(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl.[7]
G06F 17/00
H04L 29/06

(11) 공개번호 10-2005-0044379
(43) 공개일자 2005년05월12일

(21) 출원번호 10-2004-7007024
(22) 출원일자 2004년05월07일
번역문 제출일자 2004년05월07일
(86) 국제출원번호 PCT/US2002/036064
국제출원출원일자 2002년11월07일
(87) 국제공개번호 WO 2003/040893
국제공개일자 2003년05월15일

(30) 우선권주장 10/010,616 2001년11월08일 미국(US)

(71) 출원인 라이트서프 테크놀로지스, 인크.
미합중국 캘리포니아 (우편번호:95060) 산타 크루즈 쿠퍼 스트리트 110 포쓰 플로어
(72) 발명자 미에쯔에글리폴
미국95066캘리포니아주스코츠밸리블루베리드라이브116
키라니셰카르
미국95010캘리포니아주캐피톨라와시번애브뉴109
이스워벤카트
미국95014캘리포니아주쿠퍼티노린다비스타드라이브10736

(74) 대리인 특허법인코리아나

심사청구 : 없음

(54) 성능에 기초하여 복수의 다른 클라이언트 장치로 미디어를전송하는 시스템 및 방법

명세서

저작권 공지

컴퓨터 프로그램 리스트 부록

컴퓨터 프로그램 리스트를 포함하는 컴퓨터 프로그램 리스트 부록 A 는 이 출원에 포함되어 있다. 본 명세서에서는, 컴퓨터 프로그램 리스트 부록 A 의 개시를 참조로 한다.

발명의 배경

본 발명은 일반적으로 정보 처리에 대한 것으로, 더욱 자세하게는, 디지털 미디어와 관련 정보에 대한 온라인 액세스를 개선하는 방법을 제공하는 온라인 시스템에 대한 것이다.

현재, 소비자들은 여러 디지털 이미지, 디지털 비디오, 및 디지털 오디오 제품들을 이용할 수 있다. 디지털 미디어가 어떻게 기록되었는지의 방법과 관계없이, 이후 언제 어디서든, 정보가 다른 장치에 대해 이용가능해야 하는, 즉, 더 큰 네트워크의 디지털 장치에 대해 이용가능하여, 이러한 정보가 스크린 상에 디스플레이되거나 하드카피로 프린트되거나 저장되거나 다른 사람과 공유될 수 있다. 오늘날, 매우 많은 수의 웹 사이트가 성능을 가진 서버 컴퓨터를 이용하여 인터넷 상에 존재하여, 이미지, 비디오, 오디오, 및 다른 형태의 미디어 및 디지털 객체 (object) 를 조직화하고 표시한다. 보충적인 방식으로, 이러한 미디어를 잠재적으로 시청하는 (및/또는 청취하는) 충분한 그래픽 성능을 갖는 많은 상이한 클라이언트 장치가 존재한다. 예를 들면, 이러한 클라이언트 장치의 범위는 웹 브라우저 소프트웨어를 실행시키는 데스크톱 또는 랩탑 컴퓨터에서부터 휴대용 장치 (예를 들면, Palm 또는 Windows CE 하에서 실행하는 개인 휴대 정보 단말기) 에 이르며, 이 모든 장치가 TCP/IP 를 통하여 인터넷에 접속하며 각각의 장치는 정보를 디스플레이하는 성능을 가진다.

더욱 최근에는, 디스플레이 성능 뿐만 아니라 무선 데이터 성능을 가진 더욱 새로운 분야의 장치가 도입되어 왔다. 통상적으로, 이들 장치는 WAP (무선 애플리케이션 프로토콜) 을 통하여 인터넷에 접속한다. WAP 는 TCP/IP 와 다르지 않은 통신 프로토콜이며, 무선 네트워크를 통하여 데이터를 전송하기 위하여 Motorola, Ericsson 및 Nokia 를 포함하는 무선 회사들의 컨소시엄에 의해 개발되었다. WAP 의 설명에 대해서는, 예를 들면, Mann, S., The Wireless Application Protocol (Dr. Dobb's Journal, pp. 56-66, October 1999) 를 참조한다.

현재, 이들을 전부 합하면, 무선접속 (예를 들면, 9600 baud) 및 유선접속 (예를 들면, 56 kbaud, DSL 및 케이블 모뎀) 을 통하여 인터넷에 접속될 수 있는 수많은 그래픽 장착 장치가 존재한다. 이들 장치는 디지털 이미지를 포함하여 그래픽을 디스플레이할 수 있다. 휴대용 연산 및 무선 기술에서의 최근의 발전에 의해, 제조자들이 팜형 오거나이저 (palm-type organizers), 무선폰, 및 2 웨이 페이저를 포함하는 장치들의 넓은 어레이에 웹 브라우저를 내장시키거나 포함시킬 수 있다. 이들 모든 웹-인에이블 클라이언트는 텍스트의 최소한의 기본 디스플레이를 할 수 있으면, 또한, 몇몇은 이미지, 비디오 및 오디오와 같은 여러 멀티미디어 컨텐츠를 지원한다. 더욱 고속의 무선 인터넷 액세스에 의해 더욱 대형이고 더욱 풍부한 미디어 포맷을 플레이할 수 있는 클라이언트 장치를 개발할 수 있다.

그러나, 이들 많은 장치에 대하여 이미지나 또다른 디지털 미디어를 디스플레이하는 현재의 접근 방법에는 기본적인 문제가 존재한다. 개인용 또는 랩탑 컴퓨터와 같은 몇몇 장치들은 디지털 이미지, 스트리밍 (streaming) 오디오, 및 스트리밍 비디오를 포함하는 많은 상이한 형태의 미디어를 수신하고 렌더링할 수 있다. 그러나, 그 외의 장치들은 디지털 미디어를 렌더링하는데 매우 많은 제한된 성능을 가진다. 또한, 상이한 장치들은 상이한 포맷과 통신 전송 프로토콜을 종종 지원한다.

예를 들면, 팜 (Palm) 휴대용 장치의 예를 고려할 수 있다. 사용자가 웹 상에서 가족의 "트루 컬러" (예를 들면, 32-bit 농도) 1024 x 768 디지털 사진을 갖는다고 가정한다. 사용자가 팜 장치를 이용하여 인터넷에 접속하는 경우, 팜장치가 4-레벨 또는 16-레벨 그레이스케일만을 지원할 수 있기 때문에 사용자는 사진을 디스플레이할 수 없다. 이미지가 디스플레이될 수 있는 경우에도,그 이미지를 팜장치로 다운로드하기 위한 전송시간이 비실용적이다. 또한, 이미지를 다운로드할 수 있는 경우에도, 팜에 대한 디스플레이가 실제로 너무 작기 때문에, 사용자에게 허용가능한 방식으로 이미지를 렌더링할 수 없다.

이러한 문제는 단지 이미지 데이터에만 제한된 것이 아니라, 다른 형태의 디지털 객체에도 적용되는 것이다. 많은 인터넷 사이트는, 컬러링된 이미지, 스트리밍 비디오, 스트리밍 오디오, 및 데스크톱과 랩탑 컴퓨터 장치와 웹 브라우저 소프트웨어를 가진 사용자에게 대상이 되는 또 다른 컨텐츠를 디스플레이한다. 데스크톱 또는 랩탑 컴퓨터는 상당한 처리, 저장, 및 디스플레이 리소스를 가지며, 여러 컬러와 포맷으로 대형 이미지와 비디오 파일을 렌더링할 수 있다. 반면, 셀룰러 전화 또는 개인 휴대 정보 단말기 ("PDA") 와 같은 보다 소형의 장치들은 특별한 포맷으로 미디어를 처리하거나 컬러를 디스플레이하는데 필요한 성능을 통상 갖고 있지 않다. 예를 들면, 통상적으로 메시지 디스플레이를 위한 소형 스크린을 가진 셀룰러 전화는 JPEG 이미지로 렌더링하는 소프트웨어나 디스플레이 성능을 가지지 않는다. 통상적으로, PDA 또는 셀룰러 전화를 가진 사용자는 많은 인터넷 사이트 상에서 이용가능한 많은 정보를 액세스할 수 없다.

셀룰러 전화와 PDA 사용자에 주로 초점을 맞춘 일부 컨텐츠 제공자들은 이들 사용자에 대하여 적절한 미디어를 디스플레이하는 인터넷 사이트를 제공한다. 이들 사이트는 이러한 형태들의 장치에 의해 지원되는 포맷으로 하위의 해상도와 소수의 컬러를 가진 이미지를 포함한다. 그러나, 컨텐츠 제공자가 셀룰러 전화와 휴대용 장치 사용자에 초점을 맞춘 경우에도, 컨텐츠 제공자는 통상적으로 사용하고 있는 매우 많은 장치를 지원하는 문제를 갖는다. 많은 상이한 형태의 셀룰러 전화와 휴대용 장치가 사용되며, 그 각각은 상이한 사양과 성능을 갖고 있다. 이들 장치는 상이한 스크린 크기, 해상도, 및 컬러 성능을 가진다. 따라서, 컨텐츠 제공자가 셀룰러 전화 사용자에 초점을 맞춘 경우에도, 종종, 컨텐츠 제공자는 가장 많은 수의 사용자를 지원하기 위하여 가장 작은 성능의 장치에 대해 조정되는 이미지 및 또 다른 미디어 제공물을 형성해야만 한다.

무선 사용자에 대한 미디어의 전송에서 발생하는 또 다른 특정 문제는 제한된 대역폭이다. 많은 무선 장치의 이미지 디스플레이와 오디오 출력 제한 외에, 이용가능한 대역폭도 더욱 풍부한 미디어 포맷을 액세스하고 이용하는 이들 장치의 성능을 제한한다. 따라서, 보다 우수한 오디오와 비디오 성능을 가진 장치의 경우에도, 이들 장치에 제공될 수 있는 미디어 타입을 상당히 구속할 수 있다. 사용자의 무선 장치가 고해상도 이미지를 디스플레이 할 수 있는 경우에도, 주어진 제한된 이용가능한 대역폭으로 이미지를 다운로드하는데 상당한 시간을 필요로 하기 때문에 사용자는 낮은 해상도 이미지를 요청할 수도 있다.

많은 무선 장치에 이용가능한 제한된 대역폭과 디스플레이 리소스로부터 발생되는 이들 문제 외에도, 현재 인터넷 서비스는 특정한 접속 장치의 성능들을 이해하려고 시도하지 않는다는 문제를 갖는다. 또한, 이러한 인터넷 서비스가 특정 클라이언트 장치의 성능을 이해하고 있는 경우에도, 현재의 서버들은 그 정보들을 실행하고 이러한 클라이언트 장치에 대하여 의미있는 포맷으로만 정보를 전송하도록 설계되어 있지 않다. 네트워크 접속된 장치의 분류가 많을수록, 각각의 장치는 자체 특정한 특성을 갖기 때문에 이러한 문제가 커짐을 예상할 수 있다.

선행 클라이언트 검출 성능들을 재포맷하는 온-더-플라이 (on-the-fly) 미디어를 결합하여 제공자의 미디어의 가능한 적절하고 가장 우수한 구현을 모든 접속된 클라이언트 장치에 전송할 수 있는 솔루션이 필요하다.

용어설명

다음 정의들은 후술할 설명의 이해를 돕기 위하여, 제한이 아닌 설명을 위한 것으로 제공된다.

아파치 (Apache) : 아파치는 Apache Group 이라는 지원 프로그래머들의 그룹에 의해 개발된 퍼블릭 도메인 웹 서버이다. 아파치 서버의 초기 버전은 National Center for Supercomputing Applications (University of Illinois, Urbana-Champaign) 에서 개발된 httpd 웹서버에 기초하여 1995 년에 개발되었다. 아파치 웹 서버와 이 웹서버에 대한 소스 코드의 복사본에 대한 추가 정보는 http://www. apache.org 에서 인터넷을 통하여 통상적으로 이용가능하다.

CC/PP : CC/PP 는 복합 성능/선호도 프로파일의 약어로서, 제안된 표준은 월드 와이드 웹 컨소시엄 (World Wide Web Consortium; W3C) 에 의해 개발되었다. CC/PP 프로파일은 이 장치에 나타낼 수 있는 컨텐츠의 적응을 안내하는데 이용될 수 있는 장치의 성능과 사용자의 선호도의 설명이다. 현재 제안되는 사양은 World Wide Web Consortium (W3C) 으로부터 이용가능한 "Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation" (W3C Note, 27 July 1999) 에 의해 설명된다. 제안된 표준의 복사본은 http://www.w3.org/TR/NOTE-CCPP/ 에서 인터넷을 통하여 통상적으로 찾을 수 있다.

HTML : HTML 는 하이퍼텍스트 생성언어를 나타낸다. 모든 HTML 문서는 웹브라우저에 의해 정확하게 구현되기 위하여 어떤 표준 HTML 태그들 (tags) 를 필요로 한다. 각각의 문서는 헤드와 보디 텍스트로 이루어진다. 헤드는 타이틀을 포함하며, 보디는 단락 (paragraph), 리스트 및 그 외의 요소들로 이루어진 실제 텍스트를 포함한다. 브라우저는 HTML 및 SGML 사양들에 따라 프로그래밍되기 때문에 고유 정보를 예상한다. HTML 문서의 추가 설명은 기술 및 트레이드 문헌 (예를 들면, Ray Duncan, Power Programming: An HTML Primer, PC Magazine, 13 June 1995 참조) 에서 이용가능하다.

HTTP : HTTP 는 하이퍼텍스트 전송 프로토콜을 나타내는데, 이것은 인터넷 상의 월드 와이드 웹에 의해 이용되는 하위 통신 프로토콜이다. HTTP 는 메시지를 포맷하고 전송하는 방법 및 어떤 실행 웹 서버와 브라우저가 여러 커맨드에 응답하여 수행되는지를 규정한다. 예를 들면, 사용자가 브라우저에서 URL 에 진입하는 경우, HTTP 는 HTTP 커맨드를 전송하는 웹서버에 HTTP 커맨드를 실제로 전송하여 요청된 웹페이지를 페치하고 전송한다. HTTP 의 추가 설명은 RFC 2616 : HypertextTransfer Protocol--HTTP/1.1 에서 이용가능하다. RFC 2616 는 World Wide Web Consortium (W3C) 으로부터 이용가능하며 http://www. w3.org/Protocols/ 에서 인터넷을 통하여 통상적으로 이용가능하다. HTTP 의 추가 설명은 기술 및 트레이드 문헌 (예를 들면, William Stallings, The Backbone of the Web, BYTE, October 1996 참조) 에서 이용가능하다.

JPEG : 풀사이즈 디지털 이미지는 JPEG (Joint Photographic Experts Group) 또는 JPEG 포맷으로 유지된다. 예를 들면, Nelson, M. 등의 The Data Compression Book, Second Edition, Chapter 11: Lossy Graphics Compression (particularly at pp. 326-330), M & T Books, 1996 를 참조한다. 또한, 예를 들면, JPEG-like Image Compression (Parts 1 and 2), Dr. Dobb's Journal, July 1995 및 August 1995 을 각각 참조한다 (캘리포니아 산 마테오의 Dr. Dobb's/CD Release 6 from Dr. Dobb's Journal 에서 CD ROM으로 이용가능함).

SMTP : SMTP 는 서버들 간에 e-메일 메시지들을 전송하는 프로토콜인 단순 메일 전송 프로토콜을 나타낸다. 인터넷을 통하여 e-메일을 전송하는 대부분의 e-메일 시스템은 SMTP 을 이용하여 한 서버로부터 다른 서버로 메시지를 전송하는데; 이후 이 메시지는 POP 또는 IMAP 를 이용하는 메일 클라이언트에 의해 검색될 수 있다. 또한, 일반적으로 SMTP 가 이용되어 메일 클라이언트로부터 메일 서버로 메시지를 전송한다.

TCP : TCP 는 전송 제어 프로토콜을 나타낸다. TCP 는 TCP/IP 네트워크에서의 메인 프로토콜들 중 한 프로토콜이다. IP 프로토콜이 패킷들만을 처리하는 반면, TCP 는 두 개의 호스트들로 하여금 데이터의 스트림들의 접속 및 교환을 확립시키게 한다. TCP 는 데이터의 전송을 보장하며, 또한 이들이 전송되었던 순서와 동일한 순서로 패킷이 전송되는 것을 보장한다. TCP 에 대한 도입에 대해서는, 예를 들면, RFC 793 를 참조한다.

TCP/IP : TCP/IP 는 전송 제어 프로토콜/인터넷 프로토콜을 나타내며, 통신 프로토콜들의 슈트는 인터넷 상에서 호스트들을 연결하는데 이용된다. TCP/IP 는 수개의 프로토콜을 이용하는데, 이들 중 2개의 메인 프로토콜이 TCP 및 IP 이다. TCP/IP 는 UNIX 운영 체제으로 구축되며 인터넷에 의해 이용되며 이를, 네트워크를 통하여 데이터를 전송하는 사실 표준 (de facto standard) 으로 만든다. TCP/IP 에 대한 도입에 대해서는, 예를 들면, RFC 1180: A TCP/IP Tutorial 를 참조한다. RFC 1180 의 복사본은 ftp://ftp.isi.edu/in-notes/rfcll8O.txt 에서 통상적으로 이용가능하다.

UAProf : UAProf 또는 WAG UAProf 는 셀룰러폰과 같은 장치들이 그들의 성능을 서버에 설명하는 방법에 대한 제안된 무선 액세스 그룹 사용자 에어전트 프로파일 사양 (*Wireless Access Group User Agent Profile Specification*) 을 나타낸다. 현재 제안된 사양은 WAP Forum 으로부터 이용가능한 "WAG UAProf" (Wireless Application Group User Agent Profile Specification; Wireless Application Protocol Forum, Ltd., Version 30-May-2001 으로 제안됨) 으로서 설명된다. 이 사양의 복사본은 http://wwwl.wapforum.org/techldocumehts/WAP-248-UAProf-20010530-p.pdf 에서 인터넷을 통하여 통상적으로 찾을 수 있다.

