Journal of Digital Contents Society Vol. 7. No. 1. March. 2006(pp. 25-33)

콘텐츠 적응화 시스템에서의 다양한 프로파일을 지원하기 위한 유연성 있는 매커니즘

임목화* 장병철** 차재혁*** 강수용****

요 약

유비쿼터스 시대가 도래함에 EK라 플랫폼 프로파일을 표현하는 다양한 방법돌이 논의되었고 표준으로 발표되고 있다. 이러한 다양한 플랫폼 프로파일들은 몇 가지 문제점을 안고 있는데 이러한 문제점을 해결하기 위해, 다양한 프로파일을 인식하고 프로파일의 단순화를 통하여 적응화 엔진의 성능을 향상 시키는 프로파일 처리 방식을 제안하고 프로파일링 매커니즘을 개발한다. 즉, 서로 다른 적응화 시스템에서 사용하는 프로파일을 인식하지 못하는 문제를 해결하기 위한 여러 프로파일의 통합방식을 제시하고, 프로파일의 속성을 필수 속성 및 유사 값으로 그룹핑하여 프로파일의 단순화를 실현한다. 이 시스템을 통하여 기존 적응화 시스템에 비교하여 다양한 프로파일 형식의 사용자 요청을 처리할 수 있게 되고, 컴팩트한 프로파일은 콘텐츠 적응화 시스템을 운영하며 동적으로 생성되는 프로파일의 개수를 대폭 줄이게 되어 메모리 사용을 줄이고 콘텐츠 변환의 횟수를 줄이면서 적응화 엔진의 성능을 향상시키는데 기여한다.

A flexible mechanism for supporting various profiles in the content adaptation systems

Mok-Hwa Lim* Byung-chul Chang** Jae-hyuk Cha*** Soo-young Kang****

Abstract

As becoming the age of ubiquitous, there are many discussion and publication for standards to express platform profile. These various platform profiles have some problems. To solve these problems, we propose and develop a new profiling mechanism which improve the performance of contents adaptation engine by simplifying platform profile and processing various type of platform profiles; we introduce the way to integrate different profiles to solve not to recognize the profile which is used by different adaptation system and simplify platform profile by grouping the similar attributes of platform profiles. Proposed system can processing different profiles and also the system overhead to processing a profile and creating a dynamic profile has reduced. As a result overall performance of content adaptation system has increased.

Key words : Device Profile

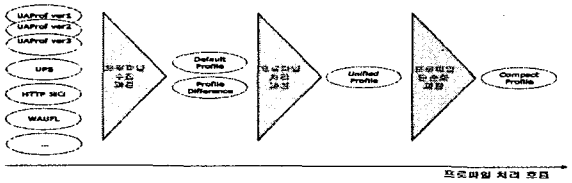
1. 서 론

근래 IT 기술의 발전과 더불어 다양한 정보 통신 기기들이 개발되고 있다. PC 뿐만 아니라 웹 콘텐츠들을 휴대전화, PDA 그리고 D-TV 등의 다양한 장치를 통하여 손쉽게 접할 수 있게 되었다. 그러나 현재는 이를 위한 해당 기기에 적합한 콘텐츠들이 부족하고, 기존 웹 콘텐츠를 재사용하기가 쉽지 않다. 이에 기존에 개발된 콘텐츠 혹은 특정 플랫폼에 맞게 개발된 콘텐츠의 맥락을 유지하면서도 다양한 플랫폼에 효율적이고 적합하게 콘텐츠를 변형하는 연구가 수행되고 있다.

위와 같이 다양한 플랫폼에 맞게 콘텐츠를 변형하는 것을 콘텐츠 적응화(Content Adaptation)[1]라고 정의한다. 콘텐츠 적응화를 수행하기 위해서는, 콘텐츠 서버에 접속하려는 사용자가 이용하는 기기의 특성을 정확히 아는 것이 필요하다. 즉, 클라이언트 플랫폼의 형태를 알아야 가장 적합하게 콘텐츠를 변형하여 전송할 수 있기 때문이다. 여기서 사용자가 이용하는 정보기기의 성능 및 상태의 특성 값들을 플랫폼 프로파일(Platform Profile)이라고 정의한다. 본 연구에서의 프로파일링 메커니즘 개발의 목적은 종래의

※ 제일저자(First Author) 임목화
※ 교신저자(Corresponding Author) 차재혁
접수일 2005년 8월 24일, 완료일 2005년 12월 2일
* 한양대학교 정보통신대학원 석사
moka113@gmail.com
** 한양대학교 일반 대학원 박사 과정
*** 한양대학교 정보통신학부 교수
**** 한양대학교 컴퓨터교육과 조교수
이 논문은 서울시정개발연구원 '유비쿼터스 지향적 OSMU기반의 미래형교육 시스템 개발 및 기술 이전' 연구비 지원에 의하여 연구되었음

프로파일 처리에서 가지고 있던 비효율적인 방식을 개선함에 있다. 기존의 프로파일 표현 방식은 실제 적응화 엔진에서 별로 사용되어 지지 않은 속성들을 많이 포함하고 있다. 본 연구에서는 다음의 [그림 1]과 같이 프로파일 처리를 수행함으로써, 여러 종류의 다양한 프로파일을 수용가능하고 다양한 프로파일에서 수집된 내용을 적응화 엔진에서 필수적인 요소만을 추출하여 적응화 엔진의 효율을 높일 수 있는 프로파일링 메커니즘을 제안한다.

