 'belangrijk' is als een andere.

Naast gegevens over network streams, beschikt de NIProxy ook over informatie over de netwerktoestand tussen de proxy en de client. Deze informatie wordt bekomen door regelmatig

een peiling uit te voeren naar de doorvoersnelheid, de vertraging en het pakketverlies van de verbinding.

5.4.3 Interne communicatie

De communicatie tussen de NILayer-library (die op het gebruikersapparaat wordt uitgevoerd) en de NIProxy zelf, gebeurt via een event-systeem. Deze events, NIEvents, hebben een type dat in de boodschap gerepresenteerd wordt door een nummer dat vooraf in de broncode gedefinieerd wordt door **#define**-statements. Verder kunnen de boodschappen bestaan uit parameters van verschillende datatypes, zoals een string of een integer. Dit systeem wordt bijvoorbeeld gebruikt om vanuit de NILayer de hiërarchie van network streams door te geven aan de NIProxy.

5.4.4 Uitbuiting

De NIProxy maakt op twee onderling samenwerkende manieren gebruik van de beschikbare informatie. De eerste manier is via de "Bandwidth Manager". Met behulp van de gemeten beschikbare bandbreedte en de boomstructuur hierboven beschreven, wordt de client downstream bandbreedte verdeeld over de network streams die de clientapplicatie kiest. Niet alleen zorgt dit ervoor dat de clientapplicatie zekerder kan zijn dat bepaalde network streams aankomen, ook wordt de *effectieve* bandbreedte (traffiek waar de clientapplicatie uiteindelijk iets aan heeft) gemaximaliseerd door te proberen congestieverschijnselen te voorkomen. Bijkomend wordt er door de NIProxy bij een verlaging van de beschikbare bandbreedte, indien de stream hiërarchie degelijk is opgesteld, automatisch naar minder traffiekintensieve streams overgeschakeld (door het ontwerp van de Weight-node).

De tweede manier waarop de NIProxy de QoE vergroot is een systeem van "services" (die aan de NIProxy worden toegevoegd in de vorm van plugins) die de inhoud van network streams mogen aanpassen. Zo is er bijvoorbeeld een plugin voorhanden die in staat is om video streams te transcoden zodat ze lagere bitrates hebben. De services kunnen interageren met de Bandwidth Manager: een videostream die oorspronkelijk te groot was, kan na transcoding eventueel wel nog binnen de gealloceerde bandbreedte liggen. Dit kan in de network stream hiërarchie voorgesteld worden door een mutex met als onderliggende streams de originele stream en de transcoded stream.

Hoofdstuk 6

Implementatie

Het implementatie-gedeelte van dit werk is gebaseerd op de Network Intelligence Proxy (NIProxy) beschreven in sectie 5.4. Het bestaande proxy-systeem heeft al kennis van de network streams die de clientapplicatie aanvraagt en de toestand van de netwerkverbinding tussen de proxyserver en het clientapparaat. Het systeem wordt uitgebreid met functionaliteit die de proxyserver informeert over het gebruikersapparaat en de voorkeuren en omgeving van de eindgebruiker. Die informatie wordt beschikbaar gesteld aan alle onderdelen van de NIProxy-server, waardoor deze geïnformeerde beslissingen kunnen maken en hun werking er aan kunnen aanpassen. Om het nut van dit systeem aan te tonen, is er een minimale NIProxy-plugin en NIProxy-client geïmplementeerd.

Om de informatie te transporteren van de client naar de NIProxy-server is er gekozen voor het MPEG-21 Usage Environment Description (UED) representatieformaat. Gegeven de vergelijking in sectie 3.6, waar al aangehaald werd dat de UED zich voornaam toespitst op eigenschappen en voorkeuren met betrekking tot multimedia, en het feit dat het NIProxy-systeem zich uitstekend leent voor het adapteren van multimedia, leek dit een geschikte keuze.

De implementatie valt ruwweg op te delen in vier delen:

- Client: een NIProxy-client waarmee de gebruiker zijn voorkeuren en apparaateigenschappen kenbaar kan maken en die de van de NIProxy ontvangen data weergeeft;
- ClientInfoManager: een toevoeging aan de NIProxy-software om gebruikersvoorkeuren en apparaateigenschappen in MPEG-21 UED-profielen van clients te ontleden, op te slaan en op een bruikbare manier beschikbaar te stellen aan alle andere delen van de NIProxy-server;
- Transformer: een module die gebruik maakt van de NIProxy-pluginarchitectuur om bepaalde network streams die de NIProxy ontvangt te bewerken, rekening houdend met de voorkeuren van de gebruiker verkregen via de ClientInfoManager;

- Transmitter: een kleine NIProxy-client die data verstuurt naar de NIProxy.

6.1 Client

De client is het gedeelte van de implementatie dat de eindgebruiker te zien krijgt. Het bestaat zelf uit drie grote delen:

- Detectie: een library die instaat voor het automatisch detecteren van eigenschappen van het gebruikersapparaat;
- Graphical User Interface: de grafische interface, die de gedetecteerde eigenschappen weergeeft, en de gebruiker toelaat deze en nog vele extra voorkeuren aan te passen.
- Connectie: het gedeelte van het clientprogramma dat instaat voor de connectie en communicatie met de NIProxy, zoals het verzenden van de apparaateigenschappen en voorkeuren, en het ontvangen van data.

6.1.1 Detectie ("libued")

De eigenlijke detectie van een groot deel van de eigenschappen gebeurt in een softwarelibrary, "libued" genaamd. Deze werd voor dit werk geïmplementeerd in C++ en werd succesvol getest onder Microsoft Windows XP en verschillende Linux-distributies met X.org (versies 7.0–7.3). Zoals al aangehaald in hoofdstuk 2 verloopt detectie van eigenschappen op verschillende besturingssystemen op een verschillende manier. Zo is de de library onder Linux afhankelijk van Xlib [106] (een deel van de X-server X.org) omdat deze nodig is voor het achterhalen van informatie omtrent schermen en invoerapparaten. Een voorbeeld van een stuk code dat detectie uitvoert en deels uit deze library komt, kan teruggevonden worden in listing 2.5 in hoofdstuk 2. De eigenschappen die de library detecteert, worden in afwachting van gebruik door het clientprogramma tijdelijk in het werkgeheugen van de library opgeslagen.

Listing 6.1: Prototype van functie voor het opvragen van de schermbreedte

```
1 int GetScreenWidth( bool redetect = false );
```

Listing 6.2: Prototype van functie voor het opvragen van een compleet MPEG-21 UED-profiel

```
1 const char *GetUED( bool redetect = false );
```

In de library werd per gedetecteerde eigenschap een functie gedefinieerd om deze op te vragen. Een voorbeeld hiervan kan gevonden worden in listing 6.1. De `redetect`-parameter kan meegegeven worden om de library te forceren de eigenschappen opnieuw te detecteren. Dit kan nuttig zijn wanneer de gebruiker zelf aanpassingen heeft gemaakt aan het systeem waardoor de eigenschappen veranderd zijn.

Als alternatief voor het per eigenschap opvragen van waardes uit de library, exporteert de library naast de functies voor het opvragen van individuele eigenschappen ook een `GetUED`-functie (weergegeven in listing 6.2) die een tekstrepresentatie van een MPEG-21 UED-profiel in XML-vorm teruggeeft. Enkel indien deze functie aangeroepen wordt, stelt de library een *Document Object Model*-boom (DOM-boom [107, 108]) op die een MPEG-21 UED-document voorstelt. Een DOM-boom laat software toe om dynamisch de inhoud van XML-documenten in te lezen, op te stellen en aan te passen. De gedetecteerde eigenschappen worden op de juiste plaats in de DOM-boom ingevuld. Vervolgens wordt deze DOM-boom, die nu een MPEG-21 UED-document voorstelt dat 'ingevuld' werd met waarden die overeenstemmen met het clientsysteem, omgezet naar een tekstrepresentatie van dat document. Het wordt met andere woorden geserialiseerd naar XML.

