(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2012년03월28일
(11) 등록번호 10-1130367
(24) 등록일자 2012년03월19일

(51) 국제특허분류(Int. Cl.)
G06F 9/445 (2006.01) *G06F 15/16* (2006.01)
(21) 출원번호 10-2005-7007318
(22) 출원일자(국제) 2004년07월23일
심사청구일자 2009년07월17일
(85) 번역문제출일자 2005년04월27일
(65) 공개번호 10-2006-0109284
(43) 공개일자 2006년10월19일
(86) 국제출원번호 PCT/US2004/023965
(87) 국제공개번호 WO 2005/060387
국제공개일자 2005년07월07일
(30) 우선권주장
10/737,726 2003년12월15일 미국(US)
(56) 선행기술조사문헌
KR1020000041146 A
KR1020010076025 A
KR1020030016073 A
전체 청구항 수 : 총 23 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
로버츠, 제이슨
미국 02144 매사추세츠주 섬빌리 코리지 애비뉴 2번 103
모하메드, 마즈하
미국 98074 워싱턴주 삼마미시 코트 사우스이스트 127-242
(뒷면에 계속)
(74) 대리인
제일특허법인

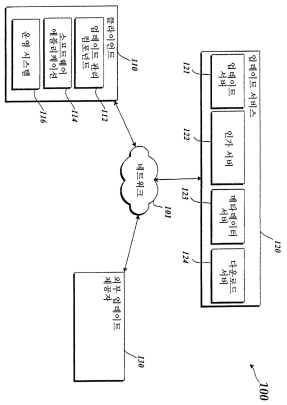
심사관 : 지정훈

(54) 발명의 명칭 **소프트웨어 분산 서비스를 위한 시스템 및 방법**

(57) 요 약

본 발명은 소프트웨어 업데이트를 관리하는 시스템 및 방법에 관한 것이다. 보다 구체적으로, 본 발명은 소프트웨어 업데이트의 선택 및 실행에 요구되는 대역폭 및 처리 자원을 최소화하면서 소프트웨어 업데이트의 선택 및 실행을 용이하게 하는 시스템 및 방법에 관한 것이다. 본 발명의 한 특징에 따르면, 소프트웨어 업데이트 서비스는 서버상에 저장된 소프트웨어 업데이트에의 액세스를 제어한다. 본 발명의 다른 특징에 따르면, 소프트웨어 업데이트 서비스는 클라이언트 머신과 동기화를 이루어 적용 가능한 업데이트를 식별한다.

대 표 도 - 도1



(72) 발명자

위틀, 왈터

미국 98052 워싱턴주 레드몬드 노스이스트 33로드 스트리트 17821

세파드, 마크

미국 98006 워싱턴주 빌리뷰 사우스이스트 42엔디 플레이스 13908

특허청구의 범위

청구항 1

클라이언트 컴퓨팅 장치와 서버 컴퓨팅 장치 사이의 통신 방법으로서,

(a) 동기화 요구를 상기 클라이언트 컴퓨팅 장치가 발행하는 단계 - 상기 동기화 요구는 상기 클라이언트 컴퓨팅 장치가 설치된 소프트웨어 업데이트를 저장하는 경우에 상기 설치된 소프트웨어 업데이트의 식별자를 포함함 - 와,

(b) 상기 동기화 요구에 응답하여 복수의 소프트웨어 업데이트의 명령 컴포넌트들을 상기 클라이언트 컴퓨팅 장치가 수신하는 단계 - 각각의 명령 컴포넌트는 적어도 하나의 적용가능성 규칙(applicability rule)을 포함하고, 상기 적어도 하나의 적용가능성 규칙은 상기 명령 컴포넌트와 연관된 상기 선택된 소프트웨어 업데이트를 설치하기 전에 상기 클라이언트 컴퓨팅 장치에 요구되는 적어도 하나의 조건을 정의함 - 와,

(c) 상기 클라이언트 컴퓨팅 장치 내의 각각의 소프트웨어 업데이트의 상기 명령 컴포넌트를 상기 클라이언트 컴퓨팅 장치가 저장하는 단계와,

(d) 상기 소프트웨어 업데이트들 중 적어도 첫 번째 소프트웨어 업데이트의 로컬화된 데이터 컴포넌트를 상기 각각의 적용가능성 규칙의 상기 적어도 하나의 조건이 충족되었다는 상기 클라이언트 컴퓨팅 장치의 판정에 적어도 부분적으로 기초하여 상기 클라이언트 컴퓨팅 장치가 요구하는 단계와,

(e) 상기 첫 번째 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트를 상기 클라이언트 컴퓨팅 장치가 수신하는 단계 - 각각의 로컬화된 데이터 컴포넌트는 상기 각각의 소프트웨어 업데이트를 설명하는 추가 정보를 포함함 -

를 포함하는 통신 방법.

청구항 2

제1항에 있어서, 상기 방법은,

상기 첫 번째 소프트웨어 업데이트의 상기 명령 컴포넌트가 상기 클라이언트 컴퓨팅 장치에 의한 처리를 위한 추가 소프트웨어 업데이트가 존재하는 것을 나타내는 표시를 제공하는 경우에, 상기 클라이언트 컴퓨팅 장치에 의해, 상기 단계 (a) 내지 (e)를 반복하는 단계를 더 포함하는, 통신 방법.

청구항 3

제1항에 있어서,

상기 클라이언트 컴퓨팅 장치는 상기 로컬화된 데이터 컴포넌트를 수신시 디스플레이 상에 상기 첫 번째 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트의 디스플레이를 생성하는 애플리케이션을 호출하는, 통신 방법.

청구항 4

제1항에 있어서, 상기 방법은,

상기 첫 번째 소프트웨어 업데이트와 연관된 데이터 파일의 통신을 호출하기 위한 커맨드를 상기 클라이언트 컴퓨팅 장치에 의해 수신하는 단계와,

상기 첫 번째 소프트웨어 업데이트와 연관된 상기 데이터 파일의 통신을 위한 커맨드를 상기 클라이언트 컴퓨팅 장치에 의해 발행하는 단계와,

상기 데이터 파일의 다운로드를 상기 클라이언트 컴퓨팅 장치에 의해 수신하는 단계

를 더 포함하는 통신 방법.

청구항 5

제1항에 있어서,

상기 클라이언트 컴퓨팅 장치 내에 각각의 소프트웨어 업데이트의 상기 명령 컴포넌트를 저장하는 단계는,

상기 클라이언트 컴퓨팅 장치가 상기 소프트웨어 업데이트 내에 저장된 상기 적어도 하나의 적용가능성 규칙의 상기 적어도 하나의 조건을 충족하는 적어도 하나의 컴포넌트를 포함하는 경우에, 설치된 소프트웨어 업데이트로서 각각의 소프트웨어 업데이트의 상기 명령 컴포넌트를 저장하는 단계

를 포함하고,

상기 방법은,

상기 클라이언트 컴퓨팅 장치가 상기 설치된 소프트웨어 업데이트를 저장하는 경우에, 상기 설치된 소프트웨어 업데이트의 식별자를 상기 동기화 요구 내로 상기 클라이언트 컴퓨팅 장치에 의해 통합하는 단계

를 더 포함하는 통신 방법.

청구항 6

제1항에 있어서,

상기 클라이언트 컴퓨팅 장치 내에 각각의 소프트웨어 업데이트의 상기 명령 컴포넌트를 저장하는 단계는,

각각의 해당 소프트웨어 업데이트 내에 저장된 상기 적어도 하나의 적용가능성 규칙의 상기 적어도 하나의 조건을 충족하지 않는 것으로 상기 클라이언트 컴퓨팅 장치가 판정하는 경우에, 실패한 소프트웨어 업데이트들로서 상기 클라이언트 컴퓨팅 장치 내의 상기 소프트웨어 업데이트들의 상기 명령 컴포넌트들을 상기 클라이언트 컴퓨팅 장치에 의해 저장하는 단계

를 포함하는 통신 방법.

청구항 7

제6항에 있어서,

상기 방법은,

상기 동기화 요구 내의 임의의 실패한 소프트웨어 업데이트의 식별자를 상기 클라이언트 컴퓨팅 장치에 의해 통합하는 단계

를 더 포함하고,

상기 동기화 요구에 응답하여 상기 소프트웨어 업데이트들의 명령 컴포넌트를 수신하는 단계는,

상기 실패한 소프트웨어 업데이트들과 관련이 없는 소프트웨어 업데이트들의 명령 컴포넌트들만을 수신하는 단계

를 포함하는 통신 방법.