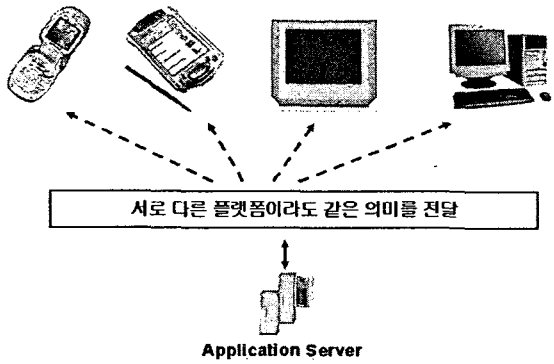


[그림 1] 제안된 프로파일의 처리 흐름

2. 관련 연구

2.1 콘텐츠 적응화

콘텐츠 적응화란 하나의 콘텐츠를 서로 다른 플랫폼에 적합한 형태로 변환하는 것이다. [그림 2]는 콘텐츠 적응화의 개념을 보여주고 있다.



[그림 2] 콘텐츠 적응화 시스템

콘텐츠 적응화는 적응화를 이루는 방법, 적응화의 대상 등에 따라 IBM의 Transcoding Publisher에서 사용한 주석기반 변환(Annotation-Based Transcoding)[2], XML 문서 변환 기술[3], MS에서 제안한 Function-based Object 모델[4], 네델란드 CWI의 Multimedia Transformation Engine[5] 등 다양한 시스템이 있다.

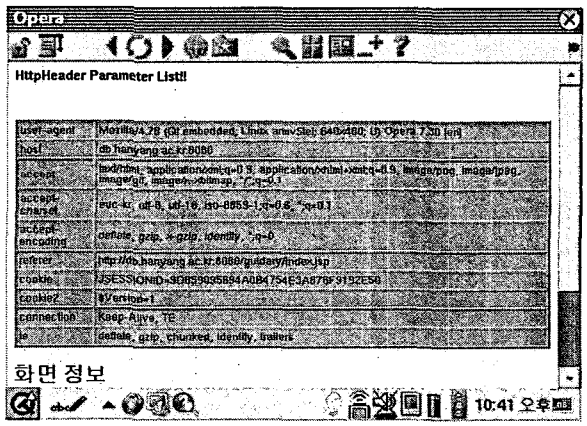
2.2 플랫폼 프로파일 모델

사용자가 웹 콘텐츠에 접속하기 위해 이용하는 시스템의 운영 체제 및 브라우저는 화면의 크기, 멀티미디어 요소의 지원, 플러그인의 설치 유무 등의 서로 다른 특징과 성능을 가지고 있다. 이러한 장치간의 차이 때문에, 해당 장치에 적합한 콘텐츠를 전송하기 위해서 장치의 성능을 자세히 기술하고 있는 플랫폼 프로파일이 반드시 필요하다.

2.2.1 HTTP 헤더 기반 모델

기존의 콘텐츠 적응화 시스템에서는 클라이언트를 제어하거나 웹 브라우저를 수정하는 작업을 하지 않기 위해 HTTP 요청 헤더가 기본적으로 전송해 주는 필드의 정보만을 이용하였다.

[그림 3]은 샤프 자우루스(Sharp Zaurus) PDA의 HTTP 요청 헤더 정보 화면을 캡쳐한 것이다. User-agent 의 값을 보면 간략하게 운영체제, 화면 크기, 브라우저 정보 등을 포함하고 있는 것을 알 수 있다.



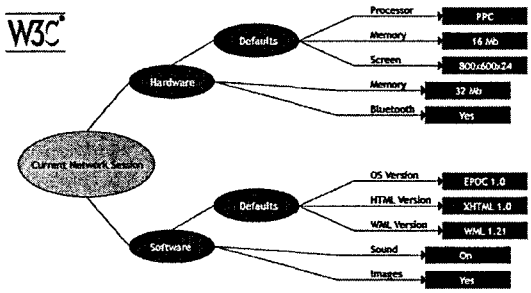
[그림 3] 자우루스 HTTP 헤더

프로파일링 메커니즘을 구현하기 위해 각각의 브라우저가 표현하는 헤더 정보를 직접 확인하여 단순히 해당 문자열이 포함되어 있는지 아닌지를 확인하는 정도였다. 따라서 이런 시스템에서 획득할 수 있는 클라이언트 정보는 브라우저 버전이나 이미지 타입 등 극히 제한적이다.

2.2.2 CC/PP

단순한 HTTP 분석 방식의 단점을 해결하기 위하여 W3C의 DIWG (Device Independence Working Group)에서는 단말 장치의 성능과 사용자 성향까지 표현할 수 있는 CC/PP (Composite Capabilities / Preferences Profile)를 제정, 권고

하였다[6]. [그림 4]처럼 CC/PP를 준수하는 프로파일은 콤포넌트(Component)와 속성 (Attributes)의 2단계로 구성되며, 하나의 프로파일은 여러 개의 콤포넌트를 가지고, 하나의 콤포넌트는 여러 개의 속성들을 포함한다. 이러한 프로파일은 주로 RDF(Resource Descript Framework)의 XML 직렬화(Serialization)를 사용하여 표현한다.