Om deze DOM-boom op te stellen en te serialiseren naar tekstvorm maakt de library gebruik van de TinyXML DOM-implementatie [109]. TinyXML is een eenvoudige XML- en DOM-implementatie in C++ die beperkt is in functionaliteit, maar ook zeer licht is en makkelijk gebruik toelaat. Een stuk voorbeeldcode dat TinyXML gebruikt om een MPEG-21 UED-profiel op te stellen met eigenschappen in verband met het scherm, wordt weergegeven in listing 6.3. Bij het gebruik van TinyXML stelt de klasse `TiXmlElement` een XML-element voor in de DOM-boom. Met de methode `SetAttribute` kunnen er XML-attributen aan toegevoegd worden. Via de methode `LinkEndChild` kan er een ander `TiXmlElement` als child-node aan het huidige element worden toegevoegd.

6.1.2 Graphical User Interface

Dit gedeelte van de implementatie, de "graphical user interface" (GUI), geeft de gebruiker de mogelijkheid alle eigenschappen, voorkeuren en het uiteindelijk profiel (in tekstvorm) te bekijken en eventueel aan te passen. Daarnaast geeft het de ontvangen data weer. Het is geschreven in C++ en maakt gebruik van GUI-toolkit Qt [110] (versie 4.4.3). Apparaateigenschappen die door de detectie-library automatisch achterhaald kunnen worden, worden als standaardwaarde van de grafische invoerelementen ingesteld. Om de waardes uit de library op te vragen wordt gebruik gemaakt van de `Get`-functies besproken in sectie 6.1.1. Verder kan de gebruiker per 'onderverdeling' van de eigenschappen en voorkeuren (bijvoorbeeld alles gerelateerd met het scherm) aangeven of alles wat daarbinnen valt al dan niet in het profiel opgenomen dient te worden.

Listing 6.3: Code voor het met TinyXML opstellen van een MPEG-21 UED-profiel dat schermeigenschappen bevat

```
1  TiXmlDocument doc;
2  doc.LinkEndChild( new TiXmlDeclaration( "1.0", "US-ASCII", "yes" ) );
3
4  TiXmlElement *rootElem = new TiXmlElement( "DIA" );
5  rootElem->SetAttribute( "xmlns", "urn:mpeg:mpeg21:2003:01-DIA-NS" );
6  rootElem->SetAttribute( "xmlns:xsi",
       "http://www.w3.org/2001/XMLSchema-instance" );
7
8  TiXmlElement *descElem = new TiXmlElement( "Description" );
9  descElem->SetAttribute( "xsi:type", "UsageEnvironmentType" );
10
11 TiXmlElement *ueElem = new TiXmlElement( "UsageEnvironment" );
12 ueElem->SetAttribute( "xsi:type", "TerminalCapabilitiesType" );
13
14 TiXmlElement *tcElem = new TiXmlElement( "TerminalCapabilities" );
15 tcElem->SetAttribute( "xsi:type", "InputOutputCapabilitiesType" );
16
17 TiXmlElement *displayElem = new TiXmlElement( "Display" );
18 tcElem->SetAttribute( "bitsPerPixel", info->bitsPerPixel );
19
20 TiXmlElement *resolutionElem = new TiXmlElement( "Resolution" );
21 resolutionElem->SetAttribute( "horizontal", info->screenWidth );
22 resolutionElem->SetAttribute( "vertical", info->screenHeight );
23
24 displayElem->LinkEndChild( resolutionElem );
25 tcElem->LinkEndChild( displayElem );
26 ueElem->LinkEndChild( tcElem );
27 descElem->LinkEndChild( ueElem );
28 rootElem->LinkEndChild( descElem );
29 doc.LinkEndChild( rootElem );
30
31 TiXmlPrinter printer;
32
33 // this makes the output compact: no indentation, no line breaks, ...
34 printer.SetStreamPrinting();
35 doc.Accept( &printer );
36
37 ued = printer.Str();
```

Dit gedeelte van de ontvanger maakt geen gebruik van de `GetUED`-functie voorzien in de detectielibrary. Dit heeft verschillende redenen. Indien het volledig profiel uit de library opgevraagd zou worden, moet dit eerst door de library opgesteld worden. Daarna zou dit in de GUI opnieuw ontleed moeten worden om de waarden te kunnen weergeven in de GUI-elementen. Daarna zou het hele profiel opnieuw opgesteld moeten worden. Als alternatief voor de `GetUED`-functie wordt er gebruik gemaakt van een basisvoorziening voor XML in Qt, de QXmlStreamWriter [111]. Hoewel deze beperkter is dan de TinyXML-library, is de functionaliteit ervan voldoende voor deze toepassing. In de detectielibrary werd voor TinyXML geopteerd omdat deze veel kleiner is dan de Qt toolkit.

De QXmlStreamWriter biedt een eenvoudige manier om XML-documenten op te stellen op een manier die iets gestructureerder is dan gewone tekstbewerking, maar geen creatie en koppeling van verschillende C++-klassen vereist. In tegenstelling tot de TinyXML-library maakt de QXmlStreamWriter geen gebruik van een interne datastructuur. De QXmlStreamWriter bevat een verzameling methodes voor het opbouwen van een string die het geserialiseerde XML-document voorstelt. Deze methodes doen hun werk bijna allemaal door enkel aan het eind van de string toevoegingen te doen. Dit betekent dat, opnieuw in tegenstelling tot bij de TinyXML-library, elementen en attributen altijd in de volgorde waarin ze in het resultaat moeten verschijnen, moeten worden toegevoegd. Listing 6.4 geeft een stuk voorbeeldcode weer. Bij het aanmaken van de `QXmlStreamWriter` wordt deze gekoppeld aan een `QString` welke het resultaat zal bevatten. XML namespaces kunnen toegevoegd worden met de `writeNamespace`-methode. Deze worden aan het eerstvolgende element toegevoegd. Een XML-element kan worden toegevoegd met de `writeStartElement`-methode (met als parameters de namespace en de naam van het element). De `writeAttribute`-methode (met als parameters de namespace, de naam en de waarde van het attribuut) voegt XML-attributen toe aan het laatst toegevoegde XML-element. Aan de hand van de namespace-parameter van de voorgaande twee methodes kiest de QXmlStreamWriter automatisch het juiste namespace prefix, aangegeven door de voorgaande `writeNamespace`-aanroepen, voor de elementen en attributen. Indien de namespace nog niet met een `writeNamespace`-aanroep bekend gemaakt werd, wordt deze in het huidig element gedeclareerd met een door Qt gegenereerde naam (bijvoorbeeld “n1”). De methode `writeEmptyElement` produceert een XML-element zonder inhoud of subelementen (van de vorm <ElementNaam attribuut=”waarde”/>). De methode `writeEndDocument` betekent het einde van het document en sluit meteen ook alle ‘open’ elementen af die ervoor nog niet met de `writeEndElement`-methode zijn gesloten. Een XML-document geproduceerd met behulp van de code in listing 6.4 staat in listing 6.5.

Nadat het UED-profiel gegenereerd is, kan de gebruiker het nog bekijken en aanpassen.