청구항 8

타깃 그룹 내의 클라이언트 프로세싱 장치와 분산 프로세싱 시스템 내의 서버 프로세싱 장치 사이의 통신 방법으로서,

(a) 인가 요구를 상기 클라이언트 프로세싱 장치에 의해서 발행하는 단계 - 상기 인가 요구는 타깃 그룹과 연관된 클라이언트 인가 모듈의 식별자를 포함함 - 와,

(b) 상기 타깃 그룹을 식별하는 서버 쿠키를 상기 클라이언트 프로세싱 장치에 의해서 수신하는 단계와,

(c) 소프트웨어 업데이트들을 위한 업데이트 요구를 상기 클라이언트 프로세싱 장치에 의해 발행하는 단계 - 상기 업데이트 요구는 상기 서버 쿠키를 포함함 - 와,

(d) 상기 타깃 그룹과 연관된 적어도 하나의 소프트웨어 업데이트를 위한 명령 컴포넌트를 상기 클라이언트 프로세싱 장치에 의해 수신하는 단계 - 각각의 명령 컴포넌트는 적어도 하나의 적용가능성 규칙을 포함하고, 상기 적어도 하나의 적용가능성 규칙은 상기 명령 컴포넌트와 연관된 선택된 소프트웨어 업데이트를 설치하기 이전에 상기 클라이언트 프로세싱 장치에 요구되는 적어도 하나의 조건을 정의함 - 와,

(e) 상기 소프트웨어 업데이트들 중 적어도 첫 번째 소프트웨어 업데이트를 위한 로컬화된 데이터 컴포넌트를 상기 각각의 규칙의 상기 적어도 하나의 조건이 적용가능성 충족되었다는 상기 클라이언트 프로세싱 장치에 의한 판정에 적어도 부분적으로 기초하여 상기 클라이언트 프로세싱 장치에 의해 요구하는 단계와,

(f) 상기 첫 번째 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트를 상기 클라이언트 프로세싱 장치에 의해 수신하는 단계 - 상기 로컬화된 데이터 컴포넌트는 상기 첫 번째 소프트웨어 업데이트를 설명하는 추가 정보를 포함함 -

를 포함하는 통신 방법.

청구항 9

제8항에 있어서,

상기 인가 요구는 이전에 저장된 서버 쿠키의 데이터를 포함하고, 상기 서버 프로세싱 장치에 의해서 발행된 상기 서버 쿠키는 상기 이전에 저장된 서버 쿠키의 상기 데이터를 포함하는, 통신 방법.

청구항 10

제8항에 있어서,

상기 서버 프로세싱 장치와 통신하는 서버 인가 모듈이 상기 클라이언트 인가 모듈과 연관된 동일한 타깃 그룹을 식별하는 경우에, 상기 클라이언트 프로세싱 장치가 타깃 그룹과 연관된 것으로 판정되는, 통신 방법.

청구항 11

제8항에 있어서,

상기 서버 쿠키는 만료 시간을 포함하고,

각각의 소프트웨어 업데이트의 상기 명령 컴포넌트를 수신하는 단계는,

상기 서버 쿠키의 상기 만료 시간이 경과되지 않은 경우에만 상기 명령 컴포넌트를 수신하는 단계를 포함하는, 통신 방법.

청구항 12

제11항에 있어서,

상기 서버 쿠키는 상기 만료 시간을 명백한 텍스트 포맷(clear text format) 및 암호화된 포맷으로 저장하고,

상기 서버 쿠키는 적어도 하나의 타깃 그룹을 식별하는 암호화된 데이터를 포함하고,

상기 방법은,

상기 서버 쿠키로부터 상기 만료 시간을 상기 클라이언트 프로세싱 장치에 의해서 획득하는 단계 - 상기 클라이언트 프로세싱 장치는 상기 명백한 텍스트 포맷으로 저장된 상기 만료 시간으로부터 상기 만료 시간을 획득함 - 와,

적어도 하나의 소프트웨어 업데이트에 대한 요구를 상기 클라이언트 프로세싱 장치에 의해서 발행하는 단계- 상기 요구는 상기 서버 쿠키를 포함함 -

를 더 포함하는, 통신 방법.

청구항 13

제8항에 있어서,

상기 소프트웨어 업데이트들 중 하나의 상기 명령 컴포넌트가 추가 소프트웨어 업데이트들이 이용 가능함을 나타내는 경우에는, 상기 방법은,

상기 추가 업데이트들 중 적어도 하나의 선택을 위한 상기 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트를 상기 클라이언트 프로세싱 장치에 의해서 처리하는 단계

를 더 포함하는, 통신 방법.

청구항 14

소프트웨어 업데이트들을 통신하고 관리하는 시스템으로서,

(a) 복수의 소프트웨어 업데이트를 저장하는 제1 데이터베이스 -

상기 데이터베이스내에 저장된 각각의 개별 소프트웨어 업데이트는

(ⅰ) 상기 개별 소프트웨어 업데이트의 설치 이전에 충족되어야 하는 적어도 하나의 조건을 정의하는 적용가능성 규칙을 포함하는 명령 데이터 필드 - 상기 명령 데이터 필드는 상기 개별 소프트웨어 업데이트의 설치를 위해 필요한 임의의 소프트웨어 업데이트를 식별하는 필요조건들의 세트를 포함함 - 와,

(ii) 상기 개별 소프트웨어 업데이트에 연관된 추가 정보를 설명하는 텍스트 데이터를 포함하는 로컬화된 데이터 필드와,

(iii) 상기 개별 소프트웨어 업데이트와 연관된 적어도 하나의 데이터 스트림 파일을 식별하는 데이터 컴포넌트

를 포함함 -와,

(b) 상기 제1 데이터베이스 내에 저장된 소프트웨어 업데이트들의 액세스를 인가하는 인가 서버 컴퓨팅 장치 - 상기 인가 서버 컴퓨팅 장치는 클라이언트 컴퓨팅 장치가 타깃 그룹과 연관되어 있음을 나타내는 인가 모듈을 클라이언트 컴퓨팅 장치가 포함하는 경우에, 상기 클라이언트 컴퓨팅 장치가 상기 개별 소프트웨어 업데이트에 액세스하는 것을 허용하도록 상기 클라이언트 컴퓨팅 장치로 서버 쿠키를 발행하고, 상기 서버 쿠키는 상기 타깃 그룹에 대한 식별자를 포함하며, 상기 서버 쿠키는 상기 서버 쿠키의 저장을 위하여 상기 클라이언트 컴퓨팅 장치로 송신됨 - 와,

(c) 상기 클라이언트 컴퓨팅 장치에 상기 소프트웨어 업데이트들 중 적어도 하나의 상기 로컬화된 데이터 필드와 상기 명령 데이터 필드를 제공하기 위한 메타데이터 서버 컴퓨팅 장치 - 상기 개별 소프트웨어 업데이트가 상기 타깃 그룹과 연관되는 경우에, 각각의 개별 소프트웨어 업데이트의 상기 로컬화된 데이터 필드와 상기 명령 데이터 필드는 상기 클라이언트 컴퓨팅 장치로 송신되고, 상기 메타데이터 서버 컴퓨팅 장치는 상기 제 1 데이터베이스와 통신가능하게 결합되고,

상기 메타데이터 서버 컴퓨팅 장치는

(ⅰ) 동기화 요구를 획득하고 - 상기 동기화 요구는 상기 클라이언트 컴퓨팅 장치가 설치된 소프트웨어 업데이트를 저장하는 경우에 상기 설치된 소프트웨어 업데이트의 식별자를 포함하고, 상기 동기화 요구는 상기 서버 쿠키를 포함함 -,

(ii) 상기 동기화 요구가 적어도 하나의 설치된 소프트웨어 업데이트의 식별자를 포함하는 경우에, 상기 클라이언트 컴퓨팅 장치에의 송신을 위해서 상기 제1 데이터베이스 내에 저장되는 상기 소프트웨어 업데이트들로부터 추가 소프트웨어 업데이트를 선택하고 - 상기 추가 소프트웨어 업데이트의 선택은 상기 추가 소프트웨어 업데이트 내에서 정의되는 필요조건의 충족에 의존하며, 상기 필요조건은 상기 동기화 요구가 상기 적어도 하나의 설치된 소프트웨어 업데이트에 대한 식별자를 포함할 것을 요구하고, 상기 추가 소프트웨어 업데이트는 상기 서버 쿠키에서 식별되는 상기 타깃 그룹과 연관됨 - ,

(iii) 상기 동기화 요구가 적어도 하나의 설치된 소프트웨어 업데이트의 식별자를 포함하지 않는 경우에, 상기 클라이언트 컴퓨팅 장치에의 송신을 위해 상기 제1 데이터베이스 내에 저장된 상기 소프트웨어 업데이트들로부터 제1 레벨 소프트웨어 업데이트를 선택하고 - 상기 제1 레벨 소프트웨어 업데이트는 필요조건을 포함하지 않음 - ,

(iv) 저장을 위하여 상기 클라이언트 컴퓨팅 장치에 상기 선택된 소프트웨어 업데이트의 명령 데이터 필드를 송신하고,

(ⅴ) 상기 선택된 소프트웨어 업데이트와 연관된 상기 로컬화된 데이터 필드를 위한 요구를 획득하고,

(vi) 상기 클라이언트 컴퓨팅 장치에 상기 선택된 소프트웨어 업데이트와 연관된 상기 로컬화된 데이터 필드를 송신하도록 - 상기 로컬화된 데이터 필드는 상기 선택된 소프트웨어 업데이트의 데이터 컴포넌트의 수신

을 위하여 상기 클라이언트 컴퓨팅 장치가 다운로드 요구를 생성할 수 있도록 함 - 동작할 수 있음 -와,

(d) 상기 제1 데이터베이스에 통신가능하게 결합되는 다운로드 서버 컴퓨팅 장치 -

상기 다운로드 서버 컴퓨팅 장치는

(ⅰ) 상기 클라이언트 컴퓨팅 장치로부터 상기 다운로드 요구를 획득하고 - 상기 다운로드 요구는 상기 선택된 소프트웨어 업데이트의 데이터 컴포넌트를 식별함 - ,

(ii) 상기 클라이언트 컴퓨팅 장치에 상기 선택된 소프트웨어 업데이트의 데이터 컴포넌트를 송신하도록 동작할 수 있음 -

를 포함하는 시스템.

청구항 15

제14항에 있어서,

서버 인가 모듈이 상기 타깃 그룹을 식별하는 경우에, 상기 서버 쿠키는 상기 타깃 그룹을 위한 식별자를 포함하는, 시스템.