[그림 4] CC/PP 형식의 프로파일의 예

CC/PP의 단점은 2가지가 있다. 첫째는 표준 어휘(standard vocabulary)를 제공하지 않는 다는 것이고, 둘째는 전송 방식에 대해서 구체적으로 제시하지 않는 다는 것이다. CC/PP를 준수하며 실제 적응할 수 있는 스펙 중에 대표적인 것이 OMA(Open Mobile Alliance)에서 제안한 UAProf(User Agent Profile)[7]와 프랑스 INRIA의 UPS(Universal Profiling Schema)[8]가 있다.

2.2.3 Merkwelt

Merkwelt[9]는 정보 기기의 성능을 인식하는 자바 기반 오픈 소스 프레임워크로 소스포지(sourceforge) 사이트를 통해 공개되어 개발 중이며 Stan Wiechers에 의해 최초 개발되었고 현재 유지 보수 중이다.

이 시스템에서 사용하는 프로파일은 다음의 3가지 Properties로 구성된다. Identifier Properties는 HTTP 헤더의 이름을 표현하는 regexInput과 정규식(regular expression)을 표현하는 regexPattern 속성을 갖는다. 둘째, Dynamic Properties는 HTTP 헤더로부터 필요한 속성을 추출하는 것으로 프로파일 속성의 이름을 표현하는 Name, 추출하기 위한 HTTP 헤더의 이름을 표현하는 regexInput 그리고 추출을 위한 진보된 정규식을 표현하는 regexPattern 속성을 갖는다. 마지막으로 해당 기기의 정적인 값을 표현하는 Static Properties로 이루어진다.

2.2.4 Wireless Univeral Resource File

WAUFL[10]은 장치의 유사 속성(family of device)의 개념을 기반으로 만들어진 알려진 대부분의 무선 장치에 대한 정보를 포함하고 하나의 파일로 이루어진 플랫폼 프로파일이다. 이 역시 소스포지에서 공개되어 연구되고 있다.

이러한 fall_back 속성을 이용함으로써 WAUFL은 전체 표현하는 값들에 비해 매우 단순한 표현 방식과 업데이트가 아주 쉽다는 장점이 있다.

이 WAUFL 파일을 사용하는 라이브러리로 WAUFL을 제안한 Luca Passani에 의해 만들어진 Java 기반 WAUFL 라이브러리인 WALL이 대표적이다. 그 외에도 PHP, PERL, Python, dotNet 등을 이용한 라이브러리가 개발 중에 있다.

2.3 플랫폼 프로파일 라이브러리

2.3.1 DELI

HP에서 Mark H. Butler에 의해 개발된 DELI(A Delivery context Library for CC/PP and UAProf)[11]는 CC/PP나 UAProf를 처리할 수 있는 오픈 소스 라이브러리이다. DELI는 미리 저장된 참조 프로파일(Default Profile)과 디바이스가 전송한 상위(相違) 프로파일(Profile Differences)을 CC/PP 표준에 맞춰 조합해주는 역할을 한다. 그러나 DELI는 단말 장치의 성능을 수집하는 방식이 아니므로 플랫폼 특성 중 사용자가 변경한 값을 제대로 반영할 수 없고, 신형 플랫폼인 경우 해당 제품의 플랫폼 정보를 서버에 등록해줘야 하는 문제가 있다.

2.3.2 JSR-188 : CC/PP Processing

JSR-188[12]은 호환을 가능하게 하기 위하여 CC/PP 정보를 처리하기 위한 API 세트를 정의하는 규격으로 SUN에서 오픈 소스로 제공하는 J2EE 플랫폼을 위한 자바 확장판이다.

DELI에서 사용한 어휘 정의 파일과 다르게, JSR-188 스펙에서 사용하는 어휘 정의 파일은 <attribute-desc>의 하위 요소가 아닌 속성으로 이름, 속성 타입, 콜렉션 타입, 해결 규칙을 정의한다. 어휘에 정의 되지 않은 속성에 대해서는 현재 처리되고 있는 콤포넌트로 추가하는 특징을 가지며 각 속성별 데이터 타입을 구분하여 처리 할 수 있다. 본 연구에서 구현하는 기본적인 CC/PP 프로파일의 처리는 이 스펙을 준수하여 구현하도

록 하였다.

3. 기존 프로파일 처리의 문제점

지금까지 콘텐츠 적응화 환경에서의 프로파일링 기술로써 다양한 프로파일 모델과 라이브러리에 대해 언급하였다. 이러한 방법들은 여러 가지 콘텐츠 적응화 시스템에서 혼재 되어 사용되고 있으며, 각각의 프로파일들은 프로파일링 과정에서 다소간의 문제점을 가지고 있다.