De informatie voor het opzetten van de connectie met de NIProxy, namelijk het IP-adres en de poort, moet ook in de GUI ingevoerd worden. Eens een connectie is opgezet, kan de gebruiker kiezen om het profiel door te sturen. Erna kan de network stream die door

Listing 6.4: Code voor het met QXmlStreamWriter opstellen van een MPEG-21 UED-profiel dat schermeigenschappen bevat

```
1  QString xml;
2  QXmlStreamWriter xmlWriter( &xml );
3
4  xmlWriter.writeNamespace( "urn:mpeg:mpeg21:2003:01-DIA-NS", "dia" );
5  xmlWriter.writeNamespace( "http://www.w3.org/2001/XMLSchema-instance",
       "xsi" );
6
7  // writes the <?xml ... declaration
8  xmlWriter.writeStartDocument();
9
10 // this is the first element after the writeNamespace() calls; it gets
       all the namespace declarations
11 xmlWriter.writeStartElement( "urn:mpeg:mpeg21:2003:01-DIA-NS", "DIA" );
12 xmlWriter.writeStartElement( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "Description" );
13 xmlWriter.writeAttribute( "http://www.w3.org/2001/XMLSchema-instance",
       "type", "UsageEnvironmentType" );
14
15 xmlWriter.writeStartElement( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "UsageEnvironment" );
16 xmlWriter.writeAttribute( "http://www.w3.org/2001/XMLSchema-instance",
       "type", "TerminalCapabilitiesType" );
17
18 xmlWriter.writeStartElement( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "TerminalCapabilities" );
19 xmlWriter.writeAttribute( "http://www.w3.org/2001/XMLSchema-instance",
       "type", "InputOutputCapabilitiesType" );
20
21 xmlWriter.writeStartElement( "urn:mpeg:mpeg21:2003:01-DIA-NS", "Display"
       );
22 xmlWriter.writeAttribute( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "bitsPerPixel", QString::number(
       spinbox_Display_bitsPerPixel->value() ) );
23 xmlWriter.writeAttribute( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "colorCapable", ( check_Display_colorCapable->isChecked() ? "true" :
       "false" ) );
24 xmlWriter.writeAttribute( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "refreshRate", QString::number( spinbox_Display_refreshRate->value()
       ) );
25
26 xmlWriter.writeEmptyElement( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "Resolution" );
27 xmlWriter.writeAttribute( "urn:mpeg:mpeg21:2003:01-DIA-NS",
       "horizontal", QString::number( spinbox_Resolution_horizontal->value()
       ) );
28 xmlWriter.writeAttribute( "urn:mpeg:mpeg21:2003:01-DIA-NS", "vertical",
       QString::number( spinbox_Resolution_vertical->value() ) );
29
30 xmlWriter.writeEndDocument();
```

Listing 6.5: MPEG-21 UED geproduceerd door de code in listing 6.4

```
1  <?xml version="1.0"?>
2  <dia:DIA xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3      <dia:Description xsi:type="UsageEnvironmentType">
4          <dia:UsageEnvironment xsi:type="TerminalCapabilitiesType">
5              <dia:TerminalCapabilities
                   xsi:type="InputOutputCapabilitiesType">
6                  <dia:Display dia:bitsPerPixel="32"
                       dia:colorCapable="true" dia:refreshRate="60">
7                      <dia:Resolution dia:horizontal="1920"
                           dia:vertical="1200"/>
8                  </dia:Display>
9              </dia:TerminalCapabilities>
10         </dia:UsageEnvironment>
11     </dia:Description>
12 </dia:DIA>
```

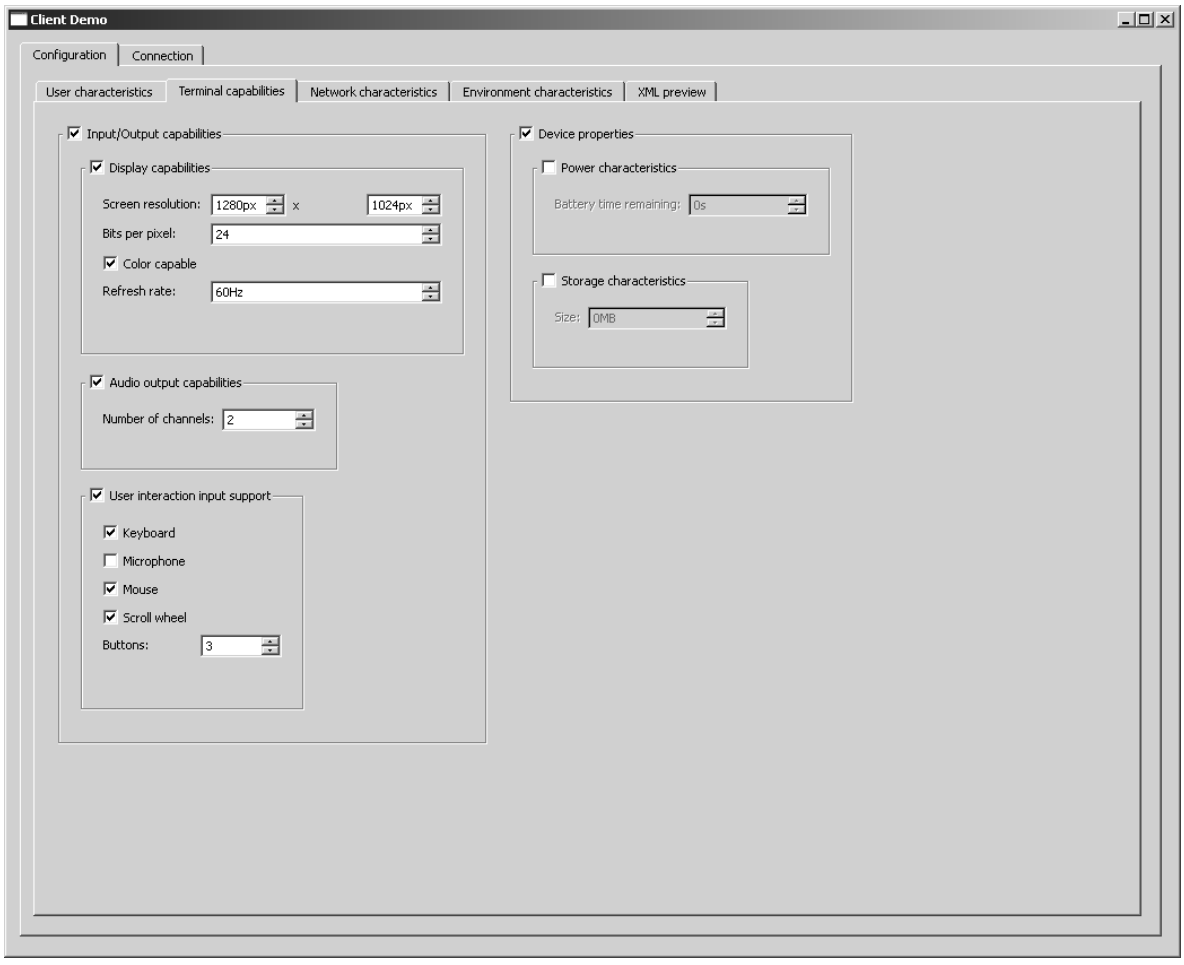
de Transmitter (zie sectie 6.4) ingevuld wordt, aangevraagd worden. De data die op deze stream ontvangen wordt, wordt dan eronder weergegeven. In figuur 6.1 wordt een tabblad van de GUI weergegeven waarop een subset van de "Terminal capabilities" van MPEG-21 UED opgegeven kan worden. Figuur 6.2 geeft het tabblad weer waarop de connectie ingesteld kan worden en de ontvangen data weergegeven wordt.