청구항 16

제14항에 있어서,

상기 서버 쿠키는 만료 시간을 포함하고,

상기 메타데이터 서버 컴퓨팅 장치는

상기 서버의 만료 시간이 경과하였는지 여부를 판정하고,

상기 서버 쿠키의 만료 시간이 경과하였다고 판정되는 경우에, 상기 선택된 소프트웨어 업데이트의 통신을 제한하도록 동작할 수 있는, 시스템

청구항 17

제14항에 있어서,

상기 메타데이터 서버 컴퓨팅 장치는

상기 클라이언트 컴퓨팅 장치가 실패한 소프트웨어 업데이트를 저장하는 경우에, 상기 동기화 요구 내의 실패한 소프트웨어 업데이트의 식별자를 획득하고,

상기 동기화 요구가 상기 실패한 소프트웨어 업데이트의 식별자를 포함하는 것으로 판정되는 경우에, 상기 실패한 소프트웨어 업데이트와 관련이 있는 선택된 소프트웨어 업데이트들의 통신을 제한하도록 동작할 수 있는, 시스템.

청구항 18

제14항에 있어서,

상기 다운로드 서버 컴퓨팅 장치는,

상기 클라이언트 컴퓨팅 장치상에 설치된 하나 이상의 파일을 업데이트하기 위하여 하나 이상의 이용가능한 소프트웨어 업데이트의 선택을 획득하고,

상기 클라이언트 컴퓨팅 장치상에 설치된 파일들 중 하나의 적어도 하나의 버전을 업데이트 하기 위하여 복수의 이용가능한 델타 패치를 식별하는 자동 추출(self-extracting) 파일을 획득하고,

상기 클라이언트 컴퓨팅 장치상에 설치된 하나 이상의 파일의 목록을 획득하고,

상기 선택된 소프트웨어 업데이트를 적용하기 위하여 상기 이용가능한 델타 패치들 중에서 하나 이상의 적용가능한 델타 패치들을 선택하고 - 상기 하나 이상의 적용가능한 패치들을 선택하는 것은 상기 이용 가능한 델타 패치들을 식별하는 상기 자동 추출 파일을 상기 클라이언트 컴퓨팅 장치상에 설치된 상기 하나 이상의 파일의

목록에 매핑하는 것에 상응함 - ,

상기 선택된 델타 패치들 중 적어도 하나에 대한 요구를 전송하도록 동작하는, 시스템.

청구항 19

제14항에 있어서,

상기 적용가능 규칙은 널(null) 값을 가지는, 시스템.

청구항 20

제14항에 있어서,

상기 필요조건들의 세트는 널 값을 가지는, 시스템.

청구항 21

소프트웨어 업데이트를 클라이언트 컴퓨팅 장치에 통신하는 시스템으로서,

복수의 서버 인가 모듈들을 저장하는 제1 컴포넌트와 - 상기 복수의 서버 인가 모듈로부터의 개별 서버 인가 모듈은 타깃 그룹을 식별함 - ,

상기 제1 컴포넌트에 통신가능하게 결합되는 서버 컴퓨터를 포함하고,

상기 서버 컴퓨터는

상기 클라이언트 컴퓨팅 장치로부터 인가 요구를 획득하고 - 상기 인가 요구는 상기 클라이언트 컴퓨팅 장치상에 저장된 클라이언트 인가 모듈의 식별자를 포함함 - ,

상기 클라이언트 컴퓨팅 장치가 상기 타깃 그룹과 연관되었는지 여부를 판정하고 - 상기 클라이언트 인가 모듈이 상기 타깃 그룹과 상기 클라이언트 컴퓨팅 장치가 연관되는 것으로 나타내는 경우에, 상기 클라이언트 컴퓨팅 장치는 상기 타깃 그룹과 연관된 것으로 판정됨 - ,

상기 클라이언트 컴퓨팅 장치가 상기 타깃 그룹과 연관된 것으로 판정되는 경우에, 소프트웨어 업데이트 서비스로부터 상기 클라이언트 컴퓨팅 장치로 서버 쿠키를 통신하고 - 상기 서버 쿠키는 상기 타깃 그룹을 식별함 - ,

상기 소프트웨어 업데이트 서비스상에 저장된 적어도 하나의 소프트웨어 업데이트를 위한 요구를 획득하고 - 상기 요구는 상기 서버 쿠키를 포함함 - ,

상기 적어도 하나의 소프트웨어 업데이트를 위한 상기 요구의 획득에 응답하여, 소프트웨어 업데이트가 상기 서버 쿠키 내에서 식별되는 타깃 그룹과 연관되었는지 여부를 판정하고,

상기 소프트웨어 업데이트가 상기 서버 쿠키 내에서 식별된 상기 타깃 그룹과 연관된 것으로 판정되는 경우에, 상기 소프트웨어 업데이트들의 명령 컴포넌트를 발행하고 - 각각의 명령 컴포넌트는 상기 명령 컴포넌트와 연관된 상기 소프트웨어 업데이트를 설치하기 이전에 상기 클라이언트 컴퓨팅 장치에 요구되는 적어도 하나의 조건을 정의하는 적어도 하나의 적용가능성 규칙을 포함함 - ,

첫 번째 소프트웨어 업데이트의 로컬화된 데이터 컴포넌트를 위한 요구를 수신하고,

상기 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트를 발행하고 - 상기 로컬화된 데이터 컴포넌트는 상기 소프트웨어 업데이트를 설명하는 추가 정보를 포함함 - ,

상기 로컬화된 데이터 컴포넌트를 수신한 이후에 상기 클라이언트 컴퓨팅 장치가 상기 소프트웨어 업데이트를 요구하는 경우에, 상기 소프트웨어 업데이트 서비스로부터 상기 클라이언트 컴퓨팅 장치로 상기 소프트웨어 업데이트를 통신하도록 동작할 수 있는, 시스템.

청구항 22

클라이언트 컴퓨팅 장치와 서버 컴퓨팅 장치 사이의 통신 방법으로서,

(a) 동기화 요구를 상기 서버 컴퓨팅 장치에 의해 수신하는 단계 - 상기 클라이언트 컴퓨팅 장치가 설치된 소프트웨어 업데이트를 저장하는 경우에, 상기 동기화 요구는 상기 설치된 소프트웨어 업데이트의 식별자를 포함함

- 와,

(b) 상기 동기화 요구에 기초하여 상기 클라이언트 컴퓨팅 장치를 위한 적절한 소프트웨어 업데이트들을 상기 서버 컴퓨팅 장치에 의해서 선택하는 단계와,

(c) 상기 동기화 요구에 응답하여 상기 각각의 적절한 소프트웨어 업데이트들의 명령 컴포넌트를 상기 서버 컴퓨팅 장치에 의해 발행하는 단계 - 각각의 명령 컴포넌트는 적어도 하나의 적용가능성 규칙을 포함하고, 상기 적어도 하나의 적용가능성 규칙은 상기 명령 컴포넌트와 연관된 상기 선택된 소트프웨어 업데이트를 설치하기 이전에 상기 클라이언트 컴퓨팅 장치에 요구되는 적어도 하나의 조건을 정의함 - 와,

(d) 상기 각각의 적용가능성 규칙의 상기 적어도 하나의 조건이 충족되었다는 상기 클라이언트 컴퓨팅 장치의 판단에 적어도 부분적으로 기초하여, 상기 적절한 소프트웨어 업데이트들 중 적어도 첫 번째 소프트웨어 업데이트의 로컬화된 데이터 컴포넌트에 대한 요구를 상기 서버 컴퓨팅 장치에 의해 수신하는 단계와,

(e) 상기 첫 번째 소프트웨어 업데이트의 상기 로컬화된 데이터 컴포넌트를 상기 서버 컴퓨팅 장치에 의해 발행하는 단계 - 각각의 로컬화된 데이터 컴포넌트는 상기 각각의 적절한 소프트웨어 업데이트를 설명하는 추가 정보를 포함함 -

를 포함하는, 통신 방법.

청구항 23

제22항에 있어서, 적어도 하나의 소프트웨어 업데이트를 상기 서버 컴퓨팅 장치에 의해 선택하는 단계는,

상기 동기화 요구가 적어도 하나의 소프트웨어 업데이트의 식별자를 포함하는 경우에, 추가 소프트웨어 업데이트를 상기 서버 컴퓨팅 장치에 의해 선택하는 단계 - 상기 추가 소프트웨어 업데이트의 선택은 상기 추가 소프트웨어 업데이트 내에 정의된 필요조건의 충족여부에 의존하고, 상기 필요조건은 상기 동기화 요구가 적어도 하나의 특정 소프트웨어 업데이트를 위한 식별자를 포함하는 것을 요구함 - 와,

상기 동기화 요구가 적어도 하나의 소프트웨어 업데이트의 식별자를 포함하지 않는 경우에, 필요조건으로 널 값을 갖는 제1 레벨 소프트웨어 업데이트를 상기 서버 컴퓨팅 장치에 의해 선택하는 단계

를 포함하는, 통신 방법.

청구항 24

삭제

청구항 25

삭제

명 세 서

기 술 분 야

[0001] 본 발명은 소프트웨어 및 컴퓨터 네트워크에 관한 것으로, 특히, 본 발명은 소프트웨어 업데이트를 관리하고 통신하기 위한 시스템 및 방법에 관한 것이다.