3.1 프로파일의 복잡성

첫째는 프로파일의 복잡성이라는 문제점을 들 수 있다. 지금까지 프로파일 모델의 연구는 가능한 많은 기기의 상태 정보를 표현하는데 목적을 두고 그 덩치를 계속 키워왔다. 그 결과 UAProf 같은 경우 가장 최근 버전에서 100여 가지를 훨씬 넘는 속성을 포함하고 있음을 알 수 있다. 이렇게 복잡한 프로파일은 실제로 적응화 엔진에서 필요로 하지 않는 많은 속성을 포함하고 있다.

<표 1> 적응화 시스템에서 사용하는 속성의 수

프로파일 이름 또는 적응화 시스템	속성의 개수
UAProf	100여개
WebSphere Transcoding Publisher	23
SMIL 2.0 testAttribute[13]	12
Media Queries[14]	11
CONNEG[15]	4

다양한 적응화 시스템에서 사용하고 있는 플랫폼의 필수 속성의 개수는 <표 1>에서 보듯이, 순수하게 플랫폼 프로파일 표준인 UAProf는 100여개를 훨씬 넘는 반면에 가장 복잡한 속성하는 WebSphere Transcoding Publisher가 23개만을 사용하고 있다. 이러한 수치를 통해 현재 구현되는 콘텐츠 적응화 시스템에서 실제 사용되는 속성보다 현재의 플랫폼 프로파일 표준이 너무 많은 내용을 포함하고 있음을 알 수 있다.

3.2 프로파일 간의 상호 호환성 부족

또 하나의 문제점은 다양한 프로파일들이 같은 목적을 위해 만들어 졌음에도 불구하고 서로간의 상호호환성이 고려되지 않아, 대부분의 적응화 시스템에서는 자신이 사용하는 프로파일 형식만을 처리한다. 이 문제점은 다시 2가지로 나눌 수 있는데, 하나는 같은 어휘를 사용하더라도 같은 어휘를 사용하더라도 버전이 다르면 같은 속성으로 인식하지 못한다는 점과 다른 어휘를 사용하는 경우이다.

```
<?xml version="1.0"?>
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
   xmlns:prf="http://www.wapforum.org/profiles/UAPROF/
     ccppschema-20010430#">
                                    ...
   <prf:ScreenSize>320x240</prf:ScreenSize>
                                    ...
```

[그림 5] 이름 공간과 연관된 CC/PP 속성

```
<?xml version="1.0"?>
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ccpp="http://www.w3.org/2002/11/08-ccpp-schema#"
   xmlns:prf="http://www.wapforum.org/UAPROF/ccppschema-
   20000405#">
          ...    <prf:ScreenSize>320x240</prf:ScreenSize>...
```

[그림 6] 다른 이름 공간을 가리키는 속성

```
...
  <neg:display>101X52Pixels</neg:display>
...
```

[그림 7] UPS에서 사용하는 화면 크기의 속성

먼저 전자의 프로파일간의 상호 호환성이 부족한 예를 직접 들어보면 다음과 같다. CC/PP Architecture 표준 문서에서 CC/PP 프로파일의 확장성과 네임스페이스 (Extensibility and Namespaces)라는 설명 부분이 있다[16]. 여기서 CC/PP의 확장이라는 것은 새로운 속성 어휘(attribute vocabularies)를 소개하는 것이고, 상호호환성(Interoperability)을 보장하기 위해서는 어휘를 보다 일반적으로 사용되어 질 수 있게 정의 하거나, 매우 작은 핵심 어휘만을 사용하기를 강력히 권장한다고 되어 있다. [그림 5]의 프로파일 예제는 속성의 이름이 각 어휘의 네임 스페이스와 관련이 있음을 보여주는 것이다.

[그림 5]와 비교하여 [그림 6]을 보면 <prf:ScreenSize>라는 같은 속성의 이름을 사용하는 것 같지만, 'prf'가 참조하고 있는 네임 스페이스의 주소가 서로 다르므로 속성의 이름이 전혀 다른 이름으로 인식된다.

[그림 7]은 UPS에서 사용하는 화면 크기를 표현하는 방식을 보여준다. UAProf의 화면 크기를 표현하는 속성과 2가지 차이점이 있다. 화면 크기를 나타내는 의미는 분명히 같으나, UAProf에서

는 'ScreenSize' 라는 이름을 사용하고, UPS에서는 'display' 라는 이름을 사용한다. 그리고 또 하나의 차이점은 화면 크기에 대한 값의 표기 방식인데, UAProf는 [숫자]x[숫자] 방식이나 UPS에서는 UAProf의 방식에 "Pixels" 라는 문자열을 포함하는 형식이다.

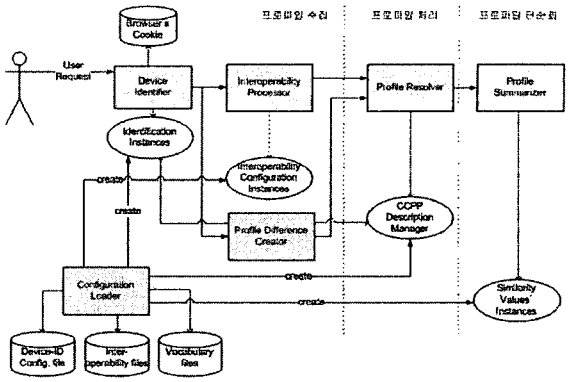
현재까지의 플랫폼 프로파일 연구에서는 이를 해결한 근본적인 해결책을 제시하지 않고 있다. CC/PP 표준의 경우에는 같은 CC/PP를 따르는 다양한 어휘의 프로파일들의 상호호환성 문제를 근본적으로 해결하지 못하고 반드시 필요한 최소한의 속성만을 사용하여 이를 해결하기를 권장한다. 또한 여러 적응화 시스템에서는 이러한 차이점을 처리하지 못하고, 자신이 정한 형식의 플랫폼 프로파일 형식만을 사용하고 있다.