6.1.3 Connectie

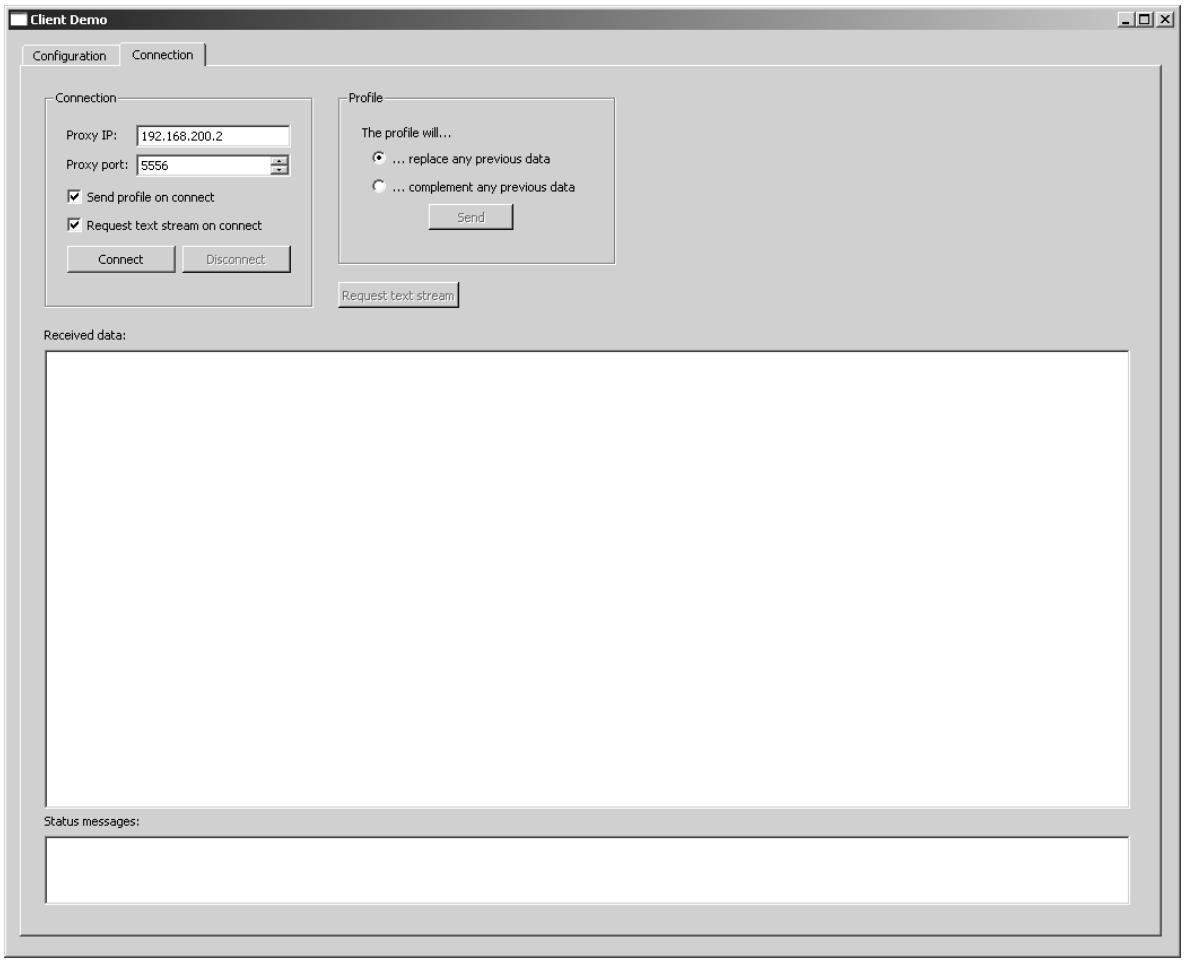
Om de connectie en communicatie met de NIProxy te verzorgen, maakt deze implementatie gebruik van de Network Intelligence library ("NILayer") en de Network Abstraction library ("NAB"). De voornaamste taken van dit gedeelte zijn het verzenden van het UED-profiel naar de proxyserver, het aanvragen van de network stream en het ontvangen van de data in de aangevraagde network stream. Om het UED-profiel te versturen, werd er eerst een nieuw type `NIEvent` gedefinieerd, namelijk `NI_EVENT_CLIENT_INFO_DATA`. In dit event wordt het profiel in tekstvorm gestoken. Vervolgens wordt dit via het event-systeem naar de NIProxy gestuurd.

Na het aanvragen van de network stream, controleert deze module continu op de aanwezigheid van nieuwe pakketten op de aangevraagde network stream. Indien er nieuwe pakketten aanwezig zijn, worden deze gelezen, en de 'payload' ervan wordt zonder verdere bewerkingen weergegeven in het tekstveld van de gebruikersinterface.

Figuur 6.1: Tabblad van de NIProxy-client GUI waar “terminal capabilities” geconfigureerd kunnen worden



Figuur 6.2: Tabblad van de NIProxy-client GUI waar connectie-informatie opgegeven kan worden en de ontvangen data weergegeven wordt



6.2 ClientInfoManager

Het ontleden, opslaan en beschikbaar stellen van de profielinformatie wordt binnen de NIProxy afgehandeld door de singleton `ClientInfoManager`-klasse. Deze krijgt het UED-profiel via de methode weergegeven in listing 6.6 in stringvorm aangeboden, samen met de `clientID` die de client identificeert waarvan het profiel afkomstig is. De parameter `isReplacement` geeft aan of het nieuwe profiel een complement of een vervanging is voor eventueel al bestaande profieldata.

Listing 6.6: Prototype van methode die profieldata in de `ClientInfoManager` invoert

```
1 void addClientInfo( const unsigned long clientID, const bool
    isReplacement, const char *pClientInfo );
```

6.2.1 Ontleding

Wanneer de `ClientInfoManager` een profiel van een client ontvangt, is dit een MPEG-21 UED-profiel in XML-vorm. Om waardes van eigenschappen uit dat profiel te halen, wordt gebruik gemaakt van XPath-expressies [70]. De XPath-implementatie die hiervoor gebruikt wordt is XQilla [112], een library die XPath implementeert, gebruik makend van de Apache Xerces-C(++) XML-parser [69] voor de interne XML-representatie. Bij het uitvoeren van de XPath-expressies wordt door XQilla geverifieerd dat de ontvangen data goed-gevormde XML is. Het resultaat van deze XPath-expressies is een waarde in tekstvorm. Listing 6.7 geeft een stuk voorbeeldcode weer dat de schermbreedte uit een MPEG-21 UED-profiel haalt.

De XPath-expressies, samen met een naam waaronder het resultaat opgeslagen moet worden, moeten door modules die gebruik maken van de `ClientInfoManager` bij de klasse geregistreerd worden door middel van de methode weergegeven in listing 6.8. Wanneer er al een XPath-expressie geregistreerd is die zijn resultaten opslaat onder de gegeven naam, geeft deze functie **false** terug. Een andere aanpak zou zijn om altijd zoveel mogelijk eigenschappen uit het profiel te halen en op te slaan. Een groot voordeel van het verplicht registreren van expressies voor iedere eigenschap die gebruikt zal worden is dat zowel het verwerken als het opslaan van het profiel beperkt blijft tot hetgene dat echt gebruikt zal worden.

6.2.2 Opslag

De volgende stap is de resultaten van de XPath-expressies, die profieleigenschappen voorstellen, onthouden. Er zijn veel mogelijke manieren om dit te doen. Eén punt waar rekening

Listing 6.7: Code voor opzoeken van de schermbreedte in een MPEG-21 UED-profiel met behulp van XQilla

```
1  XQilla xqilla;
2
3  XQQuery *query = xqilla.parse( X(
       "//*[local-name()='Resolution']/@*[local-name()='horizontal']" ) );
4  DynamicContext *dynContext = query->createDynamicContext();
5  DocumentCache *docCache = ( DocumentCache *
       )dynContext->getDocumentCache();
6  MemBufInputSource inStream( ( const XMLByte * )pClientInfo, strlen(
       pClientInfo ), "clientInfo", false );
7  Node::Ptr nodePtr;
8  try {
9    nodePtr = docCache->parseDocument( inStream, dynContext );
10 }
11 catch (...)
12 {
13   LOGERR("Error while parsing document!");
14   return;
15 }
16
17 if( nodePtr && nodePtr->isNode())
18 {
19   dynContext->setContextItem( nodePtr );
20   dynContext->setContextPosition( 1 );
21 }
22
23 Result result = query->execute( dynContext );
```