배 경 기 술

[0002] 대부분의 상업적으로 입수 가능한 소프트웨어 제품들은 특징 및/또는 기능들을 수정하거나 업그레이드하기 위한 지속적인 개정 과정을 겪는다. 소프트웨어 제품 또는 컴포넌트의 각각의 개정은 새로운 파일들의 부가 및/또는 기존의 파일들을 새로운 버전의 파일들로 교체하는 것을 필요로 할 수 있다. 일단 판매업자(vendor)가 소프트웨어 제품 문제를 분리하여 그 문제에 대한 해법을 안출하면, 그 판매업자는 그 수정(fix)을 업데이트 안에 반영하고 그 업데이트를 고객들이 널리 이용할 수 있게 하기를 원할 것이다. 소프트웨어 판매업자들은 소프트웨어 업데이트를 가능한 한 신속하고 문제없이 고객들에게 배포하고자 하는 사업 의욕을 갖고 있다.

[0003] 인터넷은 고객들이 소프트웨어 제품들에 대한 최신의 업데이트를 획득하는 중요한 채널을 제공한다. 인터넷 이용의 폭발적인 성장으로 고객들은 소프트웨어 제품들 및 업데이트들이 다운로드를 위해 온라인으로 제공되는 공

통의 기대를 갖게 되었다. 또한 업데이트를 배포하기 위해 인터넷의 이용을 촉진하는 것이 소프트웨어 판매업자들의 관심사이다. 왜냐하면 그렇게 함으로써 그들의 비용이 저감되고 확인된 문제에 대한 수정이 다운로드 가능하게 되자마자 고객들은 그 수정을 획득할 수 있게 되기 때문이다. 인터넷 상의 판매업자 사이트들은 애플리케이션에 대한 업데이트 파일들을 발견하고 위치를 찾아내는 것이 매우 간단하도록 설계될 수 있다. 다운로드의 기술적 측면들은 대부분 사용자의 시야에서 사라졌고, 지금은 일반적으로 운영 시스템(operating system)에 의해 핸들링된다.

[0004] 종래의 방법에서는, 소프트웨어 판매업자가 소프트웨어 업데이트를 다운로드를 위한 "패키지"로서 구성한다. 이 패키지는 일반적으로 셋업 프로그램 및 제품의 업데이트된 파일들 각각이 내장되고 패키지를 작게 하기 위해 압축되어 있는 자동 압축 풀기(self-extracting) 실행 가능 파일이다. 패키지의 사이즈는 일반적으로 각각의 변경된 파일의 압축된 사이즈에 압축 풀기 코드 자체의 사이즈를 더한 합계이다. 실행 시에, 패키지는 내포된 파일들 각각을 임시 위치로 추출해낸 다음, 셋업 프로그램을 개시하여 각각의 파일을 시스템의 디렉토리 내의 적당한 위치에 설치한다. 압축된 형태로 발송되는 파일들은 설치될 때 압축 해제된다. 동일한 위치 내의 동일한 명칭의 임의의 기존 파일은 단순히 교체 파일에 의해 겹쳐쓰기(overwrite)될 것이다.

[0005] 비록 인터넷이 소프트웨어 업데이트의 광범위하고 신속한 배포를 가능케 하지만, 네트워크 송신의 제한된 대역폭이 문제들을 야기하였다. 공통의 소프트웨어 애플리케이션들의 완전한 사이즈들로 인해 업데이트의 다운로드 사이즈가 터무니없이 커지게 되었다. 통상적으로 제품의 갖가지 문제들에 대한 다수의 수정들이 하나의 업데이트로 분류(group)될 것이다. 만일 판매업자가 소프트웨어 제품을 정기적으로 업데이트할 경우, 업데이트 패키지의 다운로드 사이즈는 계속해서 커질 것이다. 왜냐하면 판매업자는 사용자가 이전의 업데이트로부터의 파일들을 이미 갖고 있다는 가정 하에 그러한 파일들을 누락시킬 수는 없기 때문이다. 업데이트 패키지는 다수의 전체 파일들을 결합시키기 때문에, 그것은 파일들이 압축되더라도 상당히 클 수 있다. 때때로, 가장 빠른 모뎀 접속에서조차, 다운로드의 대역폭 효율이 저하된다.

[0006] 종래의 다운로드 프로세스의 시간 소비적 측면은, 물론, 바람직하지 않다. 어떤 경우에, 고객들은 이들 파일 다운로드 동안에 장거리 및 접속 시간 요금을 지불한다. 접속 시간의 단축은 이들 고객에 대한 직접적인 금전상의 비용을 저감시킬 것이다. 일반적으로 판매업자들도 그들이 제공하는 다운로드의 사이즈에 관한 얼마간의 구별 가능한 비용을 가지므로, 사이즈를 감소시키는 것은 그들에게도 직접적인 금전상의 이익이 될 것이다. 다운로드의 사이즈를 감소시킬 경우 그들의 이용 가능한 네트워크 대역폭이 증가하여, 그들은 기존의 네트워크 서버 장비로 더 많은 고객들에게 서비스할 수 있게 될 것이다.

[0007] 또한 대용량 업데이트를 다운로드하는 데 걸리는 긴 시간은 다운로드 프로세스를 갖가지 네트워크 접속 문제들에 대해 취약하게 만든다. 전화선 노이즈, 통화 대기 신호, 및 의도하지 않은 커맨드를 포함하여, 인터넷 세션이 조기에 접속 분리될 수 있는 다수의 이유들이 있다. 일부 인터넷 서비스 제공자들은 접속 시간 제한을 실시하여, 사용자가 단일 세션에서 온라인일 수 있는 시간량을 제한한다. 만일 사용자가 네트워크 접속이 차단될 때 대용량 파일을 다운로드하고 있다면, 그는 다시 시작해야 할 것이다. 대부분의 공통의 운영 시스템 및 파일 전송 프로토콜은 파일 전송이 재개되는 것을 허용하지 않으므로, 임의의 중간 진행 과정이 유실될 것이고, 전송은 다시 시작되어야 할 것이다. 실패의 기회들이 너무 많으므로 많은 사용자들이 업데이트를 온라인으로 획득하는 것이 거의 불가능하다는 것을 안다. 만일 업데이트 패키지의 사이즈가 너무 크면, 사용자들은 결코 그것을 완전히 다운로드할 수 없을 것이다.

[0008] 소프트웨어의 사이즈를 줄이고 대역폭 효율을 증가시키려는 하나의 시도는 델타 패치, 또는 바이너리 패치의 사용과 관련이 있다. 숙련된 당업자라면 델타 패치가 컴퓨터에 의해 실행될 때 기존의 파일을 수정하는 전문화된 소프트웨어 코드에 해당한다는 것을 인식할 것이다. 델타 패치는 전문화된 소프트웨어 코드를 포함하기 때문에, 파일의 각각의 고유 버전에 대해 고유의 델타 패치가 요구된다. 소프트웨어 업데이트에 적용될 때, 소프트웨어 업데이트 서비스는 완전한 업데이트된 파일을 송신하는 대신에 그보다 작은 사이즈의 업데이트 델타 패치를 송신할 수 있다. 그 후 업데이트된 델타 패치는 기존의 파일을 업데이트된 파일로 수정하는 데 이용된다.

[0009] 업데이트 델타 패치는 파일을 업데이트하는 데 필요한 데이터의 양을 잠재적으로 줄일 수 있지만, 델타 패칭에 대한 현재의 접근법은 파일의 다수의 버전들이 존재하는 경우에 적용 가능한 델타 파일들의 선택을 관리하는 데에 결함이 있다. 파일의 각각이 버전에 대해 고유의 델타 패치가 요구되기 때문에, 전형적인 소프트웨어 업데이트 시스템들은 종종 수천은 아니더라도 수백 개의 고유의 델타 패치들이 파일의 각각의 고유 버전에 대응할 것을 요구할 수 있다. 하나의 접근법에서는, 델타 패칭을 지원하는 일부 업데이트 서비스들이 모든 가능한 델

타 패치들을 클라이언트 컴퓨팅 디바이스에 송신한다. 그러나, 이러한 접근법은 전형적으로 가능한 업데이트 델타 패치들의 수가 증가할 때 소프트웨어 업데이트를 실행하는 데 필요한 데이터의 양을 증가시킨다. 따라서, 잠재적으로 적용 가능한 델타 패치들의 수는 완전한 업데이트된 파일과 같은 사이즈로 신속히 커질 수 있다. 다른 접근법에서는, 네트워킹된 업데이트 소프트웨어 서비스가 클라이언트 머신을 스캔하여 각각의 클라이언트 머신에 대해 어느 델타 패치가 적용 가능한지를 선택한다. 이 방법은 송신되는 델타 패치 정보의 양을 줄이기는 하지만, 소프트웨어 업데이트 서비스 상에 클라이언트 머신을 스캔하고 적용 가능한 델타 패치를 선택하기 위한 부가적인 로직을 필요로 한다. 부가적인 로직의 사용은 서비스에 의해 제공되어야 하는 시스템 자원들을 증가시킨다. 또한, 이 접근법은 전형적으로, 종래의 웹 서버에 의해 전형적으로 달성되는, 네트워크 캐싱의 사용을 방해한다.

[0010] 상술한 단점들 외에, 기존의 시스템들은 하드웨어 드라이버와 같은 특정 타입의 소프트웨어 업데이트들을 전달할 수 없다. 이 기술 분야에서 알려진 바와 같이, 하드웨어 드라이버에 적용되는 것들과 같은, 전문화된 소프트웨어 업데이트들은 대규모 배포로 사용자들에게 제공하기가 곤란하다. 왜냐하면 대부분의 전문화된 소프트웨어 업데이트들은 특정한 하드웨어를 갖는 클라이언트 컴퓨터 상에서만 작용할 것이기 때문이다. 대부분의 경우에, 예를 들어, 클라이언트 컴퓨터가 호환성이 없는 하드웨어 드라이브 업그레이드를 획득할 경우에는, 그 드라이브 업그레이드의 설치는 치명적인 오류를 야기할 수 있고, 심지어는 컴퓨터가 작동하지 못하게 할 수도 있다.