4. 제안 시스템

4.1 전체 구성

본 연구에서 제안하는 프로파일링 시스템은 [그림 8]에서 보듯이, 크게 프로파일 수집, 프로파일 처리 그리고 프로파일 단순화 과정으로 나누어진다. 프로파일링 시스템은 총 6개의 모듈로 이루어져 있으며 첫째는 전체 시스템을 수행하기 위해 설정 파일들을 인스턴스화 하는 Configuration Loader이며 이는 서버가 구동되어질 때 한 번 실행되는 모듈이다. 그 외 5개의 모듈은 사용자로부터 요청이 있을 때마다 사용자의 프로파일을 분석하는 역할을 수행한다.

이 5개의 모듈은 앞에서 말한 3개의 과정으로 분류할 수 있다. 처음 프로파일 수집과정에서 Device Identifier, Interoperability Processor 그리고 Profile Difference Creator가 포함되어 있으며 이 모듈들을 통해 기본적인 CC/PP 프로파일 해석을 수행하는 Profile Resolver에 필요한 2개의 프로파일을 생성한다. 다음에 프로파일 처리 과정에서는 Profile Resolver가 프로파일 수집 과정의 결과물을 처리하여 하나의 프로파일로 만들어 주고, 이 하나의 큰 프로파일은 프로파일 단순화 처리를 하는 Profile Summarizer를 통해 실제 적응화 엔진에서 사용하는 속성만을 간추리고 유사 값들을 그룹핑 함으로써 컴팩트한 프로파일을 완성하게 된다. 본 프로파일링 시스템의 특징은 설정 파일을 조금만 변경함으로써 새로 생성된 어휘들이나 다른 방식의 프로파일 표현방식을 포용할 수 있어 적응화 엔진의 효용성을 높일 수 있다는 것이다.



[그림 8] 전체 구성도

4.2 프로파일 수집 과정

[그림 21]은 'mspie.rdf'를 Default Profile로 사용하는 장치는 <identifier> 요소에서 기술하듯이 HTTP 헤더의 ua-os라는 항목의 값은 문자열 중간에 "Pocket" 이라는 문자열을 포함하고 있음을 의미한다. <dynamic> 요소로부터 플랫폼 프로파일의 "ScreenSize" 정보는 HTTP 헤더의 ua-pixels 라는 항목의 값으로부터 숫자 그리고 'x' 또는 'X' 문자 그리고 숫자로 이어진 정보를 그대로 가져오라는 의미이다.

```
<device value="mspie.rdf">
  <identifier input="ua-os" pattern=".+Pocket.+" />
  <dynamic elem="ScreenSize" input="ua-pixels"
pattern="(₩d+[x|X]₩d+)"/>
  <dynamic elem="BitsPerPixel" input="ua-color"
pattern="color(₩d+)"/>
</device>
```

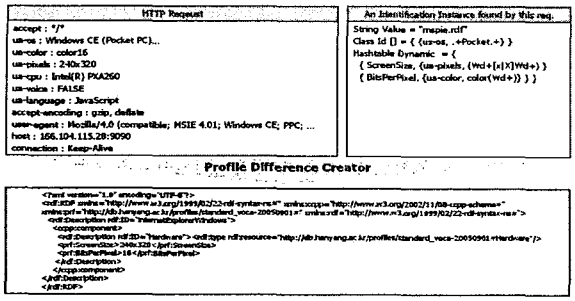
[그림 9] Pocket IE의 Identification 정보

Device Identifier는 웹 서버에서 동작하는 프로그램으로 프로파일 분석의 시발점 역할을 하는 것이 Device Identifier 이다.

이 모듈의 첫번째 역할은 HTTP 요청 헤더를 분석하여 사용자 단말을 구분하여 Default Profile을 선택하여 Interoperability Processor를 호출한다. 사용자 기기를 구분하기 위해서 기존에 DELI에서 사용하던 단순 문자열 비교 방식에서 발전하여 보다 유연성 있게 기기를 구분하기 위해 정규식(Regular Expression)을 이용하여 처리한다. 이 때 사용된 자료 구조는 4.2.1.1 Identification Instance의 구조를 참조한다.