URL : Uniform Resource Locator 의 약어로서, 월드 와이드 웹 상의 문서와 또 다른 리소스의 글로벌 어드레스이다. 이 어드레스의 제 1 부분은 어떤 프로토콜이 이용되는지를 나타내며, 제 2 부분은 리소스가 위치되는 IP 어드레스 또는 도메인 명칭을 구체화한다.

WAP : WAP 는 무선 애플리케이션 프로토콜을 나타내며, 이는 TCP/IP와 같은 통신 프로토콜로서, 무선 네트워크를 통하여 데이터를 전송하기 위하여 Motorola, Ericsson, 및 Nokia 를 포함하는 무선회사의 컨소시엄에 의해 개발되었다. WAP 의 설명에 대해서는, 예를 들면 Mann, S., The Wireless Application Protocol, Dr. Dobb's Journal, pp. 56-66, October 1999 를 참조한다. 더욱 자세하게는, WAP 는 현존하는 인터넷 프로토콜의 여러 균등물을 포함한다. 예를 들면, WML 는 HTML 프로토콜의 WAP 버전이다. 또 다른 샘플들은 HTML 브라우저의 WAP 균등물인 WAP 브라우저와, HTTP 서버의 WAP 균등물인 (서버측 상의) WAP 게이트웨이를 포함한다. WAP 게이트웨이에서는, HTTP 가 WAP 로부터/으로 전송된다.

XML : 확장성 생성 언어 (Extensible Markup Language) 의 약어로서 그 사양은 W3C 에 의해 개발되었다. XML 은 웹 문서에 대하여 특별히 설계된 SGML 의 삭감 버전이다. 이에 의해 설계자들은 그들 자신의 주문형 태그들을 생성할 수 있으며 애플리케이션들 간에 및 구성들 (organization) 간에 데이터의 규정, 전송, 검증 및 해석을 할 수 있다. XML 의 추가

설명에 대해서는, 예를 들면, World Wide Web Consortium (www.w3.org) 로부터 이용가능한 확장성 생성언어 (XML) 1.0 사양을 참조한다. 또한, 이 사양은 http://www.w3.org/TR/REC-xml 에서 인터넷을 통하여 통상적으로 이용가능하다.

발명의 개요

본 발명은 컨텐츠 제공자들이 클라이언트의 최광의 어레이를 효과적으로 제공할 수 있는 진보된 방법을 포함하는 온라인 시스템을 제공한다. 본 발명은 선행 클라이언트 검출 성능들을 재포맷하는 온-더-플라이 미디어를 결합하여 제공자의 미디어의 가능한 적절하고 가장 우수한 구현을 모든 접속된 클라이언트 장치에 전송한다.

본 발명의 (eSwitch^{TM}(상품명) 미디어 전송 시스템에 상업적으로 내장된) 온라인 미디어 전송 시스템은 클라이언트 장치로부터 미디어 문서 또는 객체에 대한 요청을 수신하고, 요청을 형성하는 장치의 미디어 출력 성능들을 결정하며 요청한 클라이언트 장치에 대하여 적절한 미디어 객체의 변환 버전을 제공한다. 이 시스템은 클라이언트 성능 모듈 (CCM) 과 미디어 변환 모듈 (MTM) 을 포함한다. 이 두 개의 모듈은 협력하여 HTTP 요청로부터 클라이언트를 식별하며, 그 미디어 출력 성능을 결정하고 이들 성능에 따라 소스 미디어를 재포맷한다. 또한, 이 시스템은 여러 클라이언트 장치의 성능에 대한 정보를 포함하는 데이터 저장 장치, 변환된 미디어 컨텐츠를 저장하는 (선택적) 프론트 사이드 캐쉬 및 컨텐츠의 오리지널 아이템들의 국부 저장을 위한 백사이드 캐쉬를 포함한다.

이 시스템은 복수의 상이한 클라이언트 장치에 대한 미디어 컨텐츠로의 액세스를 제공, 즉, 하드웨어와 소프트웨어 성능을 변경하는 목표 장치로의 액세스를 제공한다. 이 시스템은 접속 가능성을 가진 실제적인 어떠한 장치로 적절한 미디어 컨텐츠의 전송을 가능하게 한다. 목표 장치는 유선 장치 (예를 들면, 데스크톱 컴퓨터, 랩탑 컴퓨터 및 비디오폰) 뿐만 아니라 무선 장치 (예를들면, 셀룰러 전화, 개인 휴대 정보 단말기 (PDA) 및 페이저) 를 포함할 수 있다.

특정 장치에 대하여 적절한 미디어 컨텐츠를 제공하는 시스템의 개선된 방법은 다음과 같이 요약할 수 있다. 먼저, 인터넷 웹 사이트를 통하여 특정 풀-포맷 멀티미디어 객체의 URL 들이 본 발명의 시스템에 의해 제공되도록 변환된다. 이는 웹사이트 상의 HTML 페이지를 변형하여 이들 풀-포맷 멀티미디어 객체의 URL들을, 상부에 미디어 전송 시스템이 설치되어 있고 미디어 객체에 대한 경로를 포함하는 URL 로 교체함으로써 완성될 수 있다. 이 시스템은 오리지널 객체에 대하여 HTTP 프록시로서 기능하며 오리지널 컨텐츠에 대한 요청을 차단하고 요청한 클라이언트에 대하여 적용가능한 컨텐츠의 변환 버전을 제공한다.

클라이언트 장치가, 특정 멀티미디어 문서를 요청하기 위하여 변형된 URL을 인보크하는 사용자 입력 (예를 들면, 클릭) 을 수신하는 경우, 미디어 출력 성능들이 장치에 의해 시스템으로 전송되거나 시스템의 클라이언트 성능 모듈에 의해 추정된다. 장치가 그 성능을 전송하는 경우, 개시 동안 또는 매 대화 동안에 이를 수행한다. 다른 방법으로, 장치의 성능은 장치의 이전 지식에 기초하여 시스템의 데이터 저장 장치에 미리 저장될 수 있다. 시스템의 의해 추정되거나 시스템으로 전송되는 정보에 기초하여, 목표장치의 성능을 설명하는 정보기록이 생성된다. 이 정보는 (만약 있다면) 어떤 변환이 미디어 컨텐츠의 오리지널 아이템을 목표 장치에 적합한 포맷으로 변환하는데 필요한지를 시스템에 지시한다.

장치에 의해 요청되는 적절한 미디어 포맷이 결정된 후, 클라이언트 성능 모듈은 이 특정 장치에 대하여 적절한 포맷으로 변환되어졌던 객체의 버전을 캐쉬가 이미 저장했는지의 여부를 알 수 있도록 프론트 사이드 캐쉬를 (선택적으로) 검사한다. (목표장치에 대하여 변환되는) 객체가 프론트 사이드 캐쉬에 이미 존재하는 경우, 클라이언트 성능 모듈이 그 객체를 클라이언트 장치로 간단하기 리턴시킬 수 있다. 그러나, 적절하게 변환된 객체가 캐쉬에 존재하지 않는 경우, 클라이언트 성능 모듈은 객체 식별자와 클라이언트 장치 파라미터를 미디어 변환 모듈로 반송하는 것을 진행한다.

미디어 변환 모듈은 미디어 컨텐츠의 특정 아이템에 대한 요청을 클라이언트 성능 모듈로부터 수신한다. 미디어 변환 모듈은 오리지널 미디어 객체를 획득하고 오리지널 포맷으로부터의 그 객체를 (특정한 목표 장치 성능에 기초하여) 목표 장치에 대하여 원하는 포맷으로 변환하며 그 변환된 객체를 클라이언트 장치로 리턴한다. 미디어 변환 모듈은 최적화를 위하여 백사이드 캐쉬를 이용하여 증가된 효율을 제공한다. 미디어 객체는 백사이드 캐쉬에 유지되어 각각의 요청에 응답하여 자주 또는 최근에 요청된 아이템을 검색해야 하는 것을 방지한다. 이러한 백사이드 캐쉬의 이용은 네트워크를 통한 호출의 횟수를 감소시키며 클라이언트 장치에 대한 미디어 객체의 변환과 리턴을 신속하게 처리한다.

도면의 간단한 설명

도 1 은 내부에 본 발명의 소프트웨어 구현 프로세서가 내장될 수 있는 컴퓨터 시스템의 블록도이다.

도 2 는 컴퓨터 시스템의 동작을 제어하는 소프트웨어 시스템의 블록도이다.

도 3 은 본 발명의 온라인 미디어 전송 시스템을 나타내는 블록도이다.

도 4a 및 4b 는 목표 장치의 성능을 결정하고 이러한 목표장치에 적절한 포맷으로 컨텐츠를 전송하는 시스템의 상세한 방법 단계들을 설명하는 단일 흐름도를 포함한다.

도 5 는 비순응 (non-compliant) 장치로부터의 HTTP 요청을 입력하는 프록시로서 본 발명의 클라이언트 성능 모듈의 동작을 나타내는 흐름도이다.

바람직한 실시형태의 상세한 설명

다음 설명은 본 발명의 바람직한 실시형태에 초점을 맞춘 것으로, 이 실시형태는 IBM-호환가능 PC 상에서 실행하는 마이크로소프트 윈도우 운영 체제와 같은 데스크톱 운영 체제 하에서 실행하는 인터넷 접속환경에서 동작하는 데스크톱 애플

리케이션에서 실시된다. 그러나, 본 발명은 어떤 특정 애플리케이션이나 어떤 특정한 환경으로 제한되지 않는다. 그 대신에, 본 발명의 시스템과 방법은 Macintosh, Linux, BeOS, Solaris, UNIX, NextStep, FreeBSD 등을 포함하는 여러 다른 플랫폼 상에 바람직하게 내장될 수도 있다. 따라서, 다음에 오는 예시적인 실시형태의 설명은 설명을 목적으로 한 것이지 제한을 목적으로 한 것이 아니다.

I. 컴퓨터에 기초한 구현

A. (예를 들면, 데스크톱 및 서버 컴퓨터에 대한) 기본 시스템 하드웨어

본 발명은 IBM-호환가능 개인용 컴퓨터 (PC) 또는 서버 컴퓨터와 같은 상용 또는 범용 컴퓨터 시스템 상에서 실시될 수 있다. 도 1 은 IBM-호환가능 시스템 (100) 의 매우 일반적인 블록도이다. 도시된 바와 같이, 시스템 (100) 은 랜덤 액세스 메모리 (RAM; 102), 판독 전용 메모리 (ROM; 103), 키보드 (106), 프린터 (107), 포인팅 장치 (108), 디스플레이 장치 (105) 에 연결된 디스플레이 또는 비디오 어댑터 (104), 이동식 (대용량) 저장장치 (115; 예를 들면, 플로피 디스크, CD-ROM, CD-R, CD-RW, DVD 등), 고정식 (대용량) 저장장치 (116) (예를 들면, 하드디스크), 통신 (COMM) 포트(들) 또는 인터페이스(들) (110), 모뎀 (112) 및 네트워크 인터페이스 카드 (NIC) 또는 컨트롤러 (111) (예를 들면, 이더넷) 에 연결된 중앙처리장치(들) (CPU) 또는 프로세서(들) (101) 를 포함한다. 별도로 도시되지 않았지만, 실시간 시스템 클록이 통상적인 방식으로 시스템 (100) 과 함께 포함되어 있다.

CPU (101) 는 마이크로프로세서의 IntelPentium 계열의 프로세서를 포함한다. 그러나, 어떤 적절한 프로세서도 본 발명을 실시하는데 이용될 수 있다. CPU (101) 는 (어떤 필요 입력/출력(I/O) 컨트롤러 회로 및 다른 "글루 (glue)"로직을 포함하는) 양방향 시스템 버스를 통하여 시스템의 다른 구성요소들과 통신한다. 시스템 메모리를 어드레스화하는 어드레스 라인을 포함하는 버스는 여러 구성요소들 간에 및 중에 데이터 전송을 제공한다. 펜티엄급 마이크로프로세서, 이들의 지시 세트, 버스 아키텍쳐 및 제어 라인의 설명은 캘리포니아 산타 클라라의 Intel Corporation 으로부터 이용가능하다. 랜덤 액세스 메모리 (102) 는 CPU (101) 에 대한 작업 메모리로서 기능한다. 통상적인 구성에서, 64 Mbytes 이상의 RAM 이 이용될 수 있다. 본 발명의 범위에 벗어남이 없이 다소의 메모리가 이용될 수 있다. 리드-온리 메모리 (ROM; 103) 는 기본 입력/출력 시스템 코드 (BIOS; 애플리케이션 프로그램과 운영 체제가, 키보드로부터의 문자를 판독하고 프린트로 그 문자를 출력하는 것 등을 포함하여, 하드웨어와 상호작용하는데 이용할 수 있는 ROM 에서의 하위레벨 루틴들의 세트) 을 포함한다.

대용량 저장 장치 (115, 116) 는 마그네틱, 광학 또는 마그네틱-광학 저장 시스템, 플래시 메모리 또는 다른 이용가능한 대용량 저장 테크놀로지와 같은 고정식 또는 이동식 미디어 상에 상주하는 저장 장치를 제공한다. 이 대용량 저장 장치는 네트워크 상에서 공유될 수 있으며, 전용 대용량 저장 장치일 수도 있다. 도 1 에 나타낸 바와 같이, 고정식 저장 장치 (116) 는 운영 체제, 유저 애플리케이션 프로그램, 드라이버 및 다른 지원 파일 뿐만 아니라 그 외의 모든 정렬의 데이터 파일을 포함하는, 컴퓨터 시스템의 동작을 지시하는 프로그램과 데이터의 보디를 저장한다. 통상적으로, 고정식 저장 장치 (116) 는 시스템에 대하여 메인 하드디스크로서 기능한다.

기본 동작에서, (후술할 본 발명의 방법을 실시하는 것을 포함한) 프로그램 로직은 CPU (101) 에 의한 실행에 의해 이동식 저장 장치 (115) 또는 고정식 저장 장치 (116) 로부터 메인 (RAM) 메모리 (102) 로 로딩된다. 프로그램 로직의 동작 동안 시스템 (100) 은 키보드 (106) 와 포인팅 장치 (108) 로부터의 사용자 입력 뿐만 아니라 음성 인식 시스템 (도시되지 않음) 으로부터의 스피치 기초 입력을 받아들인다. 키보드 (106) 는 애플리케이션 프로그램의 선택, 키보드 기초 입력 또는 데이터의 엔트리, 및 스크린 또는 디스플레이 장치 (105) 상에 표시되는 별도의 데이터 객체의 선택 및 조작을 허용한다. 이와 유사하게, 마우스, 트랙 볼, 펜 장치 등과 같은 포인팅 장치 (108) 는 디스플레이 스크린 상에 객체의 선택 및 조작을 허용한다. 이러한 방법으로, 이들 입력장치는 시스템 상에서 실행하는 어떠한 프로세스에 대한 수동의 사용자 입력을 지원한다.

컴퓨터 시스템 (100) 은 텍스트 및/또는 그래픽 이미지 및 다른 데이터를 디스플레이 장치 (105) 상에 표시한다. 디스플레이 장치 (105) 와 시스템들 버스 간에 개재되는 비디오 어댑터 (104) 는 디스플레이 장치 (105) 를 구동시킨다. CPU (101) 에 액세스가능한 비디오 메모리를 포함하는 비디오 어댑터 (104) 는 비디오 메모리에 저장된 픽셀 데이터를, 음극선관 (CRT) 래스터 또는 액정표시장치 (LCD) 모니터에 이용하기에 적절한 래스터 신호로 변환하는 회로를 제공한다. 디스플레이될 정보의 하드카피는 또는 시스템 (100) 내의 다른 정보는 프린터 (107) 또는 다른 출력장치들로부터 얻어질 수 있다. 프린터 (107) 는 시스템 출력의 하드카피 이미지를 생성하는, 예를 들면, (캘리포니아 팔로 알토의 Hewlett-Packard 로부터 이용가능한) HP LaserJet(R) 프린터를 포함할 수 있다.

시스템 자체는 네트워크 (예를 들면, 이더넷 네트워크, 블루투스 무선 네트워크 등) 에 접속되는 네트워크 인터페이스 카드 (NIC; 111) 및/또는 캘리포니아 산타 클라라의 3Com 으로부터 이용가능한 예들인 모뎀 (112; 예를 들면, 56 Kbaud, ISDN, DSL, 또는 케이블 모뎀) 를 통하여 다른 장치 (예를 들면, 다른 컴퓨터들) 와 통신한다. 또한, 시스템 (100) 은 통신 (COMM) 인터페이스 (110) 를 통하여 때때로 국부적으로 접속되는 장치 (예를들면, 직렬 케이블 연결된 장치) 와 통신할 수 있는데, 이 인터페이스는 RS-232 시리얼 포트, 유니버설 시리얼 버스 (USB) 인터페이스 등을 포함할 수 있다. 인터페이스 (110) 에 국부적으로 공통 접속되는 장치는 랩탑 컴퓨터, 휴대용 오거나이저, 디지털 카메라 등을 포함할 수 있다.

IBM-호환가능 개인용 컴퓨터와 서버 컴퓨터는 여러 판매사들로부터 이용가능하다. 대표적인 판매사는 텍사스 라운드 락의 Dell Computers, 텍사스 휴스턴의 Compaq Computers, 및 뉴욕 아몽크의 IBM 을 포함한다. 다른 적절한 컴퓨터들은 캘리포니아 쿠퍼티노의 Apple Computer 로부터 이용가능한 Apple-호환가능 컴퓨터 (예를 들면, Macintosh) 및 캘리포니아 마운틴 뷰의 Sun Microsystems 로부터 이용가능한 Sun Solaris workstations 을 포함한다.