[0011] 전술한 내용으로부터 쉽게 이해되겠지만, 서버와 다수의 클라이언트들 사이의 소프트웨어 업데이트의 개선된 통신을 갖는 시스템 및 방법이 요구되고 있다. 또한, 업데이트 서비스들이 전문화된 업데이트를 전달할 때 특정한 타입의 클라이언트들을 타깃으로 정할 수 있게 하는 개선된 메커니즘을 갖는 소프트웨어 업데이트 시스템 및 방법이 요구되고 있다.

발명의 상세한 설명

[0012] 본 발명은 소프트웨어 업데이트를 관리하기 위한 시스템 및 방법에 관한 것이다. 더 구체적으로, 본 발명은 소프트웨어 업데이트를 선택하고 실행하는 데 필요한 대역폭 및 처리 자원들을 최소화하면서 소프트웨어 업데이트의 선택 및 실행을 용이하게 하는 시스템 및 방법에 관한 것이다. 본 발명의 한 특징에 따르면, 소프트웨어 업데이트 서비스는 서버에 저장된 소프트웨어 업데이트에의 액세스를 제어한다. 본 발명의 다른 특징에 따르면, 소프트웨어 업데이트 서비스는 클라이언트 머신과 동기화되어 이용가능한 업데이트를 식별한다.

[0013] 본 발명의 한 특징에 따르면, 분산 프로세싱 시스템에서 클라이언트 컴퓨팅 디바이스 프로세스와 서버 프로세스 사이에 통신하는 방법이 제공된다. 이러한 방법에 따르면, 클라이언트 프로세스는 동기화 요구를 발행한다. 동기화 요구는 설치된 소프트웨어 업데이트를 저장한다. 서버 프로세스는 동기화 요구를 수신하고, 동기화 요구가 적어도 하나의 소프트웨어 업데이트의 식별자를 포함하는 경우에는 부가적인 소프트웨어 업데이트를 선택한다. 부가적인 소프트웨어 업데이트의 선택은 부가적인 소프트웨어 업데이트에서 정의되는 필요 조건의 충족에 의존한다. 이러한 필요 조건은 동기화 요구가 적어도 하나의 특정 소프트웨어 업데이트에 대한 식별자를 포함할 것을 요구한다. 서버 프로세스는 동기화 요구가 적어도 하나의 소프트웨어 업데이트의 식별자를 포함하지 않는 것으로 판정되는 경우에 필요 조건으로서 널(null) 값을 가지는 제1 레벨 소프트웨어 업데이트를 선택한다. 서버 프로세스는 선택된 소프트웨어 업데이트의 명령 컴포넌트를 발행한다. 클라이언트 프로세스는 선택된 소프트웨어 업데이트의 명령 컴포넌트를 수신하여, 클라이언트 컴퓨팅 디바이스가 선택된 소프트웨어 업데이트에 저장된 적용 가능성 규칙의 조건을 충족하는 적어도 하나의 컴포넌트를 포함하는 경우에 선택된 소프트웨어 업데이트의 명령 컴포넌트를 설치된 소프트웨어 업데이트로서 클라이언트 컴퓨팅 디바이스에 저장한다.

[0014] 본 발명의 다른 특징에 따르면, 분산 프로세싱 시스템에서의 클라이언트 컴퓨팅 디바이스 프로세스와 서버 프로세스 사이의 통신 방법이 제공된다. 이러한 방법에 따르면, 클라이언트 프로세스는 인가 요구를 발행한다. 인가 요구는 타깃 그룹과 관련된 클라이언트 인가 모듈의 식별자를 포함한다. 서버 프로세스가 타깃 그룹과 관련된 서버 소프트웨어 모듈의 존재를 검출하는 경우에, 서버 프로세스는 인가 요구를 수신하고 타깃 그룹을 식별하는 서버 쿠키를 발행한다. 클라이언트 프로세스는 타깃 그룹을 식별하는 서버 쿠키를 수신하고, 적어도 하나의 소프트웨어 업데이트에 대한 업데이트 요구를 발행하며, 이러한 요구는 서버 쿠키를 포함한다. 서버 프로세스는 업데이트 요구를 수신하고, 소프트웨어 업데이트가 서버 쿠키에서 식별되는 타깃 그룹과 관련된 경우에 소프트웨어 업데이트를 발행한다. 소프트웨어 업데이트가 서버 쿠키에서 식별되는 타깃 그룹과 관련된 경우에 클라이언트 프로세스는 소프트웨어 업데이트를 수신한다.

[0015] 본 발명의 또 다른 특징에 따르면, 소프트웨어 업데이트를 통신하고 관리하는 시스템이 제공된다. 이러한 시스템은 다수의 소프트웨어 업데이트를 저장하는 제1 데이터베이스를 포함한다. 데이터베이스에 저장된 개별적인

소프트웨어 업데이트는 널 값을 포함하여 적어도 하나의 조건을 정의하는 적용 가능성 규칙을 포함하는 명령 데이터 필드를 포함할 수 있다. 명령 컴포넌트는 널 값을 포함하여, 개별적인 소트프웨어 업데이트의 설치에 요구되는 하나 이상의 소프트웨어 업데이트를 식별하는 필요 조건 세트를 포함한다. 데이터베이스에 저장된 개별적인 소프트웨어 업데이트는 개별적인 소프트웨어 업데이트에 관련된 일반적인 정보를 기술하는 텍스트 데이터를 포함하는 로컬화된 데이터 필드 또한 포함할 수 있을 것이다. 부가적으로, 데이터베이스에 저장된 개별적인 소프트웨어 업데이트는 개별적인 소프트웨어 업데이트와 관련된 적어도 하나의 데이터 스트림을 식별하는 데이터 컴포넌트를 포함할 수 있다.

[0016] 이러한 시스템은 제1 데이터베이스에 저장된 소프트웨어 업데이트에의 액세스를 인가하는 인가 서버를 포함할 수 있다. 클라이언트 컴퓨팅 디바이스가 타깃 그룹과 관련되었다는 것을 나타내는 인가 모듈을 클라이언트 컴퓨팅 디바이스가 포함하는 경우에, 인가 서버는 클라이언트 컴퓨팅 장치가 개별적인 소프트웨어 업데이트에 액세스하는 것을 가능하게 하는 서버 쿠키를 발행한다. 서버 쿠키는 타깃 그룹을 위한 식별자를 포함한다. 부가적으로, 서버 쿠키는 서버 쿠키의 저장을 위하여 클라이언트 컴퓨팅 장치와 통신하게 된다.

[0017] 이러한 시스템은 클라이언트 컴퓨팅 디바이스에 업데이트를 제공하는 메타데이터 서버를 더 포함할 수 있다. 소프트웨어 업데이트가 타깃 그룹과 관련되는 경우에, 소프트웨어 업데이트는 클라이언트 컴퓨팅 디바이스와 통신하게 되며, 메타데이터 서버는 상기 제1 데이터베이스에 통신가능하도록 결합된다. 메타데이터 서버는 동기화 요구를 획득할 수 있다. 클라이언트 컴퓨팅 디바이스가 설치된 소프트웨어 업데이트를 저장하는 경우에, 동기화 요구는 설치된 소프트웨어 업데이트의 식별자를 포함한다. 동기화 요구는 서버 쿠키를 포함한다. 동기화 요구가 적어도 하나의 설치된 소프트웨어 업데이트의 식별자를 포함하는 경우에, 메타데이터 서버 또한 클라이언트 컴퓨팅 디바이스와 통신을 위하여 부가적인 소프트웨어 업데이트를 선택한다. 부가적인 소프트웨어 업데이트의 선택은 부가적인 소프트웨어 업데이트에 정의된 필요 조건의 충족에 의존한다. 필요 조건은 동기화 요구가 적어도 하나의 설치된 소프트웨어 업데이트를 위한 식별자를 포함할 것을 요구한다. 부가적인 소프트웨어 업데이트는 서버 쿠키에서 식별된 타깃 그룹과 관련되어 있다. 동기화 요구가 적어도 하나의 설치된 소프트웨어 업데이트의 식별자를 포함하지 않는 경우에, 메타데이터 서버 또한 클라이언트 컴퓨팅 디바이스에의 통신을 위한 제1 레벨 소프트웨어 업데이트를 선택한다. 제1 레벨 소프트웨어 업데이트는 필요 조건을 포함하지 않는다. 메타데이터 서버는 저장을 위하여 선택된 소프트웨어 업데이트의 명령 컴포넌트를 클라이언트 컴퓨팅 장치에 통신하고, 선택된 소프트웨어 업데이트와 관련된 로컬화된 데이터에 대한 요구를 획득할 수 있다. 또한, 메타데이터 서버는 선택된 소프트웨어 업데이트와 관련된 로컬화된 데이터를 클라이언트 컴퓨팅 디바이스에 통신할 수 있다. 로컬화된 데이터는 클라이언트 컴퓨팅 디바이스가 선택된 소프트웨어 업데이트의 데이터 컴포넌트의 수신에 대한 다운로드 요구를 생성할 수 있도록 한다.