Listing 6.8: Prototype van functie voor het registreren van een XPath-expressie bij de `ClientInfoManager`

```
1  bool registerXPathExpression( const char *propertyName, const char
       *expression );
```

mee gehouden dient te worden, is dat de grootte van de dataset bij het schrijven van de implementatie niet bekend is. De grootte van de dataset is rechtstreeks afhankelijk van hoeveel XPath-expressies op ieder profiel uitgevoerd worden en het aantal clients van de NIProxy. Het maximum aantal clients dat een NIProxy ondersteunt, wordt bepaald door verschillende factoren, zoals de beschikbare bandbreedte, de hardware van het systeem, en de plugins die gebruikt worden. Zo zal een streaming video transcoding-plugin veel meer resources vereisen dan een plugin die x- en y-coördinaten aanpast. Een ander punt is dat het op voorhand niet geweten is welke plugins de NIProxy zal laden tijdens gebruik, en hoe intensief die plugins gebruik zullen maken van de dataset die de `ClientInfoManager` beschikbaar stelt. Ze kunnen aan de ene kant geen of aan de andere kant tientallen XPath-expressies registreren die op ieder profiel uitgevoerd moeten worden en een eigenschap opleveren die opvraagbaar dient te zijn.

Indien het om slechts een handvol clients zou gaan, kunnen de eigenschappen volledig in het geheugen opgeslagen worden. Bij een groot aantal clients kan deze aanpak echter voor geheugenschaarste van het NIProxy-systeem zorgen. Een andere aanpak is om de verkregen profielen als XML-bestanden (in tekstvorm, zoals ze ontvangen worden) op te slaan op vaste schijf. Dit kan echter voor onacceptabel grote vertragingen zorgen bij het opnieuw inladen van een profiel. Deze implementatie lost het probleem op door de resultaten van de XPath-expressies op te slaan in een Database Management System (DBMS). Met deze oplossing zijn de profielen meestal snel beschikbaar (gecached), maar is het risico op geheugenschaarste beperkt doordat het DBMS de vaste schijf als secundaire opslag gebruikt indien de dataset te groot wordt voor in het werkgeheugen te houden. Een bijkomend voordeel is dat Structured Query Language (SQL) gebruikt kan worden voor de manipulatie van de dataset en om makkelijk specifieke eigenschappen op te zoeken. Het gebruik van een degelijk DBMS verzekert dat de opzoekoperaties op een efficiënte manier worden uitgevoerd door middel van de interne algoritmes in het DBMS. De opslag in een DBMS is ook persistent; de data gaat niet verloren indien de NIProxy vastloopt of herstart wordt. Het is eventueel zelfs mogelijk om van een externe server gebruik te maken die het DBMS draait. Zo zouden meerdere NIProxy-instanties dezelfde dataset kunnen gebruiken. Bovendien kan het ontwerp van de databasetabellen aangepast worden zonder dat alle code die van de database gebruik maakt, aangepast moet worden. Zo kan er zonder veel moeite een extra kolom aan iedere tabel toegevoegd worden voor onvoorziene extra mogelijkheden of data.

Het Database Management System dat in deze implementatie ingezet wordt is SQLite [113]. De keuze viel op SQLite omdat dit een DBMS is dat niet bijzonder uitgebreid is, maar toch alle functionaliteit bevat die vereist is van het opslagsysteem in deze implementatie.

Alle eigenschappen van alle clients van een NIProxy-server worden opgeslagen in één SQLite database. Er wordt voor iedere proxyclient een aparte tabel in die database aangemaakt. De structuur van de tabel is een eenvoudig “name, value”-schema van twee kolommen:

`propertyName` de naam van de eigenschap;

`propertyValue` de waarde van de eigenschap `propertyName` voor de client in kwestie.

Wanneer een profiel van een client wordt doorgegeven, wordt de databasetabel voor deze client met de SQL-expressie weergegeven in listing 6.9 aangemaakt, indien deze tabel nog niet bestaat. De resultaten van de uitgevoerde XPath-expressies worden een voor een in de tabel toegevoegd met behulp van de SQL-expressie weergegeven in listing 6.10.

De genummerde vraagtekens in listing 6.10 en listing 6.12 (die later in de tekst nog volgt) staan voor waardes die later in de code pas daadwerkelijk worden ingevuld. De SQL-expressie wordt eerst voorbereid door middel van de functie sqlite3_prepare_v2 (). De eigenlijke waardes die op de plaatsen met vraagtekens komen maken op dat moment nog niets uit. Deze worden later ingevuld met behulp van de functies sqlite3_bind_int () en sqlite3_bind_text ().

Listing 6.9: SQL-expressie die de databasetabel aanmaakt voor client met clientID 52

```
1 CREATE TABLE IF NOT EXISTS client52 ( propertyName TEXT, propertyValue
    TEXT )
```

Listing 6.10: SQL-expressie die eigenschappen invoegt van client met clientID 52

```
1 INSERT INTO client52 ( propertyName, propertyValue ) VALUES( ?1, ?2 )
```

6.2.3 Toegang

Wegens de mogelijkheid van de NIProxy-architectuur om op verzoek plugins te laden die vervolgens allerlei diverse services kunnen aanbieden aan de proxy client om op de network streams toe te passen, kan de `ClientInfoManager` niet weten welke plugins geladen zullen worden. Daardoor kan de klasse er niet voor instaan om alle services te configureren met net die eigenschappen waar ze in geïnteresseerd zijn. De idee is dan ook om de `ClientInfoManager` te laten fungeren als een interface naar de eigenschappen van het gebruikersapparaat en de voorkeuren van de gebruiker, die door de services zelf aangesproken dient te worden. Een alternatieve aanpak die niet werd geïmplementeerd zou zijn om met een systeem te werken waarbij een service bij het laden ervan aan de `ClientInfoManager` aangeeft in welke eigenschappen hij geïnteresseerd is, en van de waarde ervan op de hoogte wordt gesteld als en wanneer deze eigenschap door de client wordt gerapporteerd. Het signal/slot mechanisme dat in Qt aanwezig is, zou hiervoor ingezet kunnen worden door in de `ClientInfoManager` een signal te definiëren dat uitgezonden (ge-"emit") wordt wanneer

nieuwe profieldata beschikbaar is. Alle geïnteresseerde stukken code moeten dan een slot definiëren en implementeren dat met het signaal van de `ClientInfoManager` geconnecteerd wordt. De eigenschappen zouden dan nog steeds opgeslagen moeten worden om services die pas later geladen worden ook te kunnen informeren.

Wanneer een ander deel van de NIProxy-server (zoals een service die op een network stream opereert) een eigenschap wil opvragen, kan dit via de functie `getPropertyFromDB()` waarvan het prototype gegeven wordt in listing 6.11. De naam van iedere eigenschap is vastgelegd bij het registreren van de XPath-expressie zoals uitgelegd in sectie 6.2.1. De kern van de implementatie van deze functie zit in het uitvoeren van de SQL-expressie, gegeven in listing 6.12, om de gevraagde eigenschap uit de database te halen. Dit kan natuurlijk alleen een waarde teruggeven als de client in kwestie een profiel heeft doorgestuurd, en de eigenschap in kwestie onderdeel uitmaakte van dat profiel. Aangezien sommige eigenschappen meerdere keren mogen voorkomen (zoals ondersteunde schermresoluties), wordt ermee rekening gehouden dat de SQL-expressie meerdere resultaten kan hebben. Om die reden geeft de functie in kwestie een vector terug. Via deze interface zijn de eigenschappen en voorkeuren uit het UED-profiel via de `ClientInfoManager` beschikbaar voor alle andere delen van de NIProxy.