B. 기본 시스템 소프트웨어

도 2 에 나타낸 바와 같이, 컴퓨터 시스템 (100) 의 동작을 지시하기 위하여 컴퓨터 소프트웨어 시스템 (200) 이 제공된다. 시스템 메모리 (RAM; 102) 에 및 고정식 저장 장치 (예를들면, 하드 디스크; 116) 에 저장된 소프트웨어 시스템 (200) 은 커널 또는 동작 시스템 (OS; 210) 을 포함한다. OS (210) 는 프로세스의 관리 실행, 메모리 할당, 파일 입력 및 출력 (I/O)

및 장치 I/O 를 포함하는 컴퓨터 동작의 하위 레벨 태양을 관리한다. 클라이언트 애플리케이션 소프트웨어 또는 "프로그램"과 같은 하나 이상의 애플리케이션 프로그램 (201; 예를 들면, 201a, 201b, 201c, 201d) 이 시스템 (100) 에 의한 실행을 위하여 "로딩"될 수 있다 (즉, 고정식 저장 장치 (116) 로부터 메모리 (102) 로 전송될 수 있다).

사용자 명령 및 데이터를 그래픽 (예를 들어, "포인트 앤드 클릭") 형태로 받기 위해, 시스템 (200) 은 그래픽 사용자 인터페이스 (215; GUI) 를 포함한다. 다음으로, 이러한 입력은, OS (210) 및/또는 클라이언트 애플리케이션 모듈(들) (201) 로부터의 지시에 따라 시스템 (100) 에 의해 작동될 수 있다. 또한, GUI (215) 는 운영 체제 (210) 및 애플리케이션(들) (201) 로부터의 동작 결과를 디스플레이하도록 기능하며, 이 때 사용자는 부가적인 입력을 공급하거나 세션을 종료할 수 있다. 통상적으로, OS (210) 는 특히 주변 장치들과 인터페이싱될 때, 장치 드라이버 (220) (예를 들어, "Winsock" 드라이버--TCP/IP 스택의 윈도우 구현) 및 시스템 바이오스 마이크로코드 (230) (즉, ROM-기반 마이크로코드) 와 함께 동작한다. OS (210) 는, 워싱턴 레드몬드의 Microsoft 사의 Microsoft(R) Windows 9x, Microsoft(R) Windows NT, Microsoft(R) Windows 2000, 또는 Microsoft(R) Windows XP 와 같은 종래의 운영 체제에 의해 제공될 수 있다. 다른 방법으로, OS (210) 은 또한 전술한 운영 체제와 같은 다른 운영 체제일 수 있다.

상술한 컴퓨터 하드웨어 및 소프트웨어는 본 발명을 구현하는데 이용될 수 있는 기본적 언더라잉 데스크탑 및 서버 컴퓨터 콤포넌트를 설명하기 위해 제시된 것이다. 논의를 위해, 다음의 설명은 하나 이상의 "클라이언트" (예를 들어 미디어 디스플레이 장치) 와 통신하는 "서버" (예를 들어, 웹 서버) 가 존재한다고 가정될 수 있는 예를 제시할 것이다. 그러나, 본 발명은 어떤 특정 환경 또는 장치 구성에 제한되지 않는다. 특히, 클라이언트/서버 구별은 본 발명에 필수적이지 않으나, 논의를 위한 골격을 제공하는데 이용된다. 대신에, 본 발명은 이하 상세히 제시되는 본 발명의 방법론을 지원할 수 있는 임의의 타입의 시스템 아키텍쳐 또는 처리 환경에서 구현될 수 있다.

Ⅱ. 다양한 장치의 성능에 맞추어진 미디어의 온라인 렌더링

A. 도입

오늘날, 많은 양의 다양한 타입들의 미디어 컨텐츠가 다수의 인터넷 사이트들에서 이용가능하다. 동시에, 사용자들에게 미디어를 디스플레이 (렌더링) 할 수 있는 폭 넓은 범위의 목표 장치들이 존재한다. 이러한 장치들은 개인용 컴퓨터, 랩탑 컴퓨터, 개인용 디지털 어시스턴스 (PDA), 양-방향 페이저, 및 셀룰러 전화를 포함한다. 그러나, 미디어 컨텐츠를 이러한 장치들에 전송하는데 몇몇 문제점들이 존재한다. 사용 중인 다양한 목표 장치들의 매우 다른 성능들 및 인터넷 상에서 이용가능한 서로 다른 타입의 이미지 및 미디어 컨텐츠가 주어진다면, 현재 컨텐츠 제공자는 만족스러운 방식으로 특정 목표 장치의 사용자에게 디스플레이 (또는 렌더링) 에 적합한 방식의 미디어 컨텐츠를 전송하는데 문제가 있다.

이러한 문제점들에 대한 하나의 해결방법은 인터넷 사이트가 서로 다른 많은 포맷으로 정보를 디스플레이하는 것이다. 그러나, 이 해결방법은 시장에 나와있는 다양한 타입의 장치들 및 그것들의 폭넓은 범위의 성능을 어드레싱하기 위하여, 인터넷 컨텐츠 제공자로 하여금 동일한 미디어의 많은 복사본들을 디스플레이하도록 요청한다. 이러한 접근방법의 단점 중의 하나는 그것이 컨텐츠 제공자로 하여금 지원될 장치의 수에 의존하여 다양한 포맷으로 동일한 미디어의 다수의 프리-렌더링된 버전을 생성, 저장, 및 관리하도록 요청하는 것이다.

본 발명은 컨텐츠 제공자가 하나의 형태로 컨텐츠를 개발하고 컨텐츠를 요청하는 클라이언트 장치의 성능에 기초하여 다수의 형태들로 컨텐츠를 전송하도록 한다. 본 발명은 장치의 미디어 출력 성능에 적합한 미디어 컨텐츠를 클라이언트 장치에 제공하는 온라인 시스템 및 방법론을 포함한다. 이 시스템은 연결된 클라이언트 장치들의 성능을 결정하는 클라이언트 성능 모듈 (CCM) 및 적합한 미디어를 신속히 렌더링하여 적합한 포맷으로 클라이언트 장치에 전송하는 미디어 변환 (즉 트랜스코딩) 모듈 (MTM) 을 포함한다.

본 발명의 동작은 디지털 이미지를 특정 클라이언트 장치로 렌더링하는 다음의 예로 설명될 수 있다. 이 예에서, 인터넷 사이트상의 미디어 컨텐츠의 오리지널 아이템은 24-비트 컬러 JPEG 이미지이며 이러한 이미지를 요청하는 클라이언트 장치는 16-레벨 그레이스케일을 지원하는 팜 (Palm) PDA 이다. 이 클라이언트 장치는 무선으로 인터넷 사이트에 연결되어 이러한 JPEG 이미지를 위한 URL을 호출한다. 본 발명을 이용하는 인터넷 컨텐츠 제공자는 이러한 이미지를 위하여 URL을 미리 변경하여 본 발명의 클라이언트 성능 모듈이 설치되는 기계를 언급한다. 그 결과, 이 URL 요청은 CCM 으로 라우팅되고, 이는 요청수신한 이미지 및 이 요청로부터의 클라이언트 장치를 식별한다. CCM 은 클라이언트 장치에 관한 정보를 얻기 위해 서버로의 클라이언트 요청을 조사하거나 이 데이터 저장 장치에 저장된 알려진 장치 특성 및 성능과 이 정보를 비교하여, 지능적으로 클라이언트 장치의 성능을 결정한다. 이 경우에, CCM 은 이 장치를 특정 타입의 팜 PDA 로서 식별하여 시스템의 데이터베이스에서 장치의 성능을 탐색한다. 이 정보에 기초하여, CCM 은 JPEG 이미지가 16-레벨 그레이스케일 포맷으로 이 팜 장치에 제공되어야 하는지 결정한다.

이 장치에 의해 요청되는 적합한 미디어 포맷이 결정된 후에, CCM (선택적으로) 은 이 캐쉬가 이미 이미지의 버전을 요청 수신한 포맷으로 저장하는지 여부를 알기 위해 프론트-사이드 캐쉬를 검사한다. 적합한 변환된 이미지가 캐쉬에 있지 않다면, 그 후, CCM 은 이 타입의 팜 PDA 장치로 렌더링하기에 적합한 16-레벨 그레이스케일 포맷으로 미디어 변환 모듈로부터 이미지를 요청한다. MTM은 적합한 이미지를 획득하여, 그것을 적합한 포맷으로 전환하고, 그것을 클라이언트 장치로 서브한다. 이 시스템은 그것이 이미지들을 (그것의 오리지널 포맷으로부터) 특정 목표 장치용에 적절한 포맷으로 선택적으로 전환하게 하는 인텔리전스를 포함한다. 이 전체적인 전환 또는 변환 처리는 이 시스템에서 원하는 성능 및 범위성 (scalability) 기준을 보존하는 방식으로 수행된다.

B. 미디어 전송 시스템의 개관

1. 기본 아키텍쳐

도 3 은 본 발명을 구현하는데 적합한 온라인 환경 (300) 을 설명한다. 나타낸 바와 같이, 환경 (300) 은 (310 에서 나타난) 인터넷을 통해 하나 이상의 클라이언트 장치 (301) 및 하나 이상의 인터넷 사이트 (330; 서버) 에 연결된 온라인 미디어 전송 시스템 (320) 을 포함한다. 다음으로, 이들 콤포넌트들의 각각을 더욱 상세히 설명한다.

클라이언트 장치 (301) 은 인터넷을 통해 연결되어 온라인 컨텐츠에 액세스할 수 있는 다양한 목표 장치들 ("클라이언트들") 중의 하나를 표시한다. 예를 들어, 클라이언트 장치들은 무선 장치들 (예를 들어, 데스크탑 컴퓨터, 랩탑 컴퓨터, 및 비디오폰) 과 함께 무선 장치들 (예를 들어, 셀룰러 전화, 휴대형 PDA (개인 데이터 어시스턴스), 및 페이저) 를 포함할 수도 있다. 단일 클라이언트 장치가 이 도에서 나타나지만, 통상적으로 환경 (300) 은 다수의 이러한 장치들과 연결되어 동작하곤 한다.

인터넷 서버 (330) 은 미디어 컨텐츠의 아이템들 (예를 들어, 오디오, 비디오, 문서, 블롭 객체, 또는 흥미있는 다른 아이템들) 이 저장되는 웹 서버를 표시한다. 동작하는 동안, 통상적으로 인터넷 서버 (330) 는 폭 넓은 범위의 클라이언트 장치들에 이용 가능한 미디어 컨텐츠의 다른 많은 아이템들을 저장한다. 인터넷 서버 (330) 과 온라인 미디어 전송 시스템 (320) 사이의 실질적 연결이 인터넷 상에서 발생하거나, 또는 선택적으로 이 도에서 점선 연결로 나타낸 비-인터넷 (예를 들어, WAN) 을 통해 발생할 수도 있다. 어느 하나의 경우에, 인터넷 서버 (330) 는 온라인 미디어 전송 시스템 (320) 으로서 동일한 사이트에 수용되거나, 또는 원하는 대로 원격 사이트에 수용될 수 있다.

온라인 미디어 전송 시스템 (320) 은 클라이언트 장치 성능을 검출하는 기능을 하며, 그 결정에 기초하여, 클라이언트 장치 (301) 에 적합한 포맷의 이러한 장치들로 미디어 컨텐츠를 변환하여 전송한다. 나타낸 바와 같이, 미디어 전송 시스템 (320) 은 클라이언트 성능 모듈 (CCM; 322), 미디어 변환 모듈 (MTM; 325), 및 장치 성능 데이터 저장 장치 (324) 를 포함한다. 또한, 나타낸 바와 같이, 클라이언트 성능 모듈 (322) 는 프론트-사이드 (321) 및 CCM 로그 (323) 과 직접 통신하며; 유사하게, 미디어 변환 모듈 (325) 백사이드 캐쉬 (327) 및 MTM 로그 (326) 와 직접 통신한다.

2. 기본 동작

기본 시스템 동작 동안에, 인터넷 서버 (330) 상에서 미디어 컨텐츠 (예를 들어, 웹 페이지들) 를 포함 및/또는 참조하는 아이템들은 이러한 아이템들을 요청하는 클라이언트들을 시스템 (320) 으로 안내하는 URL 로 인코딩된다. 또한, 인터넷 서버 (330) 은 미디어 컨텐츠의 오리지널 아이템들을 포함하며, 이들은 디지털 이미지, 비디오, 오디오, 문서, "블롭 (blob)" 객체 등을 포함하는 임의의 타입의 컨텐츠일 수 도 있다. 다른 방법으로, 미디어 컨텐츠의 오리지널 아이템들은 시스템 (320) 또는 이 시스템 (320) 이 연결되는 다른 국부 또는 원격 서버 상에 국부적으로 저장될 수 있다. 클라이언트 (301) 가 컨텐츠의 아이템을 요청 (예를 들어, HTTP 요청) 할 때, 이 요청은 미디어 전송 시스템 (320) 의 클라이언트 성능 모듈 (322) 로 라우팅된다. 클라이언트 장치 (301) 로부터 수신된 요청에 응답하여, 클라이언트 성능 모듈 (322) 는 (클라이언트) 장치를 식별하여 장치의 성능에 관해 이용가능한 정보를 획득한다. 이러한 식별에 기초하여, 클라이언트 성능 모듈 (322) 는 데이터 저장 장치 (324) 로부터 미디어를 디스플레이하거나 또는 출력하는 클라이언트 장치의 성능에 관한 부가적인 정보를 검색한다.

데이터 저장 장치 (324) 는 다양한 장치들의 미디어 출력 성능을 포함한다. 현재의 바람직한 실시형태에서, 대응 장치 식별자는 이 정보에 인덱싱하는데 이용된다. 데이터 저장 장치 (324) 에 저장된 성능은 스크린 해상도, 스크린 컬러 농도, 이미지가 장치의 스크린 디스플레이에 맞도록 회전되어야 하는지 여부에 관한 정보, 및 이하에서 더욱 상세히 설명할 다른 이러한 정보를 포함한다. 데이터 저장 장치 (324) 는 새로운 장치가 시장에 소개될 때, 이러한 장치들 및 이들의 성능의 프로파일들이 부가될 수 있도록 업데이트될 수 있는 필드이다. 클라이언트 성능 로그 (323) 은 식별될 수 없거나 또는 성능들이 이용가능하지 않는 임의의 클라이언트 장치들의 기록을 포함한다. 이러한 로그 기록은 이러한 장치들에 관한 정보가 획득되어 데이터 저장 장치 (324) 에 부가될 수 있도록, 어느 생략된 장치들이 식별될 수 있게 한다.

클라이언트 장치 (301) 의 성능이 결정된 후에, 클라이언트 성능 모듈 (322) (선택적으로) 은 미리 변환된 컨텐츠를 저장하는 프론트- 사이드 캐쉬 (321) 을 조사하여, 이 객체가 적합한 포맷으로 이용가능한지 여부를 파악한다. 프론트-사이드 캐쉬 (321) 는 장래의 요청에 응답하여 미리 전환된 미디어 객체들이 공급을 위해 유지되는 (선택적인) 최적화이다. 프론트-사이드 캐쉬 (321) 는 최근에 최소 사용된 아이템들을 "에이지 아웃 (age out)" 하는 (즉, 제거하는) 최근 최소 사용 (least-recently used: LRU) 기술을 이용하여 구현될 수 있다. 클라이언트 성능 모듈 (322) 이 적합한 객체가 프론트-사이드 캐쉬 (321) 에서 이용가능하지 않다고 결정한다면, 그것은 미디어 변환 모듈 (325) 로 하여금 이 객체를 제공할 신속한 변환을 수행하도록 요청한다.

미디어 변환 모듈 (MTM; 325) 는 클라이언트 성능 모듈 (322) 로부터 특정 포맷의 미디어 컨텐츠의 특정 아이템에 대한 요청을 수신한다. 이 미디어 변환 모듈 (325) 는 요청된 미디어 객체의 원본을 획득하여, 그것을 요청되는 포맷으로 변환하고, 이 변환된 미디어 객체를 클라이언트 장치 (301) 로 리턴시킨다. 백사이드 캐쉬 (327) 는 증가된 효율을 제공하는 최적화이며; 또한, 그것은 LRU 기술을 이용하여 구현될 수 있다. 인터넷 서버 (330) (또는 다른 소스) 으로부터 검색된 오리지널 객체들 백사이드 캐쉬 (327) 에 유지되어 각각의 요청에 응답하여 각각의 요청되는 아이템의 복사본을 검색해야 하는 것을 피한다. 이러한 백사이드 캐쉬의 사용은 네트워크를 통한 콜의 검색을 감소시킨다. 또한, 원격 서버로부터의 큰 객체들 (고품질 컬러 이미지들과 같은) 의 검색을 피함으로써, 미디어 변환 모듈 (325) 에 의한 이용 가능한 객체들의 전환 및 리턴을 촉진한다.

C. 장치들의 성능을 검출하고 적합한 미디어 객체들을 전송하는 방법론

도 4a, b는 연결된 클라이언트 장치들 및 적합한 방식으로 미디어 객체를 이런 장치들에 전송할 때, 이 시스템의 동작의 상세한 방법 단계들의 단일 흐름도를 포함한다. 단계 401 에서는, 이런 객체들이 본 발명의 미디어 전송 시스템에 의해 서브되도록, 인터넷 사이트에서의 웹 페이지의 멀티미디어 객체들에 대한 URL들이 변경된다. 현재의 바람직한 실시형태에서, 이 URL들은 미디어 전송 시스템이 설치된 서버의 이름이 앞에 달려있다. 예를 들어, 가입자의 사이트가 URL: http://www.subscriber.com/img/logo.gif 에 의해 정상적으로 액세스된 로고를 포함한다면, 변경된 URL 은 http://eswitch.com/www.subscriber.com/img/logo.gif 가 될 것이다.