[0018] 이러한 시스템은 제1 데이터베이스에 통신가능하도록 결합되는 다운로드서버를 더 포함할 수 있다. 다운로드 서버는 클라이언트 컴퓨팅 디바이스로부터 다운로드 요구를 획득할 수 있다. 다운로드 요구는 선택된 소프트웨어 업데이트의 데이터 컴포넌트를 식별한다. 다운로드 서버는 선택된 소프트웨어 업데이트의 데이터 컴포넌트를 클라이언트 컴퓨팅 디바이스에 통신할 수 있다.

[0019] 본 발명의 다른 특징에 따르면, 클라이언트 컴퓨팅 디바이스상의 데이터를 업데이트하는 시스템이 제공된다. 이러한 시스템은 다수의 델타 패치를 저장하는 제1 데이터베이스를 포함한다. 이러한 시스템은 상기 제1 데이터베이스 및 클라이언트 컴퓨팅 디바이스에 통신가능하도록 결합되는 서버 컴퓨터 또한 포함한다. 서버는 클라이언트 컴퓨팅 디바이스상에 설치된 하나 이상의 파일을 업데이트하기 위한 하나 이상의 이용가능한 소프트웨어 업데이트의 선택, 적어도 하나의 버전의 설치된 파일 및 클라이언트 컴퓨팅 디바이스상에 설치된 하나 이상의 파일들의 목록을 업데이트하기 위한 다수의 이용가능한 델타 패치를 식별하는 자동 압축 풀기 파일을 획득할 수 있다. 서버는 선택된 소프트웨어 업데이트를 실시하기 위하여 하나 이상의 적용 가능한 델타 패치를 또한 선택할 수 있다. 하나 이상의 적용 가능한 패치들의 선택은 다수의 이용가능한 델타 패치를 식별하는 자동 압축 풀기 파일을 클라이언트 컴퓨팅 디바이스에 설치된 하나 이상의 파일들의 목록에 맵핑하는 것에 상응한다. 서버는 하나 이상의 선택된 델타 패치들에 대한 요구를 더 전송할 수 있다.

[0020] 본 발명의 또 다른 특징에 따르면, 클라이언트 컴퓨팅 디바이스에 소프트웨어 업데이트를 통신하는 시스템이 제공된다. 이러한 시스템은 다수의 서버 인가 모듈을 저장하는 제1 컴포넌트를 포함한다. 개별적인 서버 인가 모듈은 타깃 그룹을 식별한다. 이러한 시스템은 제1 컴포넌트에 통신가능하게 결합되는 서버 컴퓨터를 포함한다. 서버는 클라이언트 컴퓨팅 디바이스로부터 인가 요구를 획득할 수 있다. 인가 요구는 클라이언트 컴퓨팅 디바이스상에 저장되는 클라이언트 인가 모듈의 식별자를 포함한다. 클라이언트 인가 모듈이 클라이언트 컴퓨팅 디바이스가 타깃 그룹과 관련되어 있음을 나타내는 경우에는, 서버는 클라이언트 컴퓨팅 디바이스가 타깃 그

룹과 관련되어 있는지 여부를 또한 판정할 수 있다. 클라이언트 컴퓨팅 디바이스가 타깃 그룹과 관련된 경우에, 서버는 소프트웨어 업데이트 서비스로부터 클라이언트 컴퓨팅 디바이스로 서버 쿠키를 또한 통신할 수 있다. 서버 쿠키는 타깃 그룹을 식별한다. 서버는 소프트웨어 업데이트 서비스상에 저장되는 적어도 하나의 소프트웨어 업데이트에 대한 요구를 획득할 수도 있으며, 이러한 요구는 서버 쿠키를 포함한다. 요구의 획득에 응답하여, 서버는 소프트웨어 업데이트가 서버 쿠키에서 식별되는 타깃 그룹과 관련되었는지 여부를 판정할 수 있다. 부가적으로, 소프트웨어 업데이트가 서버 쿠키에서 식별되는 타깃 그룹과 관련된 경우에, 서버는 소프트웨어 업데이트 서비스로부터 클라이언트 컴퓨팅 디바이스로 소프트웨어 업데이트를 통신할 수 있다.

[0021] 본 발명의 또 다른 특징에 따르면, 개별적인 소프트웨어 업데이트의 데이터 구조가 저장되는 컴퓨터 판독가능 매체가 제공된다. 컴퓨터 판독가능 매체는 널 값을 포함하여, 적어도 하나의 조건을 정의하는 적용 가능성 규칙을 포함하는 명령 데이터 필드를 포함한다. 명령 컴포넌트는 널 값을 포함하여, 개별적인 소프트웨어 업데이트의 설치에 요구되는 하나 이상의 소프트웨어 업데이트를 식별하는 필요 조건의 세트를 포함한다. 컴퓨터 판독가능 매체는 개별적인 소프트웨어 업데이트와 관련된 일반적인 정보를 기술하는 텍스트 데이터를 포함하는 로컬화된 데이터 필드를 포함할 수도 있다. 컴퓨터 판독가능 매체는 개별적인 소프트웨어 업데이트와 관련되는 적어도 하나의 데이터 스트림 파일을 식별하는 데이터 컴포넌트를 더 포함할 수 있다.

실 시 예

[0048] 일반적으로, 본 발명은 소프트웨어 업데이트를 관리하기 위한 시스템 및 방법에 관한 것이다. 더 구체적으로, 본 발명은 소프트웨어 업데이트를 선택하고 실행하는 데 필요한 대역폭 및 처리 자원들을 최소화하면서 소프트웨어 업데이트의 선택 및 실행을 용이하게 하는 시스템 및 방법에 관한 것이다. 본 발명에 따르면 소프트웨어 업데이트는 특정한 소프트웨어 애플리케이션 또는 운영 시스템에 대한 업데이트에 대응할 수 있다. 또한, 소프트웨어 업데이트는 시스템 BIOS와 같은 펌웨어에 대한 소프트웨어 드라이버 또는 업데이트를 포함할 수 있다. 본 발명의 한 특징에 따르면, 소프트웨어 업데이트를 처리하기 위한 시스템 및 컴포넌트 아키텍처가 제공된다. 본 발명의 다른 특징에 따르면, 업데이트 서비스에 의한 클라이언트 머신들의 인가 및 동기화를 용이하게 하는 업데이트 프로토콜 및 인터페이스가 제공된다. 본 발명의 또 다른 특징에 따르면, 델타 패치를 이용하여 설치 컴포넌트 및 갖가지 설치된 파일들을 업데이트하는 방법이 제공된다. 그러나, 숙련된 당업자라면 본 발명의 부가적인 특징들이 본 출원에서 제공될 수도 있다는 것을 인식할 것이다. 또한, 숙련된 당업자라면 각각의 확인된 특징이 개별적으로 또는 공통의 발명의 특징의 일부로서 고려될 수 있다는 것을 인식할 것이다.

[0049] 도 1은 본 발명에 따라서 소프트웨어 업데이트 시스템(100)을 예시하는 블록도이다. 일반적으로, 소프트웨어 업데이트 시스템(100)은 하나 이상의 클라이언트 컴퓨팅 디바이스(110), 업데이트 서비스(120) 및 외부 업데이트 제공자(130)를 포함할 수 있다. 일반적으로, 업데이트 서비스(120)는 클라이언트 컴퓨팅 디바이스(110)에 통신되고 거기에 설치되는 소프트웨어 업데이트들을 저장하고 그것들의 배포를 관리한다. 소프트웨어 업데이트들은 업데이트 서비스(120)에 의해 또는 임의의 수의 외부 업데이트 제공자(130)에 의해 제공될 수 있다.

[0050] 클라이언트 컴퓨팅 디바이스(110), 업데이트 서비스(120), 및 외부 업데이트 제공자(130)는 네트워크(101)를 경유하여 전자적으로 통신한다. 네트워크는 근거리 통신망(LAN) 또는 광역 통신망(WAN) 또는 인터넷과 같은 보다 큰 네트워크일 수 있다. 일반적으로 알려진 소프트웨어를 이용하여, 소프트웨어 업데이트 시스템(100)은 클라이언트 컴퓨팅 디바이스(110)와 업데이트 서비스(120)의 서버들(121, 122, 123 및 124) 사이에 문서, 커맨드, 또는 다른 알려진 타입의 정보를 교환하도록 구성될 수 있다. 숙련된 당업자나 다른 사람들이 인식하는 바와 같이, 도 1에 도시된 소프트웨어 업데이트 서비스(100)는 본 발명을 구현하기 위한 하나의 적당한 시스템의 간략화된 예일 뿐이고 본 발명은 이 예에 한정되지 않는다.

[0051] 아래에서 더 상세히 설명하겠지만, 일 실시예에서 업데이트 서비스(120)는 다수의 서버들을 포함한다. 도 1에 도시된 바와 같이, 업데이트 서비스(120)는 업데이트 서비스(120)의 전체적인 처리를 관리하고 업데이트 서비스(120)의 서버들(121, 122, 123 및 124)의 처리들을 코디네이트(coordinate)하기 위한 업데이트 서버(121)를 포함한다. 인가 서버(122)는 클라이언트의 요구에 따라 인가 쿠키를 생성하고, 인가 쿠키는 서버 쿠키를 생성하기 위해 이용되고, 서버 쿠키는 클라이언트 컴퓨터들이 업데이트 서비스(120)에 의해 제공된 업데이트들에 액세스할 수 있게 한다. 메타데이터 서버(123)는 업데이트 서비스(120)에 의해 제공된 업데이트들에 관한 일반 정보를 제공한다. 메타데이터 서버(123)는 본 발명의 시스템이 특정 타입의 클라이언트 컴퓨터 또는 특정 그룹의 클라이언트 컴퓨터들에 대한 특정 업데이트들을 식별할 수 있게 한다. 다운로드 서버(124)는 업데이트 서버(120)에 의해 제공된 소프트웨어 업데이트들과 관련된 데이터 파일들을 전달하기 위한 하나 이상의 소프트웨어 컴포넌트들을 제공한다.