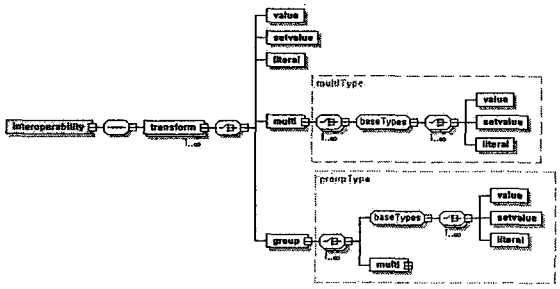
두번째 역할은 HTTP 요청 헤더를 매개 변수로 하여 Profile Difference Creator를 호출하는 것이

다. [그림 10]의 위쪽에 표현된 사용자가 요청한 HTTP 요청 헤더의 내용과 4.2.1.1 Identification Instance의 Pocket Internet Explorer의 *Identification* 정보를 이용하여 *Profile Difference* Creator를 통해서 새로 생성된 Profile Difference는 그림의 아래 부분과 같다.



[그림 10] Profile Difference Creator의 기능

마지막 역할은 매번 접속할 때마다 플랫폼 프로파일 분석기의 전 과정을 실행하여 시스템에 과부하를 주는 것을 방지하기 위해 쿠키를 이용하여 프로파일 관리 작업을 수행하는 것이다. 이렇게 저장된 쿠키 정보가 있을 경우, 프로파일링의 모든 과정을 생략하고 적응화 엔진을 바로 호출함으로써, 대부분 자주 변경되지 않은 플랫폼 프로파일에 대한 처리를 매번 하지 않게 되어 응답 시간을 높일 수 있다.



[그림 11] 상호호환성 기술 문서의 스키마 구조

기본적으로 모든 프로파일은 테이블 형식이고 제한된 데이터 타입만을 사용하고 단순 값과 집합 값을 적용할 수 있으며 누구나 이해할 수 있는 구조로 [그림 11]과 같은 XML 스키마 구조를 제안한다. 프로파일 변환의 가장 기본적은 타입으로 일반적인 단일 값으로부터 읽어오는 <value>, 집합 값으로부터 읽어 오는 <setvalue> 그리고 사용자가 직접 입력한 문자열을 읽어오는 <literal> 요소가 있다. 그리고 앞의 3가지 요소를 자식으로 포함함으로써 다중 값을 단일 값으로 연결하는 <multi>와, 다중 값을 여러 개의 단일 값인 집합 값으로 변환하는 <group> 요소로 이루어진다.

Profile Difference Creator는 HTTP 헤더로부터 필요한 Dynamic Value를 추출하여 Profile Difference를 생성하는 역할을 하는 모듈이 Profile Difference Creator 이다. 이는 W-HTTP 프로토콜을 지원하지 않는 브라우저에 대해서도 Dynamic 값을 통해 Profile Difference를 얻을 수 있다.

4.3 프로파일 처리 과정

프로파일 처리 과정에서는 이전의 프로파일 수집 단계를 거쳐 생성된 CC/PP 표준을 준수하는 같은 스키마를 갖는 2가지의 프로파일을 하나의 프로파일로 통합하는 역할을 수행한다. Profile Resolver는 JSR-188에서 제안하는 기본적인 CC/PP 프로파일 처리 구현 스펙에 따라 다음의 여러 가지 속성들에 대해 프로파일을 재가공하는 모듈이다.

4.4 프로파일 단순화 과정

프로파일 단순화 과정은 Profile Summarizer 라는 1개의 모듈로 이루어져 있으며, 앞의 프로파일 처리 과정에서 생성된 통합되어 하나의 인스턴스로 생성된 프로파일을 입력 받아 보다 콤팩트한 프로파일로 변환하는 과정이다. Profile Summarizer는 적응화 엔진이 처리할 수 있는 속성을 추출하고, 그 속성에서 사용자가 미리 정의한 값으로 매핑하여 프로파일을 단순화 시키는 모듈이다. 이 모듈이 단순화 하는 작업은 다음과 같이 2가지로 볼 수 있다.

- 적응화 엔진에서 사용하는 속성만을 추출
- 각 속성에서 자주 사용되는 유사한 값으로 그룹화

이렇게 프로파일을 단순화시킴으로써 얻을 수 있는 장점은 적응화 엔진에서 사용하는 속성만을 추출하기 때문에 실제로 사용하지 않는 속성에 대한 고려를 필요로 하지 않는다는 것과, 유사한 값들을 하나의 값으로 그룹화 시키는 작업을 통해 사용자에 의해 생성되어 실시간으로 관리되는 프로파일의 인스턴스의 수가 감소된다는 것이다. 프로파일 수가 감소하면 프로파일을 기준으로 적응화 콘텐츠를 캐싱할 경우에 캐싱에 대한 히트

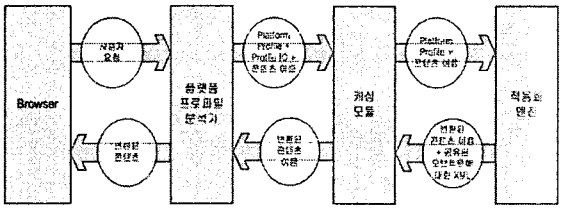
율을 높일 수 있고 적응화 수행 횟수의 감소로 엔진의 성능을 향상시킬 수 있다.

5. 구현 및 실험

5.1 실험 환경

5.1.1 플랫폼 프로파일 처리기의 위치

일반적으로 콘텐츠 적응화 시스템은 2가지 타입이 존재한다. 첫째는 플랫폼 프로파일 분석기와 적응화 엔진으로 이루어진 형태와 이에 캐싱 모듈이 추가되어 있는 형태이다. [그림 11]은 후자의 경우로, 콘텐츠 적응화 시스템의 각 모듈들의 처리 흐름을 보여준다.