Listing 6.11: Prototype van functie voor het opvragen van clienteigenschappen

```
1 vector< string > getPropertyFromDB( const unsigned long clientID, const
    char *propertyName );
```

Listing 6.12: SQL-expressie die een eigenschap ophaalt uit de databasetabel van client met clientID 52

```
1 SELECT propertyValue FROM client52 WHERE propertyName = ?
```

6.3 Transformer

Om aan te tonen dat het geheel van deze implementatie functioneert, werd er een minimale NIProxy-plugin geïmplementeerd. Deze associeert zich met een network stream waarover tekst verstuurd wordt. Wanneer de gebruiker zijn voorkeur voor content met een “G”- of “PG”-rating aangeeft door middel van een profiel, wordt de verstuurde tekst doorzocht op een aanpasbare lijst woorden die gemaskeerd moeten worden. Deze situatie zou zich bijvoorbeeld kunnen voordoen wanneer de ondertiteling van een video in een aparte network stream wordt doorgestuurd en deze volgens de client maximaal een bepaalde “Parental Guidance”-rating

mogen hebben.

Bij het instantiëren van de plugin wordt de XPath-expressie //*[local−name()='ParentalRating']/text() met benaming "parentalrating" aan de `ClientInfoManager`-klasse doorgegeven (zoals weergegeven in listing 6.13). Dit zorgt ervoor dat die XPath-expressie wordt uitgevoerd op alle profieldata die de proxyserver ontvangt, en het resultaat ervan onder de juiste benaming opgeslagen wordt. Ook wordt er bij het instantiëren van de plugin een lijst van strings, zoals woorden of stukken van zinnen, uit een bestand ingelezen die door het filterproces gemaskeerd moeten worden. Door de associatie met de network stream waarover tekst verstuurd wordt, wordt de `processPacket`-functie van de plugin door de NIProxy-server aangeroepen voor ieder pakket op die network stream. De plugin controleert met behulp van de `GetPropertyFromDB`-functie (weergegeven in listing 6.14) wat het resultaat van de eerder geregistreerde XPath-expressie is voor de client waar het pakket in kwestie voor bestemd is.

Listing 6.13: Registratie van een XPath-expressie onder de naam "parentalrating" bij de `ClientInfoManager`

```
1 pClientInfoManager->registerXPathExpression( "parentalrating",
      "//*[local-name()='ParentalRating']/text()" );
```

Listing 6.14: Opvragen van de eigenschap met naam "parentalrating" voor een bepaalde client uit de `ClientInfoManager`

```
1 vector< string > result = m_pClientInfoManager->getPropertyFromDB(
      m_pClient->getClientID(), "parentalrating" );
```

Indien het resultaat van de functie "G" of "PG" is ("General Audiences" en "Parental Guidance suggested", respectievelijk), wordt de data ("payload") in het pakket op een hoofdletterongevoelige manier doorzocht op de eerder ingeladen strings. Wanneer er zo'n string voorkomt, wordt deze vervangen door asterisken. De bewerkte inhoud wordt daarna terug in het pakket gestoken, waarna de NIProxy dit kan verdersturen naar de client.

6.4 Transmitter

Dit deel van de implementatie bestaat uit een kleine applicatie die een connectie opzet met de NIProxy, een network stream aanvraagt, en daarop data uitzendt. Omdat het doel ervan niet meer is dan voor testdoeleinden data te voorzien voor de NIProxy-plugin en de receiver van deze implementatie, komt de data simpelweg uit een tekstbestand, en wordt de inhoud ervan eindeloos herhaald. Hoe de connectie met de NIProxy in zijn werk gaat, werd al besproken

in sectie 5.4 over de NIProxy en ook in de bespreking van de receiver van deze implementatie in sectie 6.1.3.

6.5 Resultaten

De belangrijkste resultaten van deze implementatie zijn natuurlijk het opstellen, ontvangen, ontleden en beschikbaar stellen aan de proxy van de voorkeuren van de gebruiker en de eigenschappen van het gebruikersapparaat en de omgeving. Een resultaat van het implementeren van de NIProxy-plugin, transmitter en receiver is dat er nu een systeem bestaat voor het verzenden, ontvangen en onderweg aanpassen van ondertitelingen en transcripties. Hoewel het voldoen aan een vaag omschreven "parental guidance"-norm moeilijk kwantificeerbaar is, wordt in figuur 6.5 voor de volledigheid een stuk tekst weergegeven, zonder en met bewerking door de NIProxy-plugin.

Recall that earlier generations faced down fascism and communism not just with missiles and tanks, but with sturdy alliances and enduring convictions. They understood that our power alone cannot protect us, nor does it entitle us to do as we please. Instead, they knew that our power grows through its prudent use; our security emanates from the justness of our cause, the force of our example, the tempering qualities of humility and restraint.

Figuur 6.3: Excerpt uit de inauguratiespeech van Barack Obama

Recall that earlier generations faced down fascism and ********* not just with missiles and tanks, but with sturdy alliances and enduring convictions. They understood that our ***** alone cannot protect us, nor does it entitle us to do as we please. Instead, they knew that our ***** grows through its prudent use; our security emanates from the justness of our cause, the force of our example, the tempering qualities of humility and restraint.

Figuur 6.4: Bewerkt excerpt uit de inauguratiespeech van Barack Obama

Hoofdstuk 7

Conclusie

Ondanks dat er geen overduidelijke conclusie getrokken kan worden, zijn er toch enkele bevindingen omtrent het "detecteren, representeren en uitbuiten van de eigenschappen van het apparaat van de eindgebruiker" die het vermelden waard zijn. Vooreerst kan er gesteld worden dat er nog geen standaard noch de facto standaard onderdelen beschikbaar zijn om een deel van of de volledige taak uit te voeren. Een grote hindernis bij het detecteren van apparaateigenschappen is de grote variëteit van de apparaten en vooral de besturingssystemen die erop draaien. Deze hindernis is misschien een logisch gevolg van de noodzaak tot detectie; als er geen variëteit in deze dingen bestond, was detectie niet of in veel mindere mate nodig. Er bestaan in ieder geval nog geen bekende en uitgebreide frameworks of andere oplossingen waar op gebouwd kan worden.

Op het vlak van representatie bestaan er wel al een heel aantal keuzemogelijkheden. Deze hebben elk hun specifieke voor- en nadelen, zonder dat er één onbetwist als winnaar aan te duiden valt. Voorlopig hangt het vooral van het toepassingsgebied af wat de meest geschikte oplossing is. Op het vlak van multimedia is de MPEG-21 Usage Environment Description één van de aanraders, terwijl CC/PP zijn sterktes laat zien bij mobiele toegang tot het web.

Ongeacht hoe de eigenschappen gedetecteerd en voorgesteld worden, éénmaal ze er zijn, bestaan er vele mogelijkheden om er gebruik van te maken. Hoewel de MPEG-21 Digital Item Adaptation een standaard is die zich in dit deelgebied situeert, is het misschien niet nodig om hier een algemene oplossing te verwachten, aangezien iedere uitbuiting specifiek voor de situatie, de omringende software en de data in kwestie zal zijn.

Een belangrijk punt is dat de focus niet enkel op de systeemspecificaties van het gebruikersapparaat moet liggen, maar de voorkeuren van de eindgebruiker zelf ook een belangrijke rol horen te spelen, aangezien deze de uiteindelijke consument en het uiteindelijke doel is van het gehele systeem.

Bibliography

[1] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jonathan B. Postel, Lawrence G. Roberts, and Stephen S. Wolff. A brief history of the Internet, 1999. http://www.isoc.org/internet/history/brief.shtml.

[2] Internet Systems Consortium. Domain survey: Number of Internet hosts. http://www.isc.org/ops/ds/host-count-history.php.

[3] World Internet usage statistics news and world population stats. http://internetworldstats.com/stats.htm.

[4] CIA. The world factbook — rank order — Internet users. https://www.cia.gov/library/publications/the-world-factbook/rankorder/2153rank.html.