[0052] 외부 업데이트 제공자(130)는 소프트웨어 업데이트들을 배포하는 하나 이상의 서버를 포함할 수 있다. 외부 업데이트 제공자(130)는 클라이언트 컴퓨터들의 그룹들에 배포될 소프트웨어, 소프트웨어 업데이트들, 또는 다른 데이터를 제공하는 엔티티와 관련될 수 있다. 예를 들면, 외부 업데이트 제공자(130)는 하나 이상의 소프트웨어 애플리케이션에 대한 업데이트들을 배포하기 위해 업데이트 서비스(120)를 이용하기를 원하는 제3자 소프트웨어 개발자와 관련될 수 있다. 다른 예에서, 외부 업데이트 제공자(130)는 소프트웨어 업데이트 시스템(100)과 관련될 수 있다.

[0053] 클라이언트 컴퓨팅 디바이스(110)는 소프트웨어 애플리케이션들(114)을 저장하고 실행하는 임의의 컴퓨팅 디바이스일 수 있다. 클라이언트 컴퓨팅 디바이스(110)는 퍼스널 컴퓨터(PC), 개인 휴대 정보 단말기(PDA), 이동 전화, 양방향 무선 호출기(two-way pagers) 등을 포함하는(그러나 이들에 한정되지 않는) 다수의 서로 다른 컴퓨터 제품들 중 어느 하나로부터 형성될 수 있다. 숙련된 당업자나 다른 사람들이 인식하는 바와 같이, 클라이언트 컴퓨팅 디바이스(110)의 아키텍처는 임의의 적당한 형태를 취할 수 있다. 예를 들면, 클라이언트 컴퓨팅 디바이스(110)는 네트워크(101)와의 통신을 제공하는 네트워크 인터페이스를 포함할 수 있다. 네트워크 인터페이스는 임의의 유선 또는 무선 네트워크 접속과 함께 사용하기 위해 구성될 수 있고, TCP/IP 프로토콜과 같은 임의의 적당한 통신 프로토콜과 함께 사용될 수 있다. 게다가, 클라이언트 컴퓨팅 디바이스(110)는 처리 유닛, 디스플레이, 및 메모리 유닛을 포함할 수 있다. 메모리 유닛은 운영 시스템(116)과 같은 클라이언트 컴퓨팅 디바이스(110)를 운영하는 데 필요한 프로그램 코드를 저장할 수 있다. 게다가, 메모리 유닛은 본 발명의 처리들을 제어하고 실행하기 위한 업데이트 관리 컴포넌트(112)를 저장한다.

[0054] 소프트웨어 업데이트 시스템(100)은, 실행될 때, 본 발명을 구현하는 소프트웨어 프로그램들을 저장한다. 실행될 때, 소프트웨어 업데이트 시스템(100)은 소프트웨어 업데이트들을 저장하고, 관리하고, 선택적으로 통신한다. 아래에서 더 충분히 설명하겠지만, 여러 이점들 중에서도 특히, 본 발명은 소프트웨어 업데이트들을 수신하기에 적합한 클라이언트 컴퓨팅 디바이스들의 타깃 그룹들을 정의하고 선택하기 위한 메커니즘을 제공한다. 본 발명은 또한 소프트웨어 업데이트들과 관련된 데이터 파일들을 다운로드하기 위한 개선된 메커니즘을 제공한다.

[0055] 본 발명을 예시할 목적으로, 본 발명의 실시예에 대한 상세한 설명을 제공한다. 실시예를 설명하는 중에, 소프트웨어 애플리케이션의 특정 업그레이드, 예를 들면, 미디어 플레이어 버전 6.0에서 미디어 플레이어 버전 7.0으로의 업그레이드를 가리킬 수 있는 소프트웨어 업데이트들을 참조한다. 숙련된 당업자라면 인식하겠지만, 그러한 소프트웨어 업데이트는 소프트웨어 업데이트와 관련된 다수의 데이터 파일들의 통신 및 설치를 포함할 수 있다. 따라서, 본 발명을 예시할 목적으로, 소프트웨어 업데이트와 소프트웨어 업데이트를 포함하는 개개의 데이터 파일 사이에 구별이 행해진다.

[0056] 이제 도 2 내지 도 6과 관련하여, 클라이언트 컴퓨팅 디바이스(110) 상의 하나 이상의 파일들을 업데이트하기 위한 소프트웨어 업데이트 시스템(100)의 컴포넌트들 사이의 예시적인 상호 작용에 대하여 설명하겠다. 도 2와 관련하여, 하나 이상의 외부 업데이트 제공자들(130)에 의한 소프트웨어 업데이트 정보의 송신에 의해 소프트웨어 업데이트 서비스가 개시된다. 상술한 바와 같이, 외부 업데이트 제공자들(130)은 소프트웨어 업데이트 시스템(100)과 관련될 수 있다. 대안적으로, 소프트웨어 업데이트 정보는 제3자 외부 업데이트 제공자들(130)에 의해 송신될 수 있다. 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트 정보는, 파일을 업데이트하기 위해 사용되는 소프트웨어 코드, 파일을 교체하기 위해 사용되는 소프트웨어 코드, 소프트웨어 업데이트들의 적용 가능성을 판정하기 위한 갖가지 규칙, 및/또는 소프트웨어 업데이트를 설명하는 디스플레이 정보를 포함할 수 있다. 소프트웨어 업데이트 정보의 송신은 언제라도 완료될 수 있고 다른 예시된 소프트웨어 업데이트 컴포넌트 상호 작용의 개시와 동시이어야 할 필요는 없다.

[0057] 외부 업데이트 제공자(130)로부터 소프트웨어 업데이트 정보를 수신하면, 업데이트 서비스(120)는 업데이트 정보의 송신을 용이하게 하는 하나 이상의 데이터를 생성한다. 이 데이터는 파일의 서로 다른 버전들을 업데이트하기 위한 소프트웨어 델타 패치들의 세트에 대응하는 패치 저장 파일을 포함할 수 있다. 이 데이터는 또한 패치 저장 파일에서 발견된 대응하는 델타에 특정 파일 버전들을 맵핑하는 인덱스에 대응하는 패치 저장 매니페스트를 포함할 수 있다. 이 데이터는 또한 업데이트 에이전트가 특정 소프트웨어 업데이트 데이터를 요구하고 설치하기 위해 이용할 정보에 대응하는 자동 압축 풀기 파일을 포함할 수 있으며, 이에 대해서는 아래에서 더 상세히 설명될 것이다. 숙련된 당업자라면 패치 저장 파일, 패치 저장 매니페스트, 및 자동 압축 풀기 파일들의 생성이 언제라도 완료될 수 있고 다른 예시된 소프트웨어 업데이트 컴포넌트 상호 작용들과 동시이어야 할 필요는 없다는 것을 인식할 것이다.

[0058] 클라이언트들에 소프트웨어 업데이트 정보의 송신을 개시하기 위해, 클라이언트 컴퓨팅 디바이스(110)는 업데이트 서비스(120)에 인증 요구(authentication request)를 개시한다. 본 발명의 예시적인 실시예에서, 인증 요구는 클라이언트 컴퓨팅 디바이스(110)와 업데이트 서비스(120) 사이의 업데이트 프로토콜 상호 작용에 대응하며, 이에 대해서는 아래에서 더 상세히 설명될 것이다. 인증을 완료하면, 업데이트 서비스(120)는 클라이언트 컴퓨팅 디바이스(110)에 인증 쿠키를 송신한다. 이제 도 3과 관련하여, 인증된 클라이언트 컴퓨팅 디바이스(110)는 그 후에 업데이트 서비스(120)와의 이용 가능한 업데이트들의 동기화를 개시한다. 본 발명의 예시적인 실시예에서, 동기화 요구는 또한 클라이언트 컴퓨팅 디바이스(110)와 업데이트 서비스(120) 사이의 업데이트 프로토콜 상호 작용에 대응하며, 이에 대해서는 아래에서 더 상세히 설명될 것이다. 동기화를 완료하면, 클라이언트 컴퓨팅 디바이스(110)는 모든 적용 가능한 소프트웨어 업데이트들의 정보 및 업데이트들을 설명하는 정보를 수신한다. 그러나, 본 발명의 예시적인 실시예에서는, 그 업데이트를 인스턴스화(instantiate)하는 어떠한 소프트웨어 코드도 다운로드되지 않았다.

[0059] 계속해서 도 3과 관련하여, 업데이트 처리 중의 어느 때에, 설치될 업데이트들의 선택이 수신된다. 본 발명의 예시적인 실시예에서, 사용자에게 동기화 중에 수신된 소프트웨어 업데이트 정보가 제공되고 적당한 업데이트를 선택하도록 요구된다. 대안적으로, 클라이언트 컴퓨팅 디바이스(110)는 모든 적용 가능한 소프트웨어 업데이트들을 자동적으로 선택하도록 구성될 수 있다. 또한, 클라이언트 컴퓨팅 디바이스(110)는 이용 가능한 소프트웨어 업데이트들의 서브세트를 자동적으로 선택할 수 있게 하는 몇몇 규칙들을 가질 수도 있다. 또한, 사용자가 예컨대 인터넷 웹 페이지를 통하여 업데이트 서비스(120)와 통신함으로써 업데이트의 선택을 개시할 수도 있다.