단계 402 에서는, 이러한 다시 쓰여진 URL을 포함하는 웹 페이지가 열리거나 또는 클라이언트 장치가 다시 쓰여진 URL을 선택할 때 (클릭할 때), 클라이언트 장치로부터의 HTTP 요청이 미디어 전송 시스템으로 라우팅된다. 단계 403 에서, 클라이언트 성능 모듈 (CCM) 은 단계 401에서 수행된 인코딩 처리를 리버싱하며, 소스 이미지로의 완전한 URL을 결정한다. 현재의 바람직한 구현에서, 이것은 요청 URL로부터 "/eswitch.com" 을 제거하는 것으로 구성된다.

단계 404 에서, CCM 은 키로서 HTTP 사용자-에이전트 헤더를 이용하여 데이터 저장 장치로부터 클라이언트 성능 구성을 검색한다. 이 구성 정보는 디스플레이 크기, 컬러 농도, 오디오 채널 등과 같은 클라이언트 장치의 재생 성능을 지정한다. 구성 정보는 클라이언트 장치의 완전한 성능을 결정하기 위하여 부가적인 HTTP 요청 헤더의 조사를 요청할 수 있다. 이 단계 동안에 수집된 정보는 CCM 이 무슨 성능이 목표 장치에서 지원되는지 정확히 알도록 허용한다. 특히, 이 정보는 오리지널 미디어 객체를 목표 장치에 적합한 포맷으로 전환하기 위해 어떤 특정 변환 동작이 요청되는지를 이 시스템에 지시한다.

단계 405에서, CCM 은 미디어 변환 모듈 (MTM) 에 특정한 명령들을 포함하는 URL 을 구성한다. 이러한 명령들은 MTM 이 소스 미디어 문서 또는 객체를 변환하도록 지시하여 이 문서를 요청한 클라이언트 장치의 성능에 합치한다. 이 URL 은 구성 파일에서 지정한 MTM 서버를 지시한다. MTM 모듈은 CCM 보다 동일한 서버 또는 다른 서버 상에 존재할 수 있다. (선택적) 단계 406에서, CCM 은 구성된 MTM URL 에 매칭되는 객체를 위해 프론트-사이드 캐쉬를 참고한다. 즉, 그것은 프론트-사이드 캐쉬가 특정 목표 장치에 적합한 방식으로 전환된 미디어 객체의 버전을 미리 저장하였는지 여부를 조사하여 파악한다.

현재의 바람직한 실시형태에서, 이 시스템 내에서 사용된 URL 스트링들은 인코딩되어 특정 포맷으로 특정 객체에 대한 인덱스로서 기능한다. 이런 방식으로, 인코딩된 URL 스트링은 특정 포맷의 특정 문서가 특정 로케이션에 저장되어 있음을 지시할 수 있다. 예를 들어, CCM 은 특징: 크기 = 100 픽셀, 컬러 농도 = 8, 및 컬러 = 거짓 (false) 를 갖는 logo.gif 의 변환된 버전을 포함하는 URL 을 검사한다. 문서 또는 객체 (목표 장치 용으로 전환된) 이미 프론트-사이드 캐쉬에 존재한다면, CCM 은 단순히 이 문서를 목표 장치로 리턴시킬 수도 있다. 그러나, 매칭 아이템이 이 캐쉬에서 발견되지 않는다면, 그 후, 이 방법은 단계 407 에서 설명한 바와 같이 계속된다.

단계 407 에서, CCM 은 오리지널 클라이언트 요청을 프록시하며, 클라이언트가 보낸 URL 을 CCM 에 의해 생성된 재구성된 URL 로 대체한다. 이 프로세스는 클라이언트에게 완전히 명백하다: 이 요청을 하는 클라이언트는 이 요청이 MTM 으로 전송되었는지를 보고받거나 또는 인식하지 못한다. 오히려, 이 전송은 CCM 이 MTM 에 의한 실행을 위해 클라이언트가 한 요청을 전송하는 백-엔드 (back-end) 프로세스이다. 단계 408 에서, 이 MTM 은 재구성된 URL 을 수신하며, 백사이드-캐쉬 서버를 통해 오리지널 미디어 객체에 대해 HTTP 요청을 한다. 객체가 백사이드 캐쉬에 존재한다면, 로컬 디스크에서 서빙된다. 그렇지 않다면, 캐쉬 서버는 오리지널 URL 에서 식별되는 인터넷 사이트로부터이 객체를 요청하며, 장래 사용을 위해 그것을 캐쉬한다. 이 점에서, MTM 의 작업은 오리지널 포맷으로부터의 미디어 객체를 (목표 장치 성능에 기초하여) 목표 장치를 위해 원하는 포맷으로 변환한다. 단계 409 에서, MTM 은, 수신된 재구성 URL 에서 특정된 대로 미디어 변환을 수행한다. 일단, MTM 이 이 작업을 수행하면, 단계 410 에서는 미디어 객체의 새롭게 변환된 버전이 클라이언트 장치에 리턴되며, (선택적으로) 또한 프론트-사이드 캐쉬로 복사된다.

D. 클라이언트 성능을 결정하고 그에 대한 정보를 제공하는 시스템의 이용

상술한 역할에 기여하는 것 외에, 본 발명은 또한 클라이언트 장치들의 성능을 결정하고 이 클라이언트 성능 정보를 다른 시스템들 또는 장치들에 제공하는데 이용된다. 어떻게 셀룰러 전화와 같은 장치들이 그것들의 성능을 서버들에 설명하는지에 대한 더 자세한 설명이, WAP forum 에서 입수가능한, WAG UAProf (Wireless Application Group User Agent Profile Specification), Wireless Application Protocol Forum, Ltd., Proposed Version 30-May-2001 에 개시되어 있다. 이 설명의 복사본은 현재 인터넷 상의 http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf 에서 발견할 수 있다. 유사한 설명이 Comosite Capability/Preference Profiles(CC/PP): World Wide Web Consortium, (W3C) 으로부터 이용가능한 A user side framework for content negotiation, W3C Note, 27 July 1999 에 의해 설명된다. 이 설명의 복사본은 현재 인터넷 상의 http://www.w3.org/TR/NOTE-CCPP 에서 발견할 수 있다.

이러한 제안된 표준들 중 하나 또는 양자 모두는 장래에 새로운 장치들 및 장치 소프트웨어에 의해 지원될 수 있지만, 이러한 표준들에 대한 현재의 지원은 매우 제한적이다. 이러한 표준들의 장치 지원이 보편적일 때까지, 서버 애플리케이션들은 표준 장치 정보를 이용할 수 없을 것이다. 이러한 문제점은 비-순응 장치로부터의 인커밍 HTTP 요청을 위한 프록시로서 역할을 하도록 본 발명의 시스템을 구성함으로써 해결될 수 있다. 요청이 부분적인 또는 어떠한 UAProf 또는 CC/PP 정보도 갖지 않는 미디어 전송 시스템으로 전송될 때, 클라이언트 성능 모듈은 요청된 데이터를 찾아서, 이 요청을 최후 목적지로 전송하기 전에 이 정보를 요청에 첨부한다. 이는 시스템으로 하여금 UAProf, CC/PP, 또는 다른 동등한 표준들을 요청하는 그러한 인터넷 및 WAP 와 비-순응 클라이언트 장치 사이에서 다리의 역할을 할 수 있게 한다.

도 5 는 비-순응 장치들로부터의 인커밍 HTTP 요청을 위한 프록시로서 역할을 할 때, 미디어 전송 시스템의 클라이언트 성능 모듈의 동작을 나타내는 흐름도이다. 단계 501 에서, 비-클라이언트 장치로부터의 HTTP 요청은 인터넷 또는 WAP 로부터 이 시스템으로 전송된다. 이를 위해, "비-순응" 클라이언트 장치는 UAProf, CC/PP, 또는 이런 장치가 이러한 성능을 식별하도록 요청하는 유사한 표준들에 따르지 않는 장치이다.

단계 502 서, 이 시스템의 클라이언트 성능 모듈 (CCM) 은 키로서 HTTP 사용자-에이전트 헤더를 이용하여 데이터 저장 장치로부터 클라이언트 성능 구성을 검색한다. 구성 정보는 디스플레이 크기, 컬러 농도, 오디오 채널 등과 같은 클라이언트 장치의 성능을 지정한다. 구성 정보는 부가적인 HTTP 요청 헤더들의 조사를 요청하여 클라이언트 장치의 완전한 성능을 결정한다. 이 단계 동안에 수집된 정보는 CCM 어떠한 성능이 목표 장치에서 지원되는지를 정확히 이해하도록 한다.

단계 503 에서, CCM 은 아래에서 설명하는 바와 같이 이러한 클라이언트 장치의 특정 성능에 관한 정보로 특정 클라이언트 장치가 한 요청을 보충한다. 단계 504 서, CCM 은 클라이언트 성능에 관한 상세한 사항을 포함하는 보충된 요청을 이러한 클라이언트 요청에서 지정된 목적지로 리턴시킨다.

다음은 어떻게 이 시스템이 장치 성능 정보를 요청에 첨부하는데 사용될 수 있는지의 예이다. 이 시스템에 전송된 인커밍 요청의 예는 다음과 같다:

```
GET /index.wml HTTP/1.1
Host: www.lightsurf.com
Accept-Charset: ISO-8859-1
Accept-Language: en
x-up-subno: pegli_pegli-nt4.office.lightsurf.com
x-upfax-accepts: none
x-up-uplink: none
x-up-devcap-smartdialing: 1
x-up-devcap-screendepth: 1
x-up-devcap-iscolor: 0
x-up-devcap-immed-alert: 1
x-up-devcap-numsoftkeys: 2
x-up-devcap-screenpixels: 171,108
x-up-devcap-msize: 8,18
User-Agent: UP.Browser/3.1-UPG1 UP.Link/3.2
```

나타난 바와 같이, 인커밍 정보는 예를 들어, 컬러 지원 (상기 장치에 대한 "0" 또는 무 (none)) 및 스크린 픽셀 (상기 장치에 대한 171 x 108 픽셀) 을 포함하는 장치 성능을 보고한다.

상기 요청의 수신에 이어, 클라이언트 성능 모듈은 상술한 방식으로 특정 클라이언트 장치의 성능을 결정한다. 그 후, CCM 은 이 정보를 요청에 첨부하여 보충된 요청을 최후 목적지로 전송한다. CCM 에 의해 첨부된 정보를 나타내는 샘플 요청은 다음과 같다:

```
GET /index.wml HTTP/1.1
Host: www.lightsurf.com
Accept-Charset: ISO-8859-1
Accept-Language: en
User-Agent: UP.Browser/3.1-UPG1 UP.Link/3.2
x-wap-profile: http://www.eswitch.com/profiles/0A3F362B.xml
```

이 경우, "0A3F362B.xml" 은 UAProf 또는 CC/PP 프로파일 정보 중 어느 하나를 포함하는 생성된 문서이다. 이러한 요청에 대한 샘플 UAProf 파일은 다음과 같다.

```
<?xml version="1.0"/>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:prf="http://www.wapform.org/profiles/UAPROF/ccppschema-20010430#">
  <rdf:Description id="WAP-enabled cellular phone">
    <rdf:type
resource="http://www.wapform.org/profiles/UAPROF/ccpschema-
20010430#Hardware
Platform"/>
    <prf:ScreenSize>171x108</prf:ScreenSize>
    <prf:ColorCapable>No</prf:ColorCapable>
    <prf:NumberOfSoftKeys>2</prf:NumberOfSoftKeys>
    <prf:ScreenSizeChar>8x18</prf:ScreenSizeChar>
    ...
  </rdf:Description>
</RDF>
```

이 생성된 문서는 장치의 HTTP 요청 및 데이터 저장 장치로부터의 정보 및 미디어 전송 시스템에 의해 유지관리되는 지식베이스 (knowledgebase) 으로 채워진다. 이 시스템이 클라이언트 장치 특성의 지식베이스을 유지관리하기 때문에, 그것은 HTTP 헤더의 정보를 단순히 트랜스코딩하는 서버보다 완전한 UAProf 또는 CC/PP 를 훨씬 더 생성할 수 있다.

E. CCM 및 MTM 모듈들의 동작의 상세한 방법

1. 클라이언트 성능 모듈

클라이언트 성능 모듈 (CCM) 은 HTTP 요청로부터 클라이언트 장치를 식별한다. CCM 은 이 데이터 저장 장치에 저장된 미디어 출력 성능 정보와 함께 이 요청에서 제공된 정보를 이용하여 특정 클라이언트 장치의 미디어 출력 성능을 결정한다. 이는 CCM으로 하여금 흥미있는 아이템 (예를 들어, 미디어 객체) 을 요청하는 특정 타입의 클라이언트 장치를 위한 최적의 송신 크기 및 재생 포맷을 결정할 수 있게 한다. 예를 들어, HTTP 브라우저는 브라우저 이름 (예를 들어, "Netscape Navigator" 또는 "Microsoft Internet Explorer"), 브라우저 버전, 및 그것이 지원하는 그래픽 타입을 지시할 수도 있다. 이 정보는 매우 제한된 그래픽 지원 (예를 들어, 단지 JPEG만) 을 갖는 셋-톱 박스에서 실행되는 브라우저와 같이, 그래픽 지원이 제한되는 경우에 도움이 된다. 목표 장치가 그것의 성능을 지시할 수 없는 경우에, 그것은 최소한 팜 휴대형 장치, 셋-톱 박스, WAP 브라우저를 가진 폰 등과 같은 그것의 장치 클래스를 지시할 수 있다.

이 장치 클래스에 기초하여, CCM 은 장치 성능 데이터 저장 장치로부터 장치의 성능에 관한 정보를 획득한다. 예를 들어, 이 장치 클래스는 특정 모델의 팜 휴대형 장치일 수 도 있다. 이 정보에 기초하여, CCM 은 이 데이터 저장 장치에서 유지관리되는 지식베이스에서 장치 클래스를 찾음으로써, 디스플레이 성능 (예를 들어, 16 컬러), 장치 메모리 (예를 들어, 4-8 MB), 및 디스플레이 크기 (예를 들어, 300 x 500) 과 같은 장치의 성능을 구별할 수도 있다.

또한, CCM 은 CCM 로그에서 식별되지 않는 클라이언트들을 로그하는 방법 및 지식베이스이 가능한 최신이 되도록 확보하도록 일정 간격으로 통보를 제공하는 방법을 포함한다. 새로운 클라이언트 장치가 도입될 때, 지식베이스의 구성 정보는 이러한 새로운 클라이언트 장치들에 정보를 부가하도록 업데이트될 수 있다. CCM 은 그것의 구성 파일들을 업데이트하기 위해 "푸쉬" 또는 "푸쉬/풀" 방식 (scheme) 중 어느 하나를 지원한다. "푸쉬" 방식은 대체 구성 파일을 포함하는 데이터 스토어로 보내지는 안전한 HTTP 포스트 요청 또는 SMTP 메시지로 구성된다. "푸쉬/풀" 방식은 HTTP 겟 (GET) 요청 또는 서버로 하여금 지역 구성 파일의 업데이트를 스케줄링하도록 하는 SMTP 메시지를 보내는 단계로 구성된다.

CCM 은 주로 HTTP 헤더들, 특히 사용자-에이전트 헤더에 의존하여 클라이언트를 식별한다. 또한, CCM 은 HTTP (예를 들어, 요청의 IP 패킷들의 원본) 아래의 프로토콜 층의 정보를 조사한다. 일단 장치가 식별되면, CCM 은 장치 성능에 대한 디폴트 값을 제공하고, 또한 미디어 콘텐트의 적절한 출력 포맷을 결정하는데 이용될 수 있는 억셉트 (Accept) 헤더와 같은 부가적 헤더들을 특정하는 (데이터 스토어 내의) 계층 구성 파일을 찾는다. 일단, 장치의 성능이 결정되면, CCM 은 적절한 재포맷팅 정보를 포함하는 지정된 미디어 문서에 대한 MTM 으로의 요청을 구성한다. 그 후, CCM 은 재구성된 요청을 갖는 MTM 으로 클라이언트 접속을 전송한다.

현재의 바람직한 실시형태에서, CCM 은 클라이언트가 풀 (full) HTTP 요청에 접속한 아파치 (Apache) 모듈로서 구현될 수 있다. 이 요청에 포함된 정보는 일련의 질의를 통해 구성 파일에 대한 요청을 하는 클라이언트 장치를 식별하는데 이용된다. HTTP 사용자-에이전트 헤더는 요청하는 클라이언트의 주요 지시자 (indicator) 이다. 사용자-에이전트가 HTTP/1.0 및 HTTP/1.1 모두에서 선택적 헤더인 반면, 실제로 모든 웹 클라이언트 소프트웨어는 각각의 요청에 대한 헤더의 몇몇 식별 정보를 보낸다. 또한, 많은 클라이언트는 이들 장치의 물리적 성능을 설명하는 커스텀 헤더를 보낸다. 예를 들어, 모바일 폰에서 사용되는 WAP 브라우저인 UP 브라우저 (캘리포니아 레드우드의 Openwave System, Inc. 에 의해 제공되는 Unwired Planet 브라우저) 는 이 요청의 "비-표준" 헤드를 보낸다. 이 문맥에서 "비-표준"은 이러한 헤더들이 HTTP 설명에 커버되지 않는다는 것을 의미한다. 이러한 헤더의 한 예는 다음과 같다.

```
User-Agent: UP.Browser/3.04-SC02 UP.Link/3.2.3.8
x-up-devcap-charset: US-ASCII
x-up-devcap-immed-alert: 1
x-up-devcap-max-pdu: 1472
x-up-devcap-msize: 8,10
x-up-devcap-numsoftkeys: 2
x-up-devcap-screenchars: 15,5
x-up-devcap-screenpixels: 120,50
x-up-devcap-smartdialing: 1
x-up-devcap-softkeysize: 7
x-up-fax-accepts: none
x-up-fax-limit: 0
x-up-subno: RzOyzSrSj-ARAs01_up2.upl.sprintpcs.com
x-up-uplink: up2.upl.sprintpcs.com
```

상기 헤더의 "x-up-devcap-screenpixels" 부분은 클라이언트 장치가 폭과 길이 (120 x 50) 인 얼마나 많은 픽셀들을 디스플레이할 수 있는지 명확히 지시한다. 상기 예에서 나타난 헤더는 이러한 특정 클라이언트 장치의 성능에 대한 몇몇 상세한 사항을 포함한다. 그러나, 많은 경우에, 헤더들은 이러한 상세한 사항을 포함하지 않으며, 따라서 CCM 은 장치 특성을 획득하기 위해 데이터 스토어에 저장된 정보를 참조해야 한다. 그러나, CCM 은 가능할 때마다 상기 헤더의 "x-up-devcap-screenpixels" 부분과 같은 헤더들의 정보를 이용한다.