[그림 12] 콘텐츠 적응화 시스템

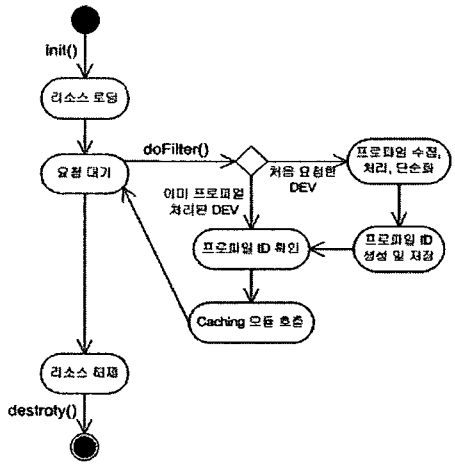
[그림 12]에서 보듯이 콘텐츠 적응화 시스템에서의 플랫폼 프로파일 분석기의 위치는 브라우저에서 발생하는 사용자 요청을 받아들이는 서블릿 프로그램을 통하여 콘텐츠 적응화 시스템의 시발점이 된다. 그리고 캐싱 모듈에서 이미 변환된 콘텐츠의 유무를 확인하고 없을 경우 적응화 엔진을 호출하여 콘텐츠 변환을 수행한다.

5.1.2 플랫폼 프로파일 처리기의 역할

콘텐츠 적응화 시스템의 모든 요청은 플랫폼 프로파일 처리, 캐싱 모듈 그리고 콘텐츠 적응화 엔진의 동작을 수행해야 한다. 기존의 웹 콘텐츠 요청에 특정 작업을 추가 수행하기 위해서는 기존의 콘텐츠의 각 소스에 직접 각각의 모듈에 대한 코드를 삽입하거나, 특정 페이지를 통해서만 콘텐츠에 접근하는 방식을 사용했다.

본 시스템에서는 이러한 단점을 보완하기 위해 동적으로 요청과 응답을 가로채고, 가로챈 응답 속에 포함된 정보를 전송하는 서블릿의 특별한 유형인 서블릿 필터(Servlet Filter)[17]를 사용하여 사용자의 모든 요청을 처리한다. 그리고 서블릿의 포워드(forward)를 사용함으로써 Device Independence Principle[18]에서 제약한 모든 클라이언트는 모양은 다른 콘텐츠 이더라도 이들의 주소는 모두 동일해야 함을 보장하였다.

[그림 13]는 콘텐츠 적응화 시스템을 위한 서블릿 필터의 동작 흐름을 간단하게 표현한 것이다. 웹 서버를 실행 시킬때 이 서블릿 필터의 init() 함수가 호출되면서 콘텐츠 적응화 시스템에 필요한 다양한 리소스를 로딩 시킨 후, 사용자 요청이 들어올 때까지 대기 상태가 된다. 사용자가 어떤 콘텐츠를 요청하는 순간, 서블릿 필터의 doFilter()가 수행되면서 이 함수에 정의된 다양한 콘텐츠 적응화 관련 작업들을 수행한다. 플랫폼 프로파일 처리기가 하는 일은 프로파일을 수집하고, 매번 프로파일 검사를 수행하지 않기 위해 이미 처리된 장치에 대한 프로파일에 대한 ID를 부여하여 관리하며 다양한 프로파일 관련 처리를 수행한다. 그리고 요청된 콘텐츠를 캐싱 모듈을 호출하여 변환된 콘텐츠를 요청한다. 그리고 마지막으로 웹 서버가 종료될 때, 서블릿 필터의 destroy() 함수가 호출되면서, 다양한 리소스를 해제하면서 서비스를 종료한다.



[그림 13] 서블릿 필터에서 동작 순서

6. 결론 및 향후 연구

6.1 결론

프로파일링 메커니즘을 구현한 본 시스템은 다양한 타입의 프로파일을 런타임시에 동적으로 변환하여 프로파일간 상호호환성을 보장하며 프로파일의 필수 속성만을 추출하여 콤팩트한 프로파일로 변환할 수 있다.

본 시스템은 여러 모듈들을 통해 기존 프로파일 처리 시스템에서 제공하지 못했던 서로 다른 프로파일 간의 상호 호환성을 제공할 수 있다. 그리고 현재 본 메커니즘에 적용되지 않은 또 다른 종류의 프로파일이 등장하여도 본 연구에서 제안하는 인터페이스를 따라서 작성함으로써 간단히 적용이 가능하다.

기존 시스템에서 고려하지 못한, 프로파일 단순화를 통해 적응화 엔진진에서 적용 대상이 아닌 속성들을 미리 제거할 수 있고, 어떤 속성의 값이 다르더라도 유사한 값으로 그룹화하여 비슷하지만 조금 다른 성능을 지닌 클라이언트 장치들 간에 같은 내용을 갖는 프로파일로 만들 수 있다. 결과적으로 수없이 증가하는 프로파일 속성을 추상화하여 프로파일 수가 많이 줄어들 것이며 그로 인해 적응화 엔진의 성능 향상을 꾀할 수 있을 것이다.