[5] Cisco. Cisco visual networking index — forecast and methodology, 2007–2012. White Paper, June 2008. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481374.pdf.

[6] Anukool Lakhina, John W. Byers, Mark Crovella, and Ibrahim Matta. On the geographic location of Internet resources. In *Internet Measurement Workshop '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 249–250, New York, NY, USA, 2002. ACM. http://www.acm.org/sigs/sigcomm/imw2002/imw2002-papers/184.pdf.

[7] Maarten Wijnants and Wim Lamotte. The NIProxy: a Flexible Proxy Server Supporting Client Bandwidth Management and Multimedia Service Provision. In *Proceedings of the 8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*, Helsinki, Finland, June 2007.

[8] Sony Ericsson S500i UAProf-profile. http://wap.sonyericsson.com/UAprof/S500iR201.xml.

[9] Randall Hyde. *The Art of Assembly Language*. No Starch Press, San Francisco, CA, USA, September 2003. http://webster.cs.ucr.edu/AoA/Windows/HTML/TheMMXInstructionSet.html.

[10] *Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture*. Intel Corporation, 1997. http://download.intel.com/design/PentiumII/manuals/24319002.PDF.

[11] Intel Processor Identification and the CPUID Instruction. Application note, Intel Corporation, December 2007. http://www.intel.com/Assets/PDF/appnote/241618.pdf.

[12] FFmpeg team. Libavcodec example of CPU-detection. http://svn.ffmpeg.org/ffmpeg/trunk/libavcodec/x86/cpuid.c?view=markup.

[13] CPUID EURL. CPU-Z versiegeschiedenis. http://www.cpuid.com/cpuz.php#history.

[14] TechPowerUp. GPU-Z Video card GPU Information Utility. http://www.techpowerup.com/gpuz/.

[15] Intel corporation. Intel corporation homepage. http://intel.com.

[16] Nvidia corporation. Nvidia corporation homepage. http://nvidia.com.

[17] Inc. Advanced Micro Devices. Advanced micro devices homepage. http://www.amd.com/.

[18] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons, Inc., New York, NY, USA, sixth edition, April 2002.

[19] FreeDesktop.org — HAL overzichtspagina. http://freedesktop.org/wiki/Software/hal.

[20] Havoc Pennington. Making hardware just work, July 2003. http://ometer.com/hardware.html.

[21] Freedesktop.org website. http://www.freedesktop.org/wiki/.

[22] David Zeuthen. HAL 0.5.10 Specification. http://people.freedesktop.org/~david/hal-spec/hal-spec.html.

[23] FreeDesktop.org — D-Bus overzichtspagina. http://freedesktop.org/wiki/Software/dbus.

[24] X.org Foundation. X.org homepage. http://www.x.org/wiki/.

[25] World Wide Web Consortium website. http://www.w3.org/.

[26] About W3C: History. http://www.w3.org/Consortium/history.

[27] About W3C: Organization. http://www.w3.org/Consortium/org.

[28] About the World Wide Web Consortium. http://www.w3.org/Consortium/.

[29] W3C Working Group map. http://www.w3.org/2003/02/W3COrg.png.

[30] Device Independence Working Group: Principles: Goals. http://www.w3.org/TR/di-princ/#section-Goals.

[31] Ubiquitous Web Applications Working Group Charter. http://www.w3.org/2006/10/uwa-charter.html.

[32] Ubiquitous Web Applications Activity. http://www.w3.org/2007/uwa/.

[33] Chris Woodrow, Johan Hjelm, Hidetaka Ohto, Luu Tran, Franklin Reynolds, Graham Klyne, and Mark H. Butler. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C recommendation, W3C, January 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[35] Cédric Kiss. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0. a WD in last call, W3C, April 2007. http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430.

[36] Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0: Changes since CC/PP 1.0. http://www.w3.org/TR/CCPP-struct-vocab2/#CCPP2Changes.

[37] Dave Beckett. RDF/XML Syntax Specification (Revised). W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[38] Eric Miller and Frank Manola. RDF primer. W3C recommendation, W3C, February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[39] Jean Paoli, C. M. Sperberg-McQueen, Tim Bray, François Yergeau, and Eve Maler. Extensible markup language (XML) 1.0 (fourth edition). W3C recommendation, W3C, August 2006. http://www.w3.org/TR/2006/REC-xml-20060816.

[40] J. Kunze and T. Baker. The Dublin Core Metadata Element Set. RFC 5013 (Informational), August 2007.

[41] Chris Woodrow, Johan Hjelm, Hidetaka Ohto, Luu Tran, Franklin Reynolds, Graham Klyne, and Mark H. Butler. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0: Architecture. W3C recommendation, W3C, January 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/#CCPPArchitecture.

[42] Chris Woodrow, Johan Hjelm, Hidetaka Ohto, Luu Tran, Franklin Reynolds, Graham Klyne, and Mark H. Butler. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0: Extensibility and namespaces. W3C recommendation, W3C, January 2004. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/#ExtensibilityNamespaces.

[43] Wireless Application Protocol Forum. User Agent Profile (UAProf) Specification, October 2001. http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-248-uaprof-20011020-a.pdf.

[44] Wireless Application Protocol Forum. http://www.wapforum.org/.

[45] Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0: Appendix: Outline of request processing in HTTP. http://www.w3.org/TR/CCPP-struct-vocab/#Outline.

[46] Hidetaka Ohto and Johan Hjelm. CC/PP exchange protocol based on HTTP Extension Framework. W3C note, W3C, June 1999. http://www.w3.org/1999/06/NOTE-CCPPexchange-19990624.

[47] H. Nielsen, P. Leach, and S. Lawrence. An HTTP Extension Framework. RFC 2774 (Experimental), February 2000.

[48] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000.

[49] MPEG home page. http://www.chiariglione.org/mpeg/.

[50] International Organization for Standardization (ISO). http://www.iso.org/iso/home.htm.

[51] MPEG Convenor. Terms of reference. http://www.chiariglione.org/mpeg/terms_of_reference.htm.

[52] ISO/IEC. ISO/IEC 11172: Coding of Moving Pictures and Associated Audio at up to About 1.5 Mbit/s. Technical report, ISO, 1991.

[53] ISO/IEC. ISO/IEC 13818: Generic Coding of Moving Pictures and Associated Audio. Technical report, ISO, 1994.

[54] ISO/IEC. ISO/IEC 14496: Coding of Audio-Visual Objects. Technical report, ISO, 1994.

[55] DivX, Inc. DivX Video Player - DivX Video Codec - DivX Converter — DivX.com. http://www.divx.com/.

[56] The Xvid project. Xvid.org: Home of the Xvid Codec. http://www.xvid.org/.

[57] The x264 project. x264 - a free h264/avc encoder. http://www.videolan.org/developers/x264.html.

[58] ISO/IEC. ISO/IEC 15938: Multimedia Content Description Interface. Technical report, ISO, 2002.

[59] ISO/IEC. ISO/IEC 21000-1: Information technology — Multimedia framework (MPEG-21) — Part 1: Vision, Technologies and Strategy. Technical report, ISO, 2002.

[60] Belgacom n.v. Belgacom TV homepage. http://www.belgacomtv.be/.

[61] Telenet n.v. Telenet Digital TV. http://telenet.be/222/0/1/nl/thuis/televisie.html.

[62] Ian S. Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen. *The MPEG-21 Book*. John Wiley & Sons, Chichester, West Sussex, England, 2006.

[63] ISO/IEC. ISO/IEC 21000-7: Information technology — Multimedia framework (MPEG-21) — Part 7: Digital Item Adaptation. Technical report, ISO, 2004.

[64] Motion Picture Association of America. Motion Picture Association of America. http://mpaa.org.

[65] Wikipedia, the free encyclopedia. Wikipedia – homonymous hemianopsia. http://en.wikipedia.org/wiki/Homonymous_hemianopsia.