현재의 바람직한 실시형태에서, 데이터 스토어의 CCM 구성 파일 (또는 지식베이스) 은 XML 로 다시 쓰여져 언어의 계층 특징을 이용한다. 이 파일은 일련의 <user-agent> 엔트리로 구성된다. 예시적인 <user-agent> 엔트리는 다음과 같이 나타날 수 있다 (라인 번호는 단지 참고임).

```
1    <user-agent header='User-Agent' pattern='UP.Browser' >>
2      <device-class>wap</device-class>
3      <content-type pattern='^image/'>
4        <capability name='color-depth' default='8'/>
5        <capability name='display-height' default='100'>
6          <header name='x-up-devcap-screenpixels'
pattern='(\d+),\d+'/>
7        </capability>
8        <capability name='display-width' default='100'>
9          <header name='x-up-devcap-screenpixels'
pattern='\d+,(\d+)'/>
10       </capability>
11       <capability name='output-format' default='image/bmp'/>
12     </content-type>
13   </user-agent>
```

일반적으로, 패턴을 명명하는 태그 속성은 이 속성들에 제공된 값들이 보통의 표현들로 취급됨을 지시한다. 정보가 보통의 표현들로부터 추출되어야하는 경우, 표준 "match remember" 구문 (삽입구) 이 사용된다. 예를 들어, 상기 라인 9 에서, 패턴 속성은 스트링 "120, 50" 이 매칭되며, 서브스트링 "50" 이 인클로징 (enclosing) 성능 값을 설정하는데 이용될 수 있음을 지시한다.

CCM 이 요청을 받을 때, 그것은 구성 파일을 통해 실행되어, 헤더 속성이 지정한 HTTP 헤더의 값을 값 속성에서 지정된 스트링과 비교함으로써 차례로 각각의 <user-agent> 태그를 검사한다. 다수의 경우에, <user-agent> 태그가 HTTP 사용자-에이전트 헤더에 대해 매칭되며, 헤더 속성이 생략된다면 이 태그를 HTTP 사용자-에이전트 헤더에 대해 매칭시키는 것은 디폴트 행위이다. 각각의 <user-agent> 블록은 그것이 구성 파일에서 나타나는 것처럼 순서대로 처리되어, 더욱 제한적 값을 속성을 갖는 <user-agent> 태그들이 덜 엄격한 값 속성을 갖는 <user-agent> 태그 앞에 나타난다. 예를 들어, 값 = 'UP.Browser/3.1'을 읽은 <user-agent> 태그는 값 = 'UP.Browser'을 읽은 <user-agent> 앞에 위치한다. <device-class> 요소는 이들의 성능에 기초하여 서로 다른 클라이언트를 임의의 그룹핑으로 분리하는데 이용된다. <device-class> 값들의 몇몇 예는 "WAP", "i-모드", "HDML", 및 "j-폰"일 수 있다.

일단, 매칭되는 <user-agent> 태그가 발견되면, CCM 은 주로 HTTP 헤드 (HEAD) 요청을 문서 소스에 발행함으로써, 요청되는 MIME 타입의 미디어 문서를 결정한다. 일단 MIME 타입이 알려지면, CCM 은 <user-agent> 블록 내에서 적합한 <content-type> 블록을 찾는다. 상기 예에서, "image/" 로 시작하는 MIME 타입에 대한 어떠한 요청도 제 1 <content-type> 태그의 패턴 속성에서 정의된 보통의 표현에 매칭될 것이다. 실제 구성 파일은 각각의 지원된 미디어 타입 또는 서브타입에 대해 분리된 블록을 가질 수 있다.

클라이언트 성능은 하나 이상의 <capability> 태그에 의해 <content-type> 내에서 정의된다. 각각의 미디어 타입은 서로 다른 성능을 요청할 수도 있다. 적절히 이미지를 디스플레이하기 위해, 디스플레이 폭, 높이, 및 컬러 농도를 알 필요가 있다. 적합한 오디오 스트림을 제공하기 위해, 대역폭 및 장치가 다수 채널을 플레이할 수 있는지 여부를 알 필요가 있다. 가장 간단한 경우, 구성 파일은 필수 (mandatory) 디폴트 속성을 통해 각각의 성능에 이전에-저장된 값들을 제공한다. 상기 예의 라인 4 및 11은, 모든 UP.Browser 사용자 에이전트들에 대한 컬러 농도 및 출력 포맷을 각각 픽셀 당 8 비트 및 image/bmp로 설정하여 이 경우를 설명한다. 상술한 바와 같이, 몇몇 사용자 에이전트들은 비=표준 HTTP 헤더들의 형태로 부가적 정보를 서버에 제공한다. <header> 태그는 <capability> 태그의 내에서 사용되어 CCM 이 이러한 헤더들이 이러한 헤더들을 분석하고 헤더 값에 의존하여 장치 성능을 설정하도록 지시할 수 있다. 이 예에서, 라인 6 은 비-표준 "x-up-devcap-screenpixels" 헤더가, 만약 존재한다면, 패턴 속성으로 제공된 보통의 표현을 헤더의 값에 적용함으로써 장치 디스플레이 폭을 설정하도록 이용되어야 함을 지시한다. 삽입구는 성능에 할당할 헤더값의 부분을 고립하는데 이용된다.

상기에서 명확히 나타나지 않았지만, 언더라잉 통신 전송은 또한 장치의 클래스 또는 타입으로부터 추론될 수 있다. 예를 들어, 목표 장치가 셀룰러 전화라면, 이 시스템은 하위 통신 전송이 무선이라고 추론할 수 있다. 다른 예로서, 목표 장치가 WAP 를 이용하여 통신하는 페이저라면, 이 시스템은 목표 장치가 제한된 대역폭 (셀룰러 전화에 비해) 을 갖는 무선 통신을 이용한다고 추론할 수 있다. 장치 클래스 및 착신 요청에 기초하여, 이 시스템은 보통 통신 전송이 무선인지 또는 유선인지 여부를 구별한다. 또한, 장치가 무선 및 유선 성능을 모두 갖는 경우는 거의 없다. 통상적인 유선 연결은 T1, DSL, 케이블 모뎀 및 다이얼-업 연결을 포함한다. 무선의 경우, 통상적인 연결은 9600-보드 (baud) 회선-교환 데이터 콜, 9600-보드 패킷-교환 콜, 또는 더 새로운 64K 보드 GPR 콜을 포함한다.

또한, 클라이언트 성능 모듈은 원본 컨텐츠의 소스를 확인 (verifying) 하여, 시스템이 제 3 의 집단이 아닌 인증된 참여 사이트의 컨텐츠를 재포맷하는데에만 이용되도록 한다. 보안은, 컨텐츠가 전송되는 지정된 서버의 이름을 포함하는 구체적인 URL 에 응답하여 CCM 을 활성화함으로써만 실시될 수 있다.

CCM 은 특정 클라이언트 장치의 성능을 유도하는 정보를 이용하여, 요청된 미디어 문서에 대하여 미디어 변환 모듈에 대한 요청을 구성한다. 이 CCM 은 이 구성된 요청을 전송하며, 적절한 재포맷 정보 및 MTM 에 대한 클라이언트 접속을 포함한다. 클라이언트 성능 모듈은 (선택적으로) 변환된 미디어 문서에 대해 프론트 사이드 캐쉬를 수행한다. 이 프론트-사이드 캐쉬는, 요청이 MTM 에 전송되기 전에 구성된 요청의 기준을 충족시키는 변형된 문서에 질의된다. 이 프론트-사이드 캐쉬의 목적은 MTM 모듈에의 부하를 최소화하는 것이다.

2. 미디어 변환 모듈

미디어 변환 모듈 (또는 트랜스코딩) 모듈 (MTM) 은, 요청 파라미터로 포맷 지시를 포함하는 미디어 문서에 대한 HTTP 요청을 수용하며, 이 지시에 따라 미디어를 재포맷한다. MTM 은 이미지 또는 비디오와 같은 단일 미디어 타입으로 특화될 수도 있으며, 또는 멀티플 타입의 미디어를 지원할 수도 있다. 기본적인 동작에서, 미디어 변환 모듈은, 이미지를 다음의 MIME 타입: image/jpg, image/bmp, image/gif, image/tiff, image/wbmp, image/iff, image/pcx, 및 image/png 로 변환함으로써; 이미지 치수를 감소, 증가시킴으로써; 클라이언트 디스플레이의 종횡비를 일치시키기 위해 이미지의 회전을 지원함으로써; 그리고 이미지 컬러 농도 (color depth) 의 감소, 증가를 지원함으로써 이미지 재포맷을 지원한다. MTM 은, 오디오 파일 및 스트림을 다음의 MIME 타입: audio/aiff, audio/au, audio/mpeg, audio/wav 로 변환함으로써; 그리고 오디오 비트 레이트를 감소시킴으로써 오디오 재포맷을 지원한다.

MTM 은 비디오 파일 및 스트림을 다음의 MIME 타입: video/mpeg, video/quicktime, video/x-msvideo (AVI), video/x-ms-asf, video/rm, 및 video/mjpeg 으로 변환함으로써 비디오 재포맷을 지원한다. 또한, MTM 은, RFC 2046, Multipurpose Internet Mail Extensions (MIME) 로 정의된 부가적인 멀티미디어 컨텐츠 타입의 재포맷을 지원할 수 있다. RFC 2046 의 복사본은 현재 인터넷 http://www.ietf.org/rfc/rfc2046.txt 에서 이용가능하다.

그 동작을 설명하는 미디어 변환 모듈의 동작의 예를 이하 설명한다. 이 예에서, MTM 은 JPEG 이미지를 전환 (translate) 하고 다음의 것을 입력으로 받는다.

출력의 치수 (폭 및 높이);

출력 장치의 타입;

컬러 공간 (예를 들어, RGB 또는 그레이스케일)

컬러 팔레트 (예를 들어, 트루 컬러 또는 인덱스); 및

압축 기술.

이러한 입력으로부터, MTM 은 특정된 출력 사이즈로 특정된 출력 포맷으로 그림을 출력할 수 있다. 일부 장치는 고유의 특징과 다르게 특징지워질 수도 있다. 예를 들어, 16 컬러 LCD 디스플레이가 트루 컬러 장치로 특징지워지는 것이 바람직할 수도 있다. 그 후, MTM 에 의해 전송된 트루 컬러 모드 이미지를 내부 소프트웨어/하드웨어를 이용하여 내부 16 컬러 모드로 전환하는 것은 장치의 책임이다. 임의의 적합한 압축 기법이 채택될 수 있으며, 이는 소유 또는 비소유의 기법을 포함한다. 예를 들어, JPEG, JBIG, GIF 등이 포함된다. 예를 들어, JPEG-like Image Compression (Parts 1 and 2), Dr. Dobb's Journal, July 1995 및 August 1995 (캘리포니아 산 마테오의 Dr. Dobb's Journal 의 Dr/Dobb's/CD Release 6 의 CD-ROM 에서 이용가능) 을 참조한다.

상술한 JPEG 이미지를 전환함에 있어서 구체적인 동작은 다음과 같다. 먼저, 입력 그림은 압축해제되어, 채택된 컬러 공간에서 비트맵을 형성한다. 예를 들어, Clikpix 는 GUV 컬러 공간을 이용하며; JPEG 는 멀티플 컬러 공간을 이용하며, 이는 YUV 및 RGB 를 이용한다. GUV 컬러 공간은 2000 년 1 월 21 일에 출원된 공유 출원 제 09/489,511 호에 기재되어 있으며; 산업-표준 컬러 공간의 기재 또한 그 출원에서 찾을 수 있다. 그 후, 그림은 "표준" 중간 포맷으로 변환되며, 통상적으로 산업-표준 컬러 공간 중의 하나이다. 예를 들어,

L,a,b 16 비트/픽셀/채널 (예를 들어, Adobe Photoshop 에서 이용됨);

SRGB 8 비트/픽셀/채널 (예를 들어, Microsoft, HP 등에 의해 이용됨); 및

YUV

를 포함한다.

그 후, 중간 포맷은 다음의 처리로 출력 장치에 의해 요구되는 포맷으로 맵핑된다.

1. 이미지 스케일링 - 이미지를 원하는 출력 사이즈로 스케일링함.

2. 오직 모노크롬 정보만을 원한다면, 표준 변환 방법 (예를 들어 R, G, B 신호로부터 발광 신호 Y 를 생성하는, 국제 전화 협회 (ITU) 권장안, 예를 들어 ITU-권장안 BT.6011 Encording parameters of Digital Television for studio 를 참조) 을 이용하여, 이미지의 모노크롬 버전이 생성된다.

3. 비트/픽셀이 8 보다 작다면, (에러 디퓨전, 블루 노이즈 마스킹 등, 예를 들어 Reiner Eschbach 가 수집 및 편집한 Recent Progress in Digital Halftoning, 1994, The Society of Imaging Science and Technology, ISBN 0-892080-181-3 를 참조) 디더링 기술이 이용된다.

4. 출력 장치가 (예를 들어, 256 컬러만을 지원하는) 컬러 팔레트를 갖는다면, 컬러 디더링 기법이 이용된다. 그러한 기법은 Foley, Van Dam 등의 Computer Graphics-Principles and Practice, 제 2 판, 1990, Addison-Wesley, Reading, MA, ISBN 0-201-12110-7 에 상세히 논의되어 있다.

5. 데이터가 스트림 아웃되기 전에 압축이 선택적으로 수행된다. 트루-컬러 이미지에 대해, 바람직한 압축 기법은 JPEG 이다. 인덱스된 이미지에 대해서는, GIF, PNG 가 바람직한 방법이다. 하프톤된 이미지에 대해서는, JBIG 압축이 바람직한 방법이다.

최종적으로, 생성된 그림이 출력되고, 장치의 디스플레이에 최종적으로 표시되게 하는 목표 장치로의 스트리밍이 준비된다.

미디어 내용의 아이템을 재포맷하는 미디어 변환 모듈의 동작의 예가 다음의 "AutoRotateOp" 함수로 표현된다.

```
 1: #include "autorotateop.h"
 2:
 3: AutoRotateOp::AutoRotateOp()
 4: {
 5: }
 6:
 7: AutoRotateOp::~AutoRotateOp()
 8: {
 9: }
10:
11: const char *AutoRotateOp::Name()
12: {
13:   return "autorotate";
14: }
15:
16: const char *AutoRotateOp::Args()
17: {
18:   return "displayWidth(160),displayHeight(120),clockwise(1)";
19: }
20:
```

```
21: const char *AutoRotateOp::Example()
22: {
23:   return "autorotate=118,157";
24: }
25:
26: void AutoRotateOp::Process(IMG_image *img)
27: {
28:   int32 w, h, cw;
29:   float displayAspect, photoAspect;
30:
31:   GetArg(&w, 160);
32:   GetArg(&h, 120);
33:   GetArg(&cw, 1);
34:
35:   displayAspect = (float)w / (float)h;
36:   photoAspect = (float)(img->width) / (float)(img->height);
37:
38:   if ((displayAspect > 1.0f && photoAspect < 1.0f) ||
39:         (displayAspect < 1.0f && photoAspect > 1.0f))
40:   {
41:         if (cw)
42:               IMG_RotateRight(img);
43:         else
44:               IMG_RotateLeft(img);
45:   }
46: }
47:
```

상기 AutoRotateOp 함수는 이미지를 자동적으로 회전시켜서 특정 장치의 사용가능한 디스플레이에 더 적합하게 한다. 상기 라인 26 에 나타난 것과 같이, 함수는 이미지에 대한 포인터를 파라미터로서 받는다. 라인 28 내지 36에서, 장치의 디스플레이 특징을 얻는다. 라인 38 내지 39 의 조건은 이미지가 포트레이트 중심으로 (즉, 이미지를 수직으로 디스플레이) 표시해야 하는지의 여부 또는 랜드스케이프 중심으로 (즉, 이미지를 수평으로 디스플레이) 표시해야 하는지의 여부를 평가하여, 장치 디스플레이에 더 적합하게 한다. 이 평가의 결과에 의존하여, 라인 41 내지 44 에 나타난 바와 같이 IMG RotateRight 또는 IMG RotateLeft 에 호출 (call) 이 생기며, 이미지는 시계 방향 또는 반시계 방향으로 회전하여 특정 장치의 디스플레이에 적합하게 된다. IMG RotateRight 및 IMG RotateLeft 함수를 더 상세히 나타내는 소스 코드 리스트가 부록 A 로서 첨부된다.