6.2 향후 연구

본 연구에서는 프로파일 단순화 과정을 위하여 여러 적응화 엔진의 동작 패턴을 분석하면서 직관적인 방식을 이용하였다. 향후 연구에서는 프로파일의 사용 측면에서 보다 많은 정보를 수렴하고 사용자의 콘텐츠 이용 만족도 등을 분석하여 인공지능적으로 유사한 값을 찾기를 시도하는 효율적인 그룹핑 방식에 대한 연구가 필요할 것이다. 그리고 아직 본 프로파일링 메커니즘에 적용되지 못한 여전히 많이 연구되고 있는 플랫폼 프로파일을 보다 효과적으로 손쉽게 적용 시킬 수 있는 방안에 대한 연구도 필요하다.

본 연구에서 구현한 프로파일링 메커니즘에서는 다양한 설정 파일을 사용하는데, 이를 시각적으로 편집하고 등록할 수 있는 GUI 기반의 편집기의 부재로 인해 프로파일링 시스템의 구축에 어려움이 있다. 따라서 본 메커니즘에 적합한 GUI 기반 편집기의 개발을 통하여 시스템의 유지보수성과 확장성을 더욱 증가 시킬 필요가 있다.

Reference

[1] W3C DIWG, Device Independence Principles, Sep 2001, W3C Device Independence Working Group, http://www.w3.org/2001/di/

[2] Masahiro Hori, Annotation-Based Web Content Transcoding, Computer Networks, Vol. 33, No. 1-6 (Proc. of WWW9), pp. 197-211, June 2000

[3] Tiong Goh, Getting Ready For Mobile Learning, Proceedings of ED-MEDIA 2004, June 2004

[4] Jinlin Chen, Function-based Object Model Towards Website Adaptation, ACM 1-58113-348-0/01/0005, May 2001

[5] J.R. van Ossenbruggen, Cuypers: a semi-automatic hypermedia generation system, Report INS-R0025 ISSN 1386-3681, Dec 2000

[6] F.Manola and E.Miller, Composite Capability / Preference Profiles (CC/PP) Structure and Vocabularies 1.0, http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, Jan 2004

[7] Wireless Application Group, User Agent Profile Specification, Wireless Application Protocol Forum, Nov 1999

[8] Tayeb Lemlouma, Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments, http://www.w3.org/2002/02/DIWS/submission/tlemlouma-W3CPositionPaper03-2002.htm, Feb 2002

[9] Stan Wiechers, Merkwelt::Profile Resolver Service, http://merkwelt.sourceforge.net, Jun 2004

[10] Luca Passani, WAUFL, Wireless Universal Resource File, http://wurfl.sourceforge.net

[11] M. Butler, DELI: A Delivery context Library for CC/PP and UAProf, Exteranl Technical Report HPL-2001-260, 1998

[12] Sun Microsystems, Inc., CC/PP Processing Specification Version 1.0, Sep 2003

[13] Dick Bulterman, SMIL 2.0 Content Control Modules, W3C Recommendation, http://www.w3.org/TR/2005/REC-SMIL2-20050107/smil-content.html, Jan 2005

[14] Hakon Wium Lie, Media Queries, W3C Candidate Recommendation, Jul 2002

[15] L. Masinter, Media Features for Display, Print, and Fax, RFC 2534, Mar 1999

[16] Tim Bray, Namespaces in XML, http://www.w3.org/TR/REC-xml-names/, Jan 1999

[17] Sun Microsystems, The Essentials of Filters, http://java.sun.com/products/servlet/Filters.html

[18] Roger Gimson, Device Independence Principles, http://www.w3.org/TR/2003/NOTE-di-princ-20030901/, Sep 2003

임 목 화

2001년 목포대학교 컴퓨터공학과 (공학사)
2006년 한양대학교 정보통신대학원 (공학석사)
2006년~현재 삼성전자 디지털 프린팅 사업부 S/W 개발 그룹

장 병 철

1996년 안동대학교 컴퓨터공학과(학사)
2001년 한양대학교 컴퓨터교육과(석사)
2003년~현재 한양대학교 정보통신학과 박사과정
관심분야 : 데이터마이닝, 시맨틱웹, e-Learning

차 재 혁

1987년 서울대학교 계산통계학과(학사)
1991년 서울대학교 컴퓨터공학과(석사)
1997년 서울대학교 컴퓨터공학과(박사)
1997년~1998년 한국학술진흥재단 부설 첨단학술정보센터선임연구원
1998년~2001년 한양대학교 사범대학 컴퓨터교육과 조교수
2001년~현재 한양대학교 정보통신대학 정보통신학부 교수
관심분야 : XML, 데이터베이스, 플래시 메모리 기반 저장시스템, 멀티미디어 콘텐츠 적응화, e-Learning

강 수 용

1996년 서울대학교 수학과(학사)
1998년 서울대학교 전산과학과(석사)
2002년 서울대학교 컴퓨터공학과(박사)
2000년~2002년 SIGn Co., 선임연구원
2002년~2003년 서울대학교 Postdoctoral researcher
2003년~현재 한양대학교 컴퓨터교육과 조교수