[66] G. Klyne and C. Newman. Date and Time on the Internet: Timestamps. RFC 3339 (Proposed Standard), July 2002.

[67] International Organization for Standardization. *ISO 8601:2000. Data elements and interchange formats — Information interchange — Representation of dates and times*. International Organization for Standardization, Geneva, Switzerland, 2000.

[68] David C. Fallside and Priscilla Walmsley. XML schema part 0: Primer second edition. W3C recommendation, W3C, October 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[69] The Apache Software Foundation. Xerces C++ Parser. http://xerces.apache.org/xerces-c/.

[70] Steven DeRose and James Clark. XML path language (XPath) version 1.0. W3C recommendation, W3C, November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[71] Open Source Native XML Database. http://exist.sourceforge.net/.

[72] The Apache Foundation. Apache Xindice. http://xml.apache.org/xindice/.

[73] PostgreSQL Global Development Group. PostgreSQL: Documentation: Manuals: PostgreSQL 8.3: XML Type. http://www.postgresql.org/docs/8.3/interactive/datatype-xml.html.

[74] PostgreSQL Global Development Group. PostgreSQL: The world's most advanced open source database. http://www.postgresql.org/.

[75] PostgreSQL Global Development Group. PostgreSQL: Documentation: Manuals: PostgreSQL 8.3: XML Functions. http://www.postgresql.org/docs/8.3/interactive/functions-xml.html.

[76] IBM Corporation. pureXML overview. http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.xml.doc/doc/c0022308.html.

[77] IBM Corporation. DB2 Product Family. http://www-01.ibm.com/software/data/db2/.

[78] Dmitry Lenkov and Oliver Goldman. XML binary characterization. W3C note, W3C, March 2005. http://www.w3.org/TR/2005/NOTE-xbc-characterization-20050331/.

[79] Mike Cokus and Santiago Pericas-Geertsen. XML binary characterization use cases. W3C note, W3C, March 2005. http://www.w3.org/TR/2005/NOTE-xbc-use-cases-20050331/.

[80] Mike Cokus and Santiago Pericas-Geertsen. XML binary characterization properties. W3C note, W3C, March 2005. http://www.w3.org/TR/2005/NOTE-xbc-properties-20050331/.

[81] Daniel Peintner and Santiago Pericas-Geertsen. Efficient XML interchange (EXI) primer. W3C working draft, W3C, December 2007. http://www.w3.org/TR/2007/WD-exi-primer-20071219/.

[82] Takuki Kamiya and John Schneider. Efficient XML Interchange (EXI) Format 1.0. W3C working draft, W3C, July 2008. http://www.w3.org/TR/2008/WD-exi-20080728/.

[83] Wireless Application Protocol Forum. Binary XML content format specification, June 2001. http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-192-wbxml-20010725-a.pdf.

[84] SyncML Initiative. SyncML Sync Protocol, version 1.1, February 2002. http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/syncml/syncml_sync_protocol_v11_20020215.pdf.

[85] Hartmut Liefke and Dan Suciu. Xmill: an efficient compressor for xml data. *SIGMOD Rec.*, 29(2):153–164, 2000.

[86] Michael Cokus and Daniel Winkowski. XML Sizing and Compression Study For Military Wireless Data. In *Proceedings of the XML 2002 Conference*, Baltimore, December 2002.

[87] WinZip® International LLC. WinZip® — The Zip File Utility for Windows — Zip/Unzip, Encrypt/Decrypt. http://www.winzip.com/index.htm.

[88] Christopher J. Augeri, Dursun A. Bulutoglu, Barry E. Mullins, Rusty O. Baldwin, and III Leemon C. Baird. An analysis of XML compression efficiency. In *ExpCS '07: Proceedings of the 2007 workshop on Experimental computer science*, page 7, New York, NY, USA, 2007. ACM.

[89] Hewlett Packard Labs. DELI: A Delivery Context Library For CC/PP and UAProf. http://delicon.sourceforge.net/.

[90] Robbie De Sutter, Frederik De Keukelaere, and Rik Van de Walle. Evaluation of usage environment description tools. In *Proceedings of the International Conference on Internet Computing*, pages 66–72, Las Vegas, June 2004.

[91] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 9–14, New York, NY, USA, 2005. ACM.

[92] Andrew Perkis, Yousri Abdeljaoued, Charilaos Christopoulos, Touradj Ebrahimi, and Joe F. Chicharo. Universal multimedia access from wired and wireless systems. *Circuits, Systems, and Signal Processing*, 20(3-4):387–402, 2001. `http://infoscience.epfl.ch/getfile.py?recid=111810&mode=best`.

[93] José M. Martínez. MPEG-7 tools for universal multimedia access. *J. Am. Soc. Inf. Sci. Technol.*, 58(9):1374–1376, 2007.

[94] E. Kasutani and T. Ebrahimi. New Frontiers in Universal Multimedia Access. Technical report, Ecublens, 2004. `http://infoscience.epfl.ch/getfile.py?mode=best&recid=87058`.

[95] H. Chaskar. Requirements of a Quality of Service (QoS) Solution for Mobile IP. RFC 3583 (Informational), September 2003.

[96] The Service Level Agreement Zone Homepage. `http://www.sla-zone.co.uk/`.

[97] Cisco Systems Inc. *Internetworking Technologies Handbook*. Cisco Press, third edition, 2000. `http://www.cisco.com/en/US/docs/internetworking/technology/handbook/QoS.pdf`.

[98] Wataru Okada, Fumihiro Kato, Kazuhiro Kitagawa, and Tatsuya Hagino. Applying CC/PP to User's Environmental Information for Web Service Customization. `http://www10.org/cdrom/posters/1066.pdf`.

[99] Henrik Frystyk Nielsen, Tim Berners-Lee, Jean-Francois Groff, et al. Libwww — the W3C Protocol Library — Homepage. `http://www.w3.org/Library/`.

[100] The Apache Software Foundation. The Apache HTTP Server Project. `http://httpd.apache.org/`.

[101] Adactus AS. Adactus AS — Home. `http://adactus.no/`.

[102] Adactus AS. Adactus mobilize. White Paper, December 2007. `http://www.adactus.no/images/stories/adactus_mobilize_whitepaper.pdf`.

[103] Sun Microsystems. The Java ME Platform homepage. `http://java.sun.com/javame/index.jsp`.

[104] Marco Lohse, Florian Winter, Michael Repplinger, and Philipp Slusallek. Network-integrated multimedia middleware (nmm). In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 1081–1084, New York, NY, USA, 2008. ACM.

[105] Motama GmbH. Features and Plug-ins of NMM. `http://www.motama.com/content/docs/features/index.htm#AEN99`.

[106] Adrian Nye. *Xlib programming manual (3rd ed.)*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1993. `http://www.x.org/wiki/ProgrammingDocumentation`.

[107] Steve Byrne, Arnaud Le Hors, Philippe Le Hégaret, Mike Champion, Gavin Nicol, Jonathan Robie, and Lauren Wood. Document object model (DOM) level 3 core specification. W3C recommendation, W3C, April 2004. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407.

[108] Johnny Stenback and Andy Heninger. Document object model (DOM) level 3 load and save specification. W3C recommendation, W3C, April 2004. http://www.w3.org/TR/2004/REC-DOM-Level-3-LS-20040407.

[109] Lee Thomason. TinyXml Main Page. http://www.grinninglizard.com/tinyxml/.

[110] Qt Software. Qt for Application Development. http://trolltech.com/products/appdev.

[111] Qt Software. Qt 4.4.3: QXmlStreamWriter Class Reference. http://doc.trolltech.com/4.4/qxmlstreamwriter.html.

[112] XQilla Homepage. http://xqilla.sourceforge.net/HomePage.

[113] SQLite Home Page. http://sqlite.org/.