MTM 은 가능할 때마다 오리지널 미디어 객체의 캐쉬된 버전을 이용한다. MTM 은, 소스 미디어를 판독하는데 요구되는 시간을 감소시키고 미디어 전송 시스템에 의한 인터넷 대역폭 소비를 감소시키기 위해, 백사이드 캐쉬에 소스 미디어 객체를 캐쉬한다. 이 백사이드 캐쉬는 MTM 에 의해서 또는 중간 리버스 프록시 캐쉬 (intermediate reverse proxy cache) 에 의해 수행될 수 있다. 소스 객체는 HTTP 캐쉬-제어 헤더에 구체화된 방향에 따라 캐쉬된다. 현재 바람직한 실시형태에서, 리버스 프록시 캐쉬로 구성된 아파치 HTTP 서버가 인터넷와 MTM 모듈 "사이에" 채택될 수 있으며, 따라서 소스 멀티미디어 문서에 대한 어떠한 요청이든지 먼저 로컬 디스크 캐쉬와 비교된다. 아파치 리버스 프록시 캐쉬 모듈은, 더 나은 범위성 및 유지 성능을 의해, 캐쉬 작업과 미디어 변환 작업을 분리시킨다.

C/C++ 프로그램 언어로 된 소스 리스트를 담고 있는 부록 A, 컴퓨터 프로그램 리스트가 첨부되며, 본 발명의 더 나은 이해를 제공한다. C/C++ 프로그램을 만들기에 적합한 환경은, 캘리포니아 스코트 밸리의 Borland Software Corporation 의 Borland(R) C++ Builder 또는 워싱턴 레드몬드의 Microsoft Corporation 의 Microsoft(R) Visual C++ 을 포함한 여러 판매자로부터 이용가능하다.

본 발명이 단일의 바람직한 실시예 및 특정한 대체안을 참조하여 상세히 설명하였으나, 본 발명을 그 특정한 실시형태 또는 특정한 대체안으로 제한하려는 의도는 아니다. 예를 들어, 당업자는 바람직한 실시형태에 행해진 변형이 본 발명의 교시로부터 일탈하지 않는다는 것을 이해할 것이다.

컴퓨터 프로그램 리스트 (부록 A)

IMGXFORM

```
#include "vimage.h"
#include "imglib.h"

IMGLIB_API void IMG_FlipH(IMG_image *img)
{
	if (img->pixelBytes != 4)
	{
		DEBUG_IMG_FORMAT;
		return;
	}

	int32 x, y, w;
	uint8 *rowPtr;
	uint32	*pixPtr, *pixPtr2, tempPix;

	w = img->width >> 1;
	rowPtr = img->baseAddr;
	for (y = 0; y < img->height; y++)
	{
		pixPtr = pixPtr2 = (uint32 *)rowPtr;
		pixPtr2 += img->width - 1;
		for (x = 0; x < w; x++)
		{
			tempPix = *pixPtr;
			*pixPtr++ = *pixPtr2;
			*pixPtr2-- = tempPix;
		}
		rowPtr += img->rowBytes;
	}
}

IMGLIB_API void IMG_FlipV(IMG_image *img)
{
	img->baseAddr += img->rowBytes * (img->height - 1);
	img->rowBytes = -img->rowBytes;
}

IMGLIB_API void IMG_RotateLeft(IMG_image *img)
{
	if (img->pixelBytes != 4)
	{
		DEBUG_IMG_FORMAT;
		return;
	}

	IMG_image	tempImg;

	if (!IMG_DuplicateImage(img, &tempImg))
		return;

	int32 x, y;
	uint8 *rowPtr, *srcRow, *srcPix;
	uint32	*pixPtr;

	if (img->rowBytes < 0)
	{
		img->baseAddr += (img->height - 1) * img->rowBytes;
		img->rowBytes = -img->rowBytes;
	}
	img->width = tempImg.height;
	img->height = tempImg.width;
	img->rowBytes = img->pixelBytes * img->width;
	rowPtr = img->baseAddr;
	srcRow = tempImg.baseAddr + tempImg.pixelBytes * (tempImg.width -
1);
	for (y = 0; y < img->height; y++)
	{
		pixPtr = (uint32 *)rowPtr;
		srcPix = srcRow;
		for (x = 0; x < img->width; x++)
		{
			*pixPtr = *((uint32 *)srcPix);
```

```
                pixPtr++;
                srcPix += tempImg.rowBytes;
            }
            rowPtr += img->rowBytes;
            srcRow -= tempImg.pixelBytes;
        }
        IMG_FreeImage(&tempImg);
}

IMGLIB_API void IMG_RotateRight(IMG_image *img)
{
        if (img->pixelBytes != 4)
        {
            DEBUG_IMG_FORMAT;
            return;
        }

        IMG_image   tempImg;

        if (!IMG_DuplicateImage(img, &tempImg))
            return;

        int32 x, y;
        uint8 *rowPtr, *srcRow, *srcPix;
        uint32      *pixPtr;

        if (img->rowBytes < 0)
        {
            img->baseAddr += (img->height - 1) * img->rowBytes;
            img->rowBytes = -img->rowBytes;
        }
        img->width = tempImg.height;
        img->height = tempImg.width;
        img->rowBytes = img->pixelBytes * img->width;
        rowPtr = img->baseAddr;
        srcRow = tempImg.baseAddr + tempImg.rowBytes * (tempImg.height -
1);
        for (y = 0; y < img->height; y++)
        {
            pixPtr = (uint32 *)rowPtr;
            srcPix = srcRow;
            for (x = 0; x < img->width; x++)
            {
                *pixPtr = *((uint32 *)srcPix);
                pixPtr++;
                srcPix -= tempImg.rowBytes;
            }
            rowPtr += img->rowBytes;
            srcRow += tempImg.pixelBytes;
        }
        IMG_FreeImage(&tempImg);
}

IMGLIB_API void IMG_Rotate180(IMG_image *img)
{
        IMG_FlipV(img);
        IMG_FlipH(img);
}

IMGLIB_API void IMG_Resize(IMG_image *img, int32 newWidth, int32
newHeight, bool highQuality)
{
        if (img->pixelBytes != 4)
        {
            DEBUG_IMG_FORMAT;
            return;
        }

        if (img->width == newWidth && img->height == newHeight)
            return;

        IMG_image   tempImg;

        if (!IMG_AllocateImage(&tempImg, newWidth, newHeight, img-
>imgFormat))
            return;
```

```
//IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight, true);
      if (highQuality)
      {
            VImage            vImg;
            vImg.Import(img);
            vImg.Scale(newWidth, newHeight, CImageServer::CUBIC);
            vImg.Export(&tempImg);
      }
      else
            IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight,
false);

      IMG_CopyTags(img, &tempImg);
      IMG_FreeImage(img);
      *img = tempImg;
}

IMGLIB_API void IMG_ClampResize(IMG_image *img, int32 maxWidth, int32
maxHeight, bool highQuality)
{
      float f, aspect, newAspect;
      int32 width, height;

      aspect = (float)(img->width) / (float)(img->height);
      newAspect = (float)maxWidth / (float)maxHeight;
      if (aspect > newAspect)
      {
            width = maxWidth;
            f = (float)width / aspect;
            height = ROUND(f);
            height = CLAMP(height, 1, maxHeight);
      }
      else
      {
            height = maxHeight;
            f = (float)height * aspect;
            width = ROUND(f);
            width = CLAMP(width, 1, maxWidth);
      }
      IMG_Resize(img, width, height, highQuality);
}
```

Apache Module

```
/**
 * This function creates a device capabilities object
 * and constructs the URL that is passed to the
 * Media Transcoding Module.
 *
 * @param r
 * @return
 */
static int uts_rewrite_uri(request_rec *r) {
    int status;

    // skip if already a proxy
    if (r->proxyreq != NOT_PROXY) {
        return OK;
    }

    // skip all but GET requests
    if (strcasecmp("GET", r->method) != 0) {
        return OK;
    }

    // get config file
    uts_dir_config      *cfg =  (uts_dir_config *)
ap_get_module_config(r->per_dir_config, &uts_module);
    uts_server_config   *scfg = (uts_server_config *)
ap_get_module_config(r->server->module_config, &uts_module);

    // (subrequest for content type)
    string contentType = "image/jpeg";
```

```
    /**
     * first, parse the full incoming URI.  We're going to treat the
path+args
     * of this URI as the full URI for the proxy request
     */
    uri_components uc;
    status = ap_parse_uri_components(r->pool, r->uri, &uc);
    if (status != HTTP_OK) {
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Bad input URI: %s", r->uri);
        return DECLINED;
    }

    char *path = ap_pstrdup(r->pool, uc.path);

    // pass in headers, get device capabilities back
    DeviceIdentifier::DeviceCapabilities devcap;
    status = dev_ident->getDeviceCapabilities(&devcap, r, contentType);
    if (status != DI_SUCCESS) {
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Device capabilities lookup failed");
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Recording headers") ;
        logHeaders (r) ;
        return DECLINED;
    }

    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server,
"%s", devcap.toString().c_str());

    char *sub;
    // pop off initial slash
    sub = ap_getword_nc(r->pool, &path, '/');   // initial slash

    // if we have an additional subscriber ID, pop it off, too
    if (cfg->subscriber_id_on_url) {
        sub = ap_getword_nc(r->pool, &path, '/');   // subscriber ID
    }

    //
    // Build option list.
    // TODO:  This should really be a function that returns a string.
    //
    char *cfg_clamp, *cfg_mimetype, *cfg_path, *cfg_quality,
*cfg_argparam, *cfg_rotate, *cfg_scale, *cfg_filesize ;
    char blank = '\0' ;

    // Clamp or scale?
    if (!devcap.scale_to_width)
    {
      cfg_clamp = ap_psprintf (r->pool, "%sclampsize=%d&",
        (strchr (cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"),
        (devcap.display_height > devcap.display_width ?
devcap.display_height : devcap.display_width));
      cfg_scale = ap_psprintf (r->pool, "%c", blank) ;
    } else {
      cfg_clamp = ap_psprintf (r->pool, "%s",
        (strchr (cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"));
      cfg_scale = ap_psprintf (r->pool, "limitsize=%i,0&",
devcap.display_width) ;
    }

    // Path
    if (path[strlen(path)-2] == '.')
    {
      switch (path[strlen(path)-1])
      {
        case 'b':
          cfg_path = ap_psprintf (r->pool, "in=\"http://%smp\"&", path)
;
          break ;
        case 'g':
          cfg_path = ap_psprintf (r->pool, "in=\"http://%sif\"&", path)
;
          break ;
        case 'j':
```

```
          cfg_path = ap_psprintf (r->pool, "in=\"http://%spg\"&", path)
;
          break ;
        case 'p':
          cfg_path = ap_psprintf (r->pool, "in=\"http://%sng\"&", path)
;
          break ;
        case 't':
          cfg_path = ap_psprintf (r->pool, "in=\"http://%sif\"&", path)
;
          break ;
      }
    } else {
      cfg_path = ap_psprintf (r->pool, "in=\"http://%s\"&", path) ;
    }

    // Mime Type
    cfg_mimetype = ap_psprintf (r->pool, "outformat=%s&",
devcap.mime_type.c_str()) ;

    // Quality
    if (devcap.image_quality != -1)
      cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
devcap.image_quality) ;
    else
      cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
devcap.color_depth) ;

    // Rotate?
    if (devcap.rotate_to_fit)
      cfg_rotate = ap_psprintf (r->pool, "autorotate=%i,%i,0&",
        devcap.display_width,
        devcap.display_height) ;
    else
      cfg_rotate = ap_psprintf (r->pool, "%c", blank) ;

    // Arguments
    if (r->args)
      cfg_argparam = ap_psprintf (r->pool, "%s&", r->args) ;
    else
      cfg_argparam = ap_psprintf (r->pool, "%c", blank) ;

    // Filesize
    if (devcap.maximum_file_size)
      cfg_filesize = ap_psprintf (r->pool, "filesize=%i&",
devcap.maximum_file_size) ;
    else
      cfg_filesize = ap_psprintf (r->pool, "%c", blank) ;

    //
    // End option list build
    //

    // create URL for lsps
    //
[1]URL[2]Clamp[3]path[4]format[5]quality[6]rotate[7]argparam[8]scale[9]f
ilesize
    char *url = ap_psprintf (r->pool, "%s%s%s%s%s%s%s%s%s",
      cfg->media_transcoder_uri,
      cfg_clamp,
      cfg_path,
      cfg_mimetype,
      cfg_quality,
      cfg_rotate,
      cfg_argparam,
      cfg_scale,
      cfg_filesize) ;

    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server,
"proxying: %s", url);

    // double-check that this is a valid URL
    status = ap_parse_uri_components(r->pool, url, &uc);
    if (status != HTTP_OK) {
      ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Bad proxy URI: %s", url);
```

```
        return DECLINED;
    }

    // internal redirect won't work for remote servers, so
    // set this up as a proxy
    /* example of working request member variables
    r->proxyreq     = PROXY_PASS;
    r->uri          = "/lsps";
    r->filename     = "proxy:http://pegli-
nt4.office.lightsurf.com:10104/lsps";
    r->args         =
"clampsize=100&in=http://www.lightsurf.com/images/misc/pk_sunset.jpg";
    r->handler      = "proxy-server";
    */

    r->proxyreq     = PROXY_PASS;
    r->uri          = ap_pstrdup(r->pool, uc.path);
    r->filename     = ap_pstrcat(r->pool, "proxy:", uc.scheme, "://",
uc.hostinfo, uc.path, NULL);
    r->args         = ap_pstrdup(r->pool, uc.query);
    r->handler      = "proxy-server";

    return DECLINED;
}
```

DeviceIdentifier

```
/**
 * DeviceIdentifier.cpp
 * implementation of DeviceIdentifier and DeviceCapabilities classes
 * 4/23/2001 pae
 */


#include <string>
#include <strstream.h>
#include <stdlib.h>

#include "httpd.h"
#include "http_log.h"

#include "Regex.hpp"
#include "DeviceIdentifier.hpp"

static Regex regex;


DeviceIdentifier::DeviceIdentifier(const char * filename) {
    try {
        config = new XMLConfigFile(filename);
    }
    catch (...) {
    }
}

DeviceIdentifier::DeviceIdentifier() {
    config = new XMLConfigFile("/lsurf/uts/conf/devices.xml");
}

DeviceIdentifier::~DeviceIdentifier() {
    if (config != NULL) {
        delete config;
    }
}


/**
 * Retrieve a named capability value from the configuration
 * file.  This function will first find the &lt;capability&gt;
 * node identified by <code>capabilityName</code>, then
 * determine if any &lt;header&gt; tags apply.  The
 * value is returned as a string, which can then be
 * converted to the appropriate type.
 *
```

```
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
 */
string DeviceIdentifier::getCapability(request_rec *r, string baseXPath,
string capabilityName) {

    /**
     * The device capabilities data storage is implemented as an XML
file.
     * A member variable named "config" allows us to make XPath queries
     * against the XML file.
     */
    string cquery =
    baseXPath +
    string("/capability[@name=\"") +
    capabilityName +
    string("\"]");

    string value;

    // first, look through headers (if any)
    XMLConfigFile::StringVectorType headernames = config->query(cquery +
string("/header/@name"));
    for (unsigned int i=0; i < headernames.size(); i++) {
        const char *headerval = ap_table_get(r->headers_in,
headernames[i].c_str());
        if (headerval != NULL) {
            // found a matching header, so grab the pattern and run the
regex
            XMLConfigFile::StringVectorType patterns = config-
>query(cquery + string("/header[@name=\"") + headernames[i] +
string("\"]/@pattern"));
            if (regex.group(patterns[0], string(headerval), &value)) {
                // yes, there is a match between the <header pattern="">
and the header value
                if (value.size() > 0) {
                    // value now contains the matched substring
                    break;
                } else {
                    // there was a pattern match, but no parenthesized
substring, so use the value of the "default" attribute
                    XMLConfigFile::StringVectorType defaults = config-
>query(cquery + string("/header[@name=\"") + headernames[i] +
string("\"]/@default"));
                    value = defaults[0];
                }
            }
        }
    }

    // if we didn't get anything from the headers, use the default value
    XMLConfigFile::StringVectorType defaults = config->query(cquery +
string("/@default"));
    if (defaults.size() > 0) {
        value = defaults[0];
    }

    // TODO: if there isn't a default, die a horrible death
    // this may not be necessary if we publish a schema for the config
file

    return value;
}


/**
 * Convert the result of <code>getCapability()</code>
 * to an integer.
 *
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
```

```
 */
int DeviceIdentifier::getCapabilityInt(request_rec *r, string baseXPath,
string capabilityName) {
    string value = getCapability(r, baseXPath, capabilityName);
    if (value.size() > 0) {
        return atoi(value.c_str());
    } else {
        return -1;
    }
}


/**
 * Convert the result of <code>getCapability()</code>
 * to a boolean.
 *
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
 */
int DeviceIdentifier::getCapabilityBool(request_rec *r, string
baseXPath, string capabilityName) {
    string value = getCapability(r, baseXPath, capabilityName);
    return(!value.compare("true") || !value.compare("1"));  // anything
but true or 1 is considered false
}


/**
 * Fill in a DeviceCapabilites structure based on the
 * HTTP request headers and the content type of the
 * requested document.  The main job of this function
 * is to find the correct node in the config file for
 * the given User-Agent header, then call
 * <code>getCapabilityStr()</code>, <code>getCapabilityInt()</code>,
 * and <code>getCapabilityBool()</code> to fill in
 * the device capabilities.
 *
 * @param devcap
 * @param r
 * @param contentType
 */
int
DeviceIdentifier::getDeviceCapabilities(DeviceIdentifier::DeviceCapabili
ties *devcap, request_rec *r, string contentType) {

    if (devcap == NULL) {
        // sorry, no can do
        ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Cannot call
getDeviceCapabilities() with a null DeviceCapabilities parameter");
        return DI_ERROR;
    }

    XMLConfigFile::StringVectorType xpath_results;
    string base_xpath_query;

    // find the correct user-agent node
    const char *ua_header = ap_table_get(r->headers_in, "User-Agent");

    xpath_results = config->query("//user-agent/@pattern");
    int found = FALSE;
    for (unsigned int i=0; i < xpath_results.size(); i++) {
        if (regex.match(xpath_results[i], string(ua_header))) {
            base_xpath_query.append("//user-agent[@pattern=\"");
            base_xpath_query.append(xpath_results[i]);
            base_xpath_query.append("\"]");
            found = TRUE;
            break;
        } else if (regex.last_error.size() > 0) {
            // this will complain a lot if there are regex errors
            ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
expression error: %s", regex.last_error.c_str());
```

```
            ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex:
%s", xpath_results[i].c_str());
        }
    }
    if (!found) {
        ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
configuration entry for User-Agent \"%s\"", ua_header);
        // TODO: log or send email
        return DI_NOTFOUND;
    }

    // within that node, find the correct mime-type node
    xpath_results = config->query(base_xpath_query + string("/mime-
type/@pattern"));

    found = FALSE;
    for (unsigned int i=0; i < xpath_results.size(); i++) {
        if (regex.match(xpath_results[i], contentType)) {
            base_xpath_query.append("/mime-type[@pattern=\"");
            base_xpath_query.append(xpath_results[i]);
            base_xpath_query.append("\"]");
            found = TRUE;
            break;
        } else if (regex.last_error.size() > 0) {
            // again, here's another big complainer
            ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
expression error: %s", regex.last_error.c_str());
            ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex:
%s", xpath_results[i].c_str());
        }
    }
    if (!found) {
        ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
<mime-type> entry in config file for document MIME type \"%s\"",
contentType.c_str());
        // TODO: log or send email
        return DI_NOTFOUND;
    }

    ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Successfully
identified device, User-Agent: \"%s\"", ua_header);

    // fill in the DeviceCapabilities structure
    // note defaults are:  string="", int=-1, bool=false
    devcap->mime_type           = getCapability(r, base_xpath_query,
"output-mime-type");
    devcap->display_width       = getCapabilityInt(r, base_xpath_query,
"display-width");
    devcap->display_height      = getCapabilityInt(r, base_xpath_query,
"display-height");
    devcap->color_depth         = getCapabilityInt(r, base_xpath_query,
"color-depth");
    devcap->image_quality       = getCapabilityInt(r, base_xpath_query,
"image-quality");
    devcap->color_capable       = getCapabilityBool(r, base_xpath_query,
"color-capable");
    devcap->rotate_to_fit       = getCapabilityBool(r, base_xpath_query,
"rotate-to-fit");
    devcap->scale_to_width    = getCapabilityBool(r, base_xpath_query,
"scale-to-width");
    devcap->palette           = getCapability(r, base_xpath_query,
"palette");
    devcap->maximum_file_size = getCapabilityInt(r, base_xpath_query,
"maximum-size");
    devcap->video_frame_rate    = getCapabilityInt(r, base_xpath_query,
"video-frame-rate");
    devcap->audio_encoding_rate = getCapabilityInt(r, base_xpath_query,
"audio-encoding-rate");
    devcap->audio_channels      = getCapabilityInt(r, base_xpath_query,
"audio-channels");

    return DI_SUCCESS;
};
```

```
/*************************************************************************
********/

// utility functions to print out the DeviceCapabilities object

void DeviceIdentifier::DeviceCapabilities::print(ostream *os) {
    *os << this->toString();
};

string DeviceIdentifier::DeviceCapabilities::toString() {
    std::ostrstream buffer;
    buffer << "DeviceCapabilites {"
    << " mime-type: " << mime_type
    << "; display width: " << display_width
    << "; display height: " << display_height
    << "; color depth: " << color_depth
    << "; color capable? " << (color_capable ? "yes" : "no")
    << "; rotate to fit? " << (rotate_to_fit ? "yes" : "no")
    << "; video frame rate: " << video_frame_rate
    << "; audio encoding rate: " << audio_encoding_rate
    << "; audio channels: " << audio_channels
    << " }"
    << "\0";
    return buffer.str();
};

ostream &operator<<(ostream &os, DeviceIdentifier::DeviceCapabilities
&devcap) {
    devcap.print(&os);
    return os;
};
```

(57) 청구의 범위

청구항 1.

온라인 시스템에서 클라이언트 장치의 성능을 결정하고 상기 장치에 적합한 포맷의 미디어 컨텐츠를 제공하는 방법으로서,

목표 장치에 특정 미디어 객체의 복사본을 제공하는 요청을 수신하는 단계;

상기 목표 장치의 성능을 결정하는 단계;

상기 목표 장치의 상기 성능에 기초하여, 상기 목표 장치에 상기 미디어 객체의 복사본을 제공하기 위해 원하는 포맷을 결정하는 단계;

상기 특정 미디어 객체를 상기 결정된 포맷을 갖는 복사본으로 전환 (translate) 시키는 단계; 및

상기 목표 장치에 상기 결정된 포맷을 갖는 상기 복사본을 제공하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 2.

제 1 항에 있어서,

상기 결정된 포맷을 갖는 상기 복사본을 캐쉬 메모리에 저장하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 3.

제 2 항에 있어서,

상기 결정된 포맷으로 상기 특정 객체에 대한 후속적인 요청을 목표 장치로부터 수신하는 단계; 및

상기 캐쉬 메모리에 저장된 상기 복사본을 상기 목표 장치에 제공하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 4.

제 1 항에 있어서,

상기 미디어 객체의 전환을 위해 접속된 서버로부터 상기 특정 미디어 객체의 복사본을 획득하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 5.

제 4 항에 있어서,

상기 접속된 서버로부터 수신한 상기 미디어 객체의 캐쉬된 복사본을 캐쉬 메모리에 저장하는 단계; 및

상기 미디어 객체의 전환을 위한 후속적인 요청에 응답하여, 캐쉬 메모리에 저장된 상기 미디어 객체의 상기 복사본을 이용하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 6.

제 1 항에 있어서,

상기 목표 장치의 상기 성능은 스크린 해상도를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 7.

제 1 항에 있어서,

상기 목표 장치의 상기 성능은 스크린 사이즈를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 8.

제 1 항에 있어서,

상기 목표 장치의 상기 성능은 컬러 지원을 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 9.

제 1 항에 있어서,

상기 목표 장치의 상기 성능은 비트 레이트를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 10.

제 1 항에 있어서,

상기 목표 장치의 상기 성능은, 상기 목표 장치가 요청을 전송하는데 채택하는 현재 이용가능한 통신 미디어를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 11.

제 10 항에 있어서,

현재 이용가능한 통신 미디어는 무선 통신을 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 12.

제 10 항에 있어서,

현재 이용가능한 통신 미디어는 유선 통신을 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 13.

제 1 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에 의해 제공 (submit) 된 요청을 조사하는 것을 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 14.

제 1 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에 의해 제공된 HTTP 헤더를 조사하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 15.

제 14 항에 있어서,

상기 장치에 의해 제공된 HTTP 헤더를 조사하는 단계는 HTTP 사용자-에이전트 헤더를 조사하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 16.

제 1 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에게 성능을 질의 (query) 하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 17.

제 1 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 목표 장치의 장치 클래스에 기초하여 지식베이스 (knowledgebase) 로부터 성능을 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 18.

제 17 항에 있어서,

상기 장치의 성능이 상기 지식베이스에 부가될 수 있도록, 인식되지 않은 목표 장치의 로그 기록을 기록하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 19.

제 18 항에 있어서,

인식되지 않은 상기 목표 장치에 대하여 통지를 자동적으로 발하는 단계를 더 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 20.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는 상기 목표 장치에서 특정 이미지를 렌더링하는데 적절한 해상도를 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 21.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는 상기 목표 장치에서 특정 이미지를 렌더링하는데 적절한 컬러 공간을 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 22.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는 상기 목표 장치에서 특정 이미지를 렌더링하기 위해 적절한 이미지 사이즈를 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 23.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는, 목표 장치 디스플레이의 종횡비를 일치시키기 위해 상기 특정 이미지를 회전시킬지의 여부를 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 24.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는 상기 목표 장치에 적절한 비트 레이트를 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 25.

제 1 항에 있어서,

상기 원하는 포맷을 결정하는 단계는, 상기 특정 미디어 객체를 상기 목표 장치로 전송하는데 이용가능한 통신 대역폭을 결정하는 단계를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 26.

제 25 항에 있어서,

상기 이용가능한 인터넷 대역폭은, 적어도 부분적으로는, 상기 목표 장치로부터 수신한 HTTP 요청 헤더에 기초하여 결정되는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 27.

제 25 항에 있어서,

상기 이용가능한 인터넷 대역폭은, 적어도 부분적으로는, 상기 목표 장치에 대한 장치 클래스에 기초하여 결정되는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 28.

제 1 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 휴대형 컴퓨팅 장치를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 29.

제 1 항에 있어서,

상기 목표 장치는 디지털 오디오 성능을 갖는 휴대형 컴퓨팅 장치를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 30.

제 1 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 셀룰러 전화를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 31.

제 1 항에 있어서,

상기 목표 장치는 디지털 오디오 성능을 갖는 셀룰러 전화를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 32.

제 1 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 페이저 장치를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 33.

제 1 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 개인용 컴퓨터를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 34.

제 1 항에 있어서,

상기 목표 장치는 디지털 오디오 성능을 갖는 개인용 컴퓨터를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 35.

제 1 항에 있어서,

상기 목표 장치는 WAP (무선 애플리케이션 프로토콜) 지원을 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 36.

제 1 항에 있어서,

상기 미디어 객체는 디지털 이미지를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 37.

제 1 항에 있어서,

상기 미디어 객체는 디지털 비디오를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 38.

제 1 항에 있어서,

상기 미디어 객체는 디지털 오디오를 포함하는, 성능 결정 및 미디어 컨텐츠 제공 방법.

청구항 39.

목표 장치에 디지털 미디어를 제공하는 온라인 시스템으로서,

특정 목표 장치의 성능을 결정하기 위한 성능 모듈; 및

자동적으로 특정 미디어 객체의 복사본을 검색 (retrieve) 하여 목표 장치에 상기 객체의 복사본을 제공하되, 상기 복사본이 상기 목표 장치의 성능에 기초하여 특정 포맷으로 자동적으로 전환되는, 변환 모듈을 포함하는, 온라인 시스템.

청구항 40.

제 39 항에 있어서,

전환된 미디어 객체의 복사본을 저장하는 캐쉬 메모리를 더 포함하는, 온라인 시스템.

청구항 41.

제 40 항에 있어서,

상기 시스템은, 상기 캐쉬 메모리로부터 상기 특정 포맷의 상기 특정 객체의 복사본을 검색함으로써, 우선 상기 요청을 만족시키려고 시도하는, 온라인 시스템.

청구항 42.

제 39 항에 있어서,

검색된 미디어 객체를 저장하는 캐쉬 메모리를 더 포함하는, 온라인 시스템.

청구항 43.

제 42 항에 있어서,

상기 시스템은, 원격 서버로부터 복사본을 검색하기 전에, 우선 상기 캐쉬 메모리로부터 특정 미디어 객체의 복사본을 검색하려고 시도하는, 온라인 시스템.

청구항 44.

제 39 항에 있어서,

상기 시스템에 의해 저장된 각각의 디지털 객체는 고유의 URL (Uniform Resource Locator) 에 의해 식별되는, 온라인 시스템.

청구항 45.

제 44 항에 있어서,

상기 고유의 URL 은 상기 디지털 객체의 특징으로 인코딩되는, 온라인 시스템.

청구항 46.

제 44 항에 있어서,

상기 고유의 URL 은 상기 디지털 객체의 컬러 농도 (color depth) 를 포함하는, 온라인 시스템.

청구항 47.

제 44 항에 있어서,

상기 고유의 URL 은 상기 디지털 객체의 이미지 사이즈를 포함하는, 온라인 시스템.

청구항 48.

제 44 항에 있어서,

상기 고유의 URL 은 상기 디지털 객체의 해상도를 포함하는, 온라인 시스템.

청구항 49.

제 44 항에 있어서,

상기 고유의 URL 은 상기 디지털 객체의 비트 레이트를 포함하는, 온라인 시스템.

청구항 50.

제 44 항에 있어서,

상기 시스템은 각각의 디지털 객체에 대해 URL 을 저장하며,

상기 성능 모듈은, 상기 URL 로부터 상기 목표 장치에 제공될 수도 있는 상기 특정 디지털 객체를 결정할 수 있는, 온라인 시스템.

청구항 51.

제 39 항에 있어서,

상기 목표 장치의 상기 성능은 스크린 해상도를 포함하는, 온라인 시스템.

청구항 52.

제 39 항에 있어서,

상기 목표 장치의 상기 성능은 스크린 사이즈를 포함하는, 온라인 시스템.

청구항 53.

제 39 항에 있어서,

상기 목표 장치의 상기 성능은 컬러 지원을 포함하는, 온라인 시스템.

청구항 54.

제 39 항에 있어서,

상기 목표 장치의 상기 성능은 비트 레이트를 포함하는, 온라인 시스템.

청구항 55.

제 39 항에 있어서,

상기 목표 장치의 상기 성능은, 상기 목표 장치가 요청을 전송하기 위해 채택하는 현재 이용가능한 통신 미디어를 포함하는, 온라인 시스템.

청구항 56.

제 55 항에 있어서,

현재 이용가능한 통신 미디어는 무선 통신을 포함하는, 온라인 시스템.

청구항 57.

제 55 항에 있어서,

현재 이용가능한 통신 미디어는 유선 통신을 포함하는, 온라인 시스템.

청구항 58.

제 39 항에 있어서,

상기 성능 모듈은, HTTP 헤더로부터 상기 목표 장치의 상기 성능을 결정하는 능력을 포함하는, 온라인 시스템.

청구항 59.

제 58 항에 있어서,

상기 성능 모듈은, HTTP 사용자-에이전트 헤더로부터 상기 목표 장치의 상기 성능을 결정하는 능력을 포함하는, 온라인 시스템.

청구항 60.

제 39 항에 있어서,

상기 성능 모듈은, 상기 목표 장치에 성능을 질의하는 능력을 포함하는, 온라인 시스템.

청구항 61.

제 39 항에 있어서,

상기 성능 모듈은, 장치 클래스에 기초하여 목표 장치의 성능을 결정하기 위한 지식베이스을 포함하는, 온라인 시스템.

청구항 62.

제 61 항에 있어서,

상기 장치의 상기 성능이 지식베이스에 추가되도록 하기 위해, 인식되지 않은 목표 장치를 기록하는 로그 기록을 더 포함하는, 온라인 시스템.

청구항 63.

제 39 항에 있어서,

상기 특정 포맷은, 목표 장치에서 특정 미디어 객체를 렌더링하는데 적합한 해상도에 기초하여 선택되는, 온라인 시스템.

청구항 64.

제 39 항에 있어서,

상기 특정 포맷은, 목표 장치에서 특정 미디어 객체를 렌더링하는데 적합한 컬러 공간에 기초하여 선택되는, 온라인 시스템.

청구항 65.

제 39 항에 있어서,

상기 특정 포맷은, 목표 장치에서 특정 미디어 객체를 렌더링하는데 적합한 이미지 사이즈에 기초하여 선택되는, 온라인 시스템.

청구항 66.

제 39 항에 있어서,

상기 특정 포맷은, 목표 장치에서 특정 미디어 객체를 렌더링하는데 적합한 비트 레이트에 기초하여 선택되는, 온라인 시스템.

청구항 67.

제 39 항에 있어서,

상기 특정 포맷은, 목표 장치에서 특정 미디어 객체의 복사본을 전송하는데 이용가능한 통신 대역폭에 기초하여 선택되는, 온라인 시스템.

청구항 68.

제 67 항에 있어서,

상기 이용가능한 통신 대역폭은, 적어도 부분적으로, 상기 목표 장치에 대한 장치 클래스에 기초하여 결정되는, 온라인 시스템.

청구항 69.

제 39 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 휴대형 컴퓨팅 장치를 포함하는, 온라인 시스템.

청구항 70.

제 39 항에 있어서,

상기 목표 장치는 오디오 성능을 갖는 휴대형 컴퓨팅 장치를 포함하는, 온라인 시스템.

청구항 71.

제 39 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 셀룰러 전화 장치를 포함하는, 온라인 시스템.

청구항 72.

제 39 항에 있어서,

상기 목표 장치는 오디오 성능을 갖는 셀룰러 전화 장치를 포함하는, 온라인 시스템.

청구항 73.

제 39 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 페이저 장치를 포함하는, 온라인 시스템.

청구항 74.

제 39 항에 있어서,

상기 목표 장치는 디스플레이 성능을 갖는 개인용 컴퓨터를 포함하는, 온라인 시스템.

청구항 75.

제 39 항에 있어서,

상기 목표 장치는 오디오 성능을 갖는 개인용 컴퓨터를 포함하는, 온라인 시스템.

청구항 76.

제 39 항에 있어서,

상기 목표 장치는 WAP (무선 애플리케이션 프로토콜) 지원을 포함하는, 온라인 시스템.

청구항 77.

온라인 시스템에서 클라이언트 장치의 성능을 결정하는 방법으로서,

성능에 관한 정보를 포함하지 않는 목표 장치로부터 오리지널 요청을 수신하는 단계;

상기 목표 장치의 성능을 결정하는 단계;

상기 목표 장치로부터 수신한 상기 오리지널 요청에, 상기 목표 장치의 성능에 관한 정보를 보충하는 단계; 및

상기 보충된 요청을, 상기 오리지널 요청에서 특정된 목적지로 전송하는 단계를 포함하는, 성능 결정 방법.

청구항 78.

제 77 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에 의해 제공된 상기 요청을 조사하는 단계를 포함하는, 성능 결정 방법.

청구항 79.

제 77 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에 의해 제공된 HTTP 헤더를 조사하는 단계를 포함하는, 성능 결정 방법.

청구항 80.

제 79 항에 있어서,

상기 장치에 의해 제공된 상기 HTTP 헤더를 조사하는 단계는 HTTP 사용자-에이전트 헤더를 조사하는 단계를 포함하는, 성능 결정 방법.

청구항 81.

제 77 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는 상기 장치에 성능을 질의하는 단계를 포함하는, 성능 결정 방법.

청구항 82.

제 77 항에 있어서,

상기 목표 장치의 성능을 결정하는 단계는, 상기 목표 장치의 장치 클래스에 기초하여 지식베이스로부터 성능을 결정하는 단계를 포함하는, 성능 결정 방법.

요약

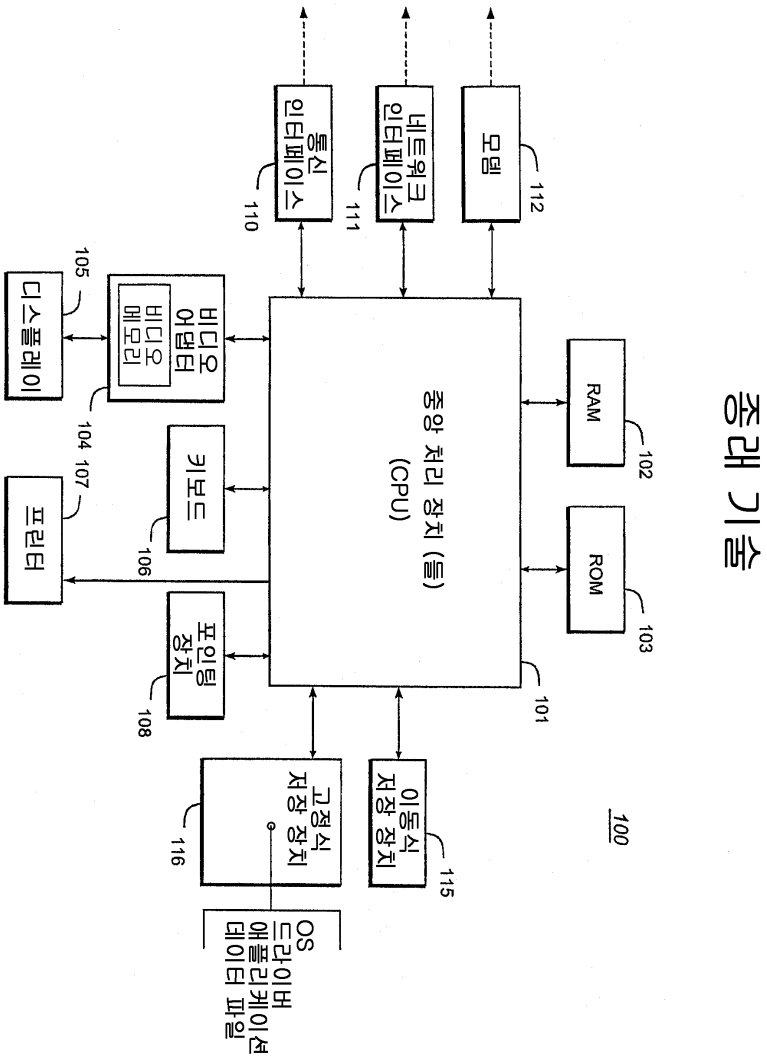
온 더 플라이 미디어 재포맷을, 접속된 클라이언트 장치에 적합한 미디어 컨텐츠의 전송을 가능하게 하는 향상된 클라이언트 탐지 성능과 결합하는, 온라인 미디어 전송 시스템을 설명한다. 이 시스템은 클라이언트 장치로부터 미디어 문서 또는 객체에 대한 요청을 받으며, HTTP 요청으로부터 특정 미디어 객체를 요구하는 클라이언트 장치를 식별하며, 클라이언트 장치의 미디어 출력 성능을 결정하며, 그 성능에 따라 소스 미디어를 재포맷하며, 재포맷된 미디어를 클라이언트 장치로 전송한다. 이 시스템은 다양한 하드웨어 및 소프트웨어 성능의 복수의 다른 클라이언트 장치에 대한 미디어 컨텐츠에 대한 액세스를 제공한다. 이 시스템은 적합한 미디어 컨텐츠를 실질적으로 접속 성능을 갖는 임의의 장치에 전송하는 것을 가능하게 한다.

대표도

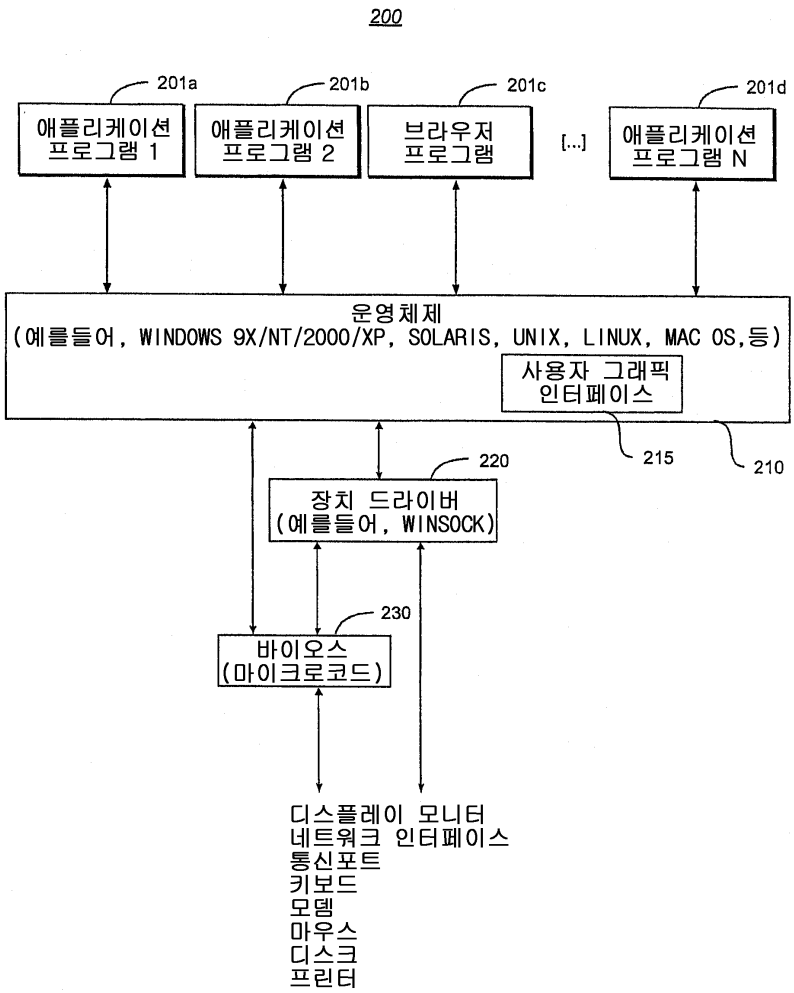
도 3

도면

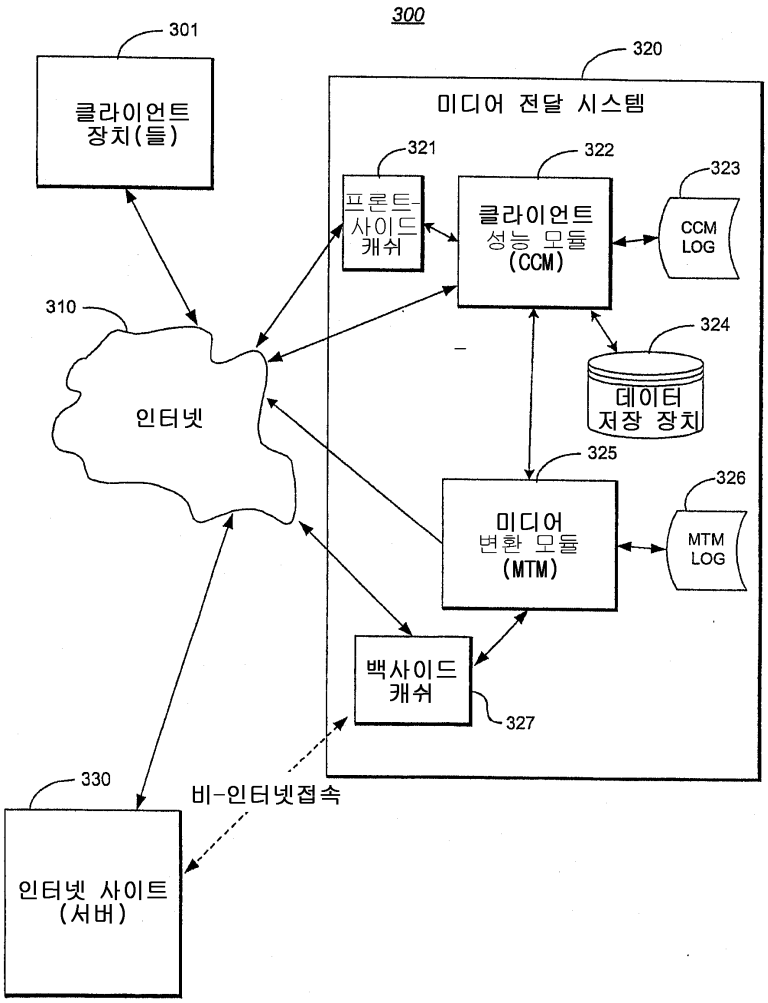
도면1



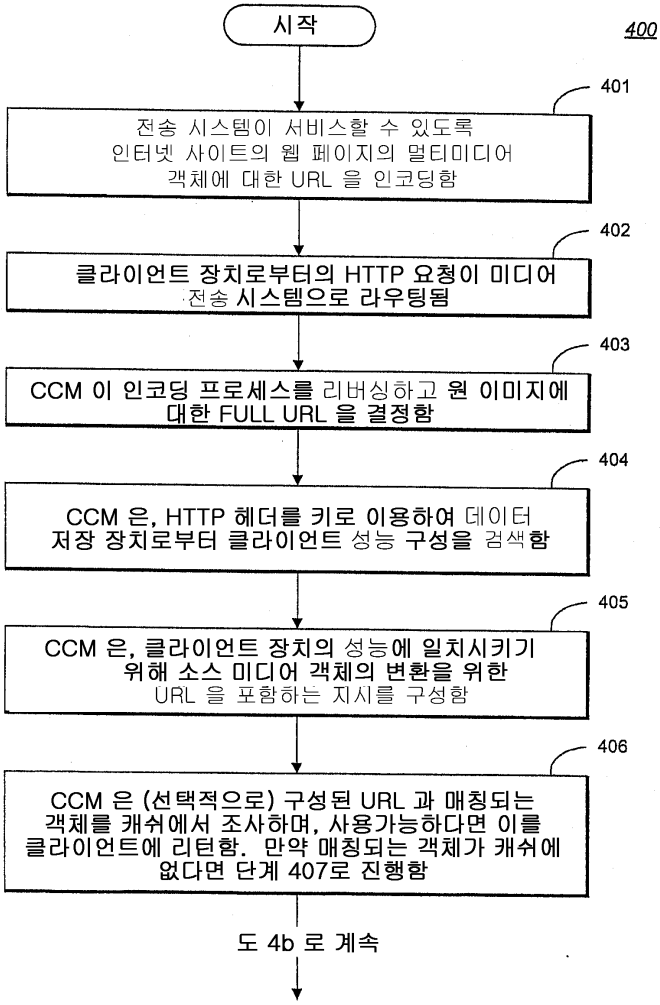
도면2


200
201a
애플리케이션 프로그램 1
201b
애플리케이션 프로그램 2
201c
브라우저 프로그램
[...]
201d
애플리케이션 프로그램 N
운영체제
(예를들어, WINDOWS 9X/NT/2000/XP, SOLARIS, UNIX, LINUX, MAC OS,등)
사용자 그래픽 인터페이스
215
210
220
장치 드라이버
(예를들어, WINSOCK)
230
바이오스
(마이크로코드)
디스플레이 모니터
네트워크 인터페이스
통신포트
키보드
모뎀
마우스
디스크
프린터

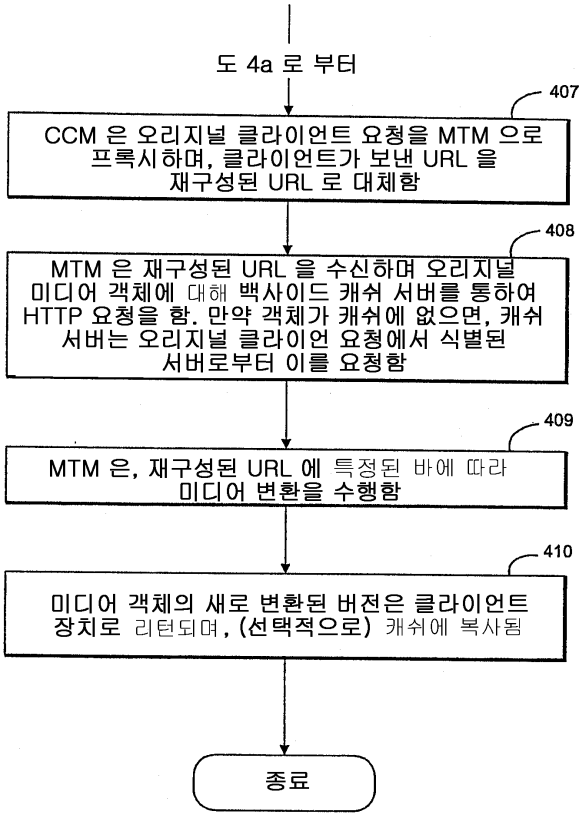
도면3


300
301
클라이언트 장치(들)
320
미디어 전달 시스템
321
프론트-사이드 캐쉬
322
클라이언트 성능 모듈 (CCM)
323
CCM LOG
324
데이터 저장 장치
310
인터넷
325
미디어 변환 모듈 (MTM)
326
MTM LOG
백사이드 캐쉬
327
330
비-인터넷접속
인터넷 사이트 (서버)

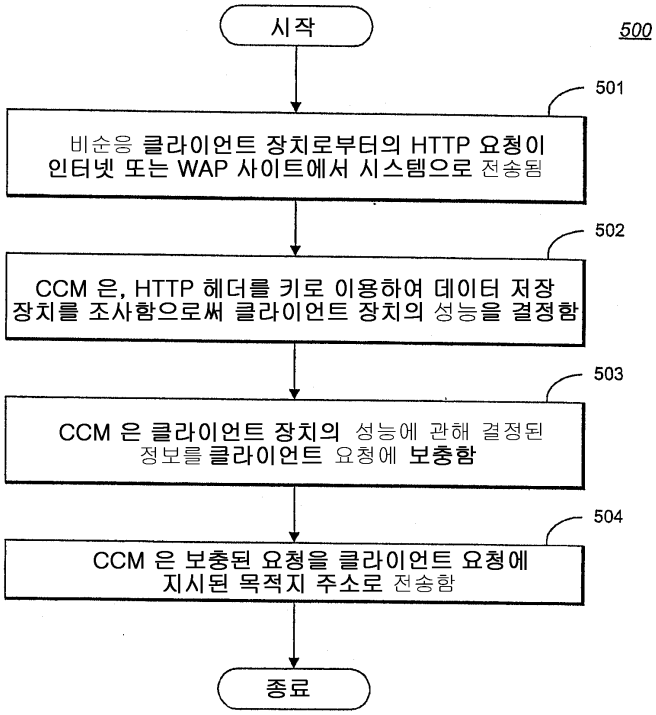
도면4a


시작
400
401
전송 시스템이 서비스할 수 있도록 인터넷 사이트의 웹 페이지의 멀티미디어 객체에 대한 URL 을 인코딩함
402
클라이언트 장치로부터의 HTTP 요청이 미디어 전송 시스템으로 라우팅됨
403
CCM 이 인코딩 프로세스를 리버싱하고 원 이미지에 대한 FULL URL 을 결정함
404
CCM 은, HTTP 헤더를 키로 이용하여 데이터 저장 장치로부터 클라이언트 성능 구성을 검색함
405
CCM 은, 클라이언트 장치의 성능에 일치시키기 위해 소스 미디어 객체의 변환을 위한 URL 을 포함하는 지시를 구성함
406
CCM 은 (선택적으로) 구성된 URL 과 매칭되는 객체를 캐쉬에서 조사하며, 사용가능하다면 이를 클라이언트에 리턴함. 만약 매칭되는 객체가 캐쉬에 없다면 단계 407로 진행함
도 4b 로 계속

도면4b


도 4a 로 부터
407
CCM 은 오리지널 클라이언트 요청을 MTM 으로 프록시하며, 클라이언트가 보낸 URL 을 재구성된 URL 로 대체함
408
MTM 은 재구성된 URL 을 수신하며 오리지널 미디어 객체에 대해 백사이드 캐쉬 서버를 통하여 HTTP 요청을 함. 만약 객체가 캐쉬에 없으면, 캐쉬 서버는 오리지널 클라이언 요청에서 식별된 서버로부터 이를 요청함
409
MTM 은, 재구성된 URL 에 특정된 바에 따라 미디어 변환을 수행함
410
미디어 객체의 새로 변환된 버전은 클라이언트 장치로 리턴되며, (선택적으로) 캐쉬에 복사됨
종료

도면5


시작
500
501
비순응 클라이언트 장치로부터의 HTTP 요청이 인터넷 또는 WAP 사이트에서 시스템으로 전송됨
502
CCM 은, HTTP 헤더를 키로 이용하여 데이터 저장 장치를 조사함으로써 클라이언트 장치의 성능을 결정함
503
CCM 은 클라이언트 장치의 성능에 관해 결정된 정보를 클라이언트 요청에 보충함
504
CCM 은 보충된 요청을 클라이언트 요청에 지시된 목적지 주소로 전송함
종료