[0060] 이제 도 4와 관련하여, 업데이트 관리 컴포넌트(112)는, 업데이트 에이전트가 이미 존재하지 않는다면, 클라이언트 컴퓨팅 디바이스(110) 상에 업데이트 에이전트(118)를 인스턴스화한다 그 후에 업데이트 에이전트(118)는 자동 압축 풀기 파일과 같은 소프트웨어 업데이트 정보 패키지의 송신을 요구한다. 업데이트 에이전트(118)는 자동 압축 풀기 파일을 수신하여 설치 관리자(installer)에 대해 임의의 업데이트를 수행한다. 이에 대해서는 아래에서 설명하겠다. 또한, 업데이트 에이전트(118)는 업데이트 서비스(120)로부터 임의의 누락되거나 훼손된 정보를 요구할 수 있다.

[0061] 이제 도 5와 관련하여, 일단 업데이트 에이전트(118)가 소프트웨어 업데이트 정보 패키지를 수신하면, 업데이트 에이전트(118)는 클라이언트 컴퓨팅 디바이스(110) 상에 설치되는 파일들의 목록(inventory)을 수행한다. 이 목록과 소프트웨어 업데이트 정보 패키지의 비교에 기초하여, 업데이트 에이전트(118)는 선택된 업데이트를 완료하기 위해 어느 델타 패치 또는 다른 업데이트 정보가 요구될 것인지를 판정한다. 그 후에 업데이트 에이전트(118)는 특정 데이터 업데이트에 대한 요구를 송신한다. 본 발명의 일 실시예에서, 소프트웨어 업데이트에 대한 요구는 직접 네트워크 접속을 통하여 송신되는 직접 요구에 대응할 수 있고, 이것은 수동 업데이트(manual update)로 불릴 것이다. 본 발명의 다른 실시예에서, 소프트웨어 업데이트에 대한 요구는 명백한 사용자 동작을 요구하지 않고 송신되는 배경 요구(background request)일 수 있다. 이 실시예는 자동 업데이트로 불릴 것이다.

[0062] 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트가 델타 패치에 대응할 경우, 업데이트 에이전트(118)는 패치 저장 매니페스트에 의해 식별된 특정 델타 패치를 식별하는 요구를 업데이트 서비스(120)에 송신한다. 대안적으로, 델타 패치가 이용 가능하지 않은 경우 또는 몇몇 델타 패치들이 실패한 경우에, 업데이트 에이전트(118)는 고장 조치 절차(fallback procedure)를 개시할 수 있다. 고장 조치 절차는 패치 저장 파일로부터 전체 업데이트된 파일의 완전한 카피의 송신에 대한 요구를 포함할 수 있다. 고장 조치 절차는 또한 자체 완비된(self-contained) 패키지 내로부터 전체 업데이트된 파일의 완전한 카피의 송신에 대한 요구를 포함할 수 있다.

[0063] 본 발명의 예시적인 실시예에서, 업데이트 서비스(120)의 다운로드 서버(124)는 업데이트 에이전트(118)로부터의 소프트웨어 업데이트 요구를 직접 처리할 수 있다. 대안적으로, 요구는 업데이트 서비스(120)로부터 요구된 업데이트 델타 패치들을 수신한 종래의 웹 서버들과 같은 임의의 수의 부가적인 다운로드 서버들에 의해 처리될 수도 있다. 예를 들면, 회사가 클라이언트 머신들을 업데이트하기 위해 내부 서버를 이용할 수도 있다. 또한, 요구는 외부 다운로드 서버들에 의해 처리될 수 있고, 여기서는 이전 요구들의 처리 중에 업데이트 델타 패치들의 일부 또는 전부가 캐싱된다. 따라서, 이 실시예에서, 다운로드는 하이퍼 텍스트 전송 프로토콜(HTTP) 데이터 요구에 대해 서비스할 수 있는 다수의 부가적인 다운로드 서버들에 배포될 수 있다.

[0064] 도 6과 관련하여, 일단 소프트웨어 업데이트 정보가 수신되면, 업데이트 에이전트(118)는 델타 패치를 설치된 파일과 병합하여 업데이트된 파일을 생성한다. 또한, 업데이트 에이전트(118)는 머저(merger)가 적당한 파일을 성공적으로 업데이트하였는지를 유효성 확인(validate)할 수 있다. 상술한 바와 같이, 델타 패치가 유효성 확

인될 수 없다면, 업데이트 에이전트(118)는 델타 패치를 다시 요구하거나 또는 다수의 실패 후에 전체 업데이트된 파일을 요구할 수 있다. 일단 업데이트 에이전트(118)가 유효성 확인되고 업데이트된 파일을 획득하면, 그 파일은 클라이언트 컴퓨팅 디바이스(110) 상에 설치된다.

[0065] 도 7은 본 발명에 따라서 클라이언트 컴퓨팅 디바이스(110)와 소프트웨어 업데이트 서비스(120) 사이의 상호 작용을 예시하는 소프트웨어 업데이트 처리 루틴(700)의 흐름도이다. 블록(702)에서, 소프트웨어 업데이트 서비스(120)는 클라이언트 컴퓨터(110)에의 액세스를 인가한다. 본 발명의 예시적인 실시예에서, 클라이언트 컴퓨터에의 액세스의 인가는 특정 그룹의 컴퓨터들과 관련되어 있는 소프트웨어 업데이트들에의 액세스를 허용하기 위한 서버 발행 쿠키(server-issued cookie)의 생성을 포함할 수 있다. 인가 처리의 더 상세한 내용에 대해서는 도 8과 관련하여 설명하겠다.

[0066] 블록(704)에서, 클라이언트 컴퓨터(110)와 소프트웨어 업데이트 서비스(120)는 업데이트 정보를 동기화한다. 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트 서비스(120)는 클라이언트 컴퓨팅 디바이스(110)에 특정 소프트웨어 업데이트를 설명하는 메타데이터를 송신한다. 이 메타데이터는 사용자가 설치하기 위한 하나 이상의 업데이트를 선택할 수 있도록 이용 가능한 소프트웨어 업데이트들을 설명하는 정보를 포함한다. 동기화 처리의 더 상세한 내용에 대해서는 아래에서 도 9 및 10과 관련하여 설명될 것이다. 블록(706)에서, 클라이언트 컴퓨팅 디바이스(110)는 다운로드하기 위한 적용 가능한 업데이트들의 선택을 획득한다. 본 발명의 예시적인 실시예에서, 적용 가능한 업데이트들의 선택은 사용자 선택을 용이하게 하는 다수의 고유의 사용자 인터페이스들의 이용에 대응할 수 있다. 사용자 인터페이스의 선택에 대해서는 도 11과 관련하여 더 상세히 설명될 것이다.

[0067] 블록(708)에서, 클라이언트 컴퓨팅 디바이스(110)는 적용 가능한 소프트웨어 업데이트의 사용자 선택을 처리하고 소프트웨어 업데이트 서비스(120)와 인터페이스 접속하여 특정 업데이트 정보를 요구한다. 본 발명의 예시적인 실시예에서, 클라이언트 컴퓨팅 디바이스(110)는 하나 이상의 적용 가능한 업데이트 델타 패치들을 선택 및 요구한다. 그 후에 클라이언트 컴퓨팅 디바이스(110) 상의 업데이트 에이전트(118)는 요구된 데이터를 처리하여 선택된 소프트웨어 업데이트를 실행한다. 블록(710)에서, 루틴(700)이 종료된다.

[0068] 도 8과 관련하여, 클라이언트 컴퓨팅 디바이스(110)에의 액세스를 인가하고 블록(702)에 대응하는 프로토콜 도면(800)에 대하여 설명하겠다. 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트 서비스(120)는 확장 가능한 타깃 지정 메커니즘(extensible targeting mechanism)을 이용하여 업데이트 및 다른 소프트웨어에의 클라이언트 컴퓨팅 디바이스(110) 액세스를 제어한다. 소프트웨어 업데이트 서비스(120)는 특정 소프트웨어 업데이트들을 클라이언트 컴퓨팅 디바이스들(110)의 하나 이상의 타깃 그룹과 관련시키는 메커니즘을 통합한다. 예를 들면, 소프트웨어 업데이트 서비스(120)는 특정 하드웨어 드라이버 업데이트의 액세스를 특정 하드웨어 디바이스를 갖는 특정 브랜드의 클라이언트 컴퓨팅 디바이스들(110)에 한정할 수 있다. 그러한 예에서, 소프트웨어 업데이트 서비스(120)는 특정 브랜드 명칭 및 특정 하드웨어 디바이스를 갖는 클라이언트 컴퓨팅 디바이스들(100)의 타깃 그룹을 정의하고 특정 소프트웨어 다운로드의 송신을 그 타깃 그룹에 한정할 수 있다.

[0069] 본 발명의 예시적인 실시예에서, 확장 가능한 타깃 지정 메커니즘은 하나 이상의 타깃 그룹들에 대한 클라이언트 컴퓨팅 디바이스의 멤버십을 정의하는 소프트웨어 컴포넌트들("인가 플러그인들"(authorization plug-ins))의 이용에 의해 용이하게 된다. 클라이언트 컴퓨팅 디바이스(110) 상의 인가 플러그인의 존재는 그 클라이언트 컴퓨팅 디바이스가 그 인가 플러그인의 특정 타깃 그룹에 속하는지를 정의한다. 예를 들어, 타깃 그룹은 특정 소프트웨어 애플리케이션에 대한 유효한 제품 ID(product identification)("PID")를 갖는 모든 컴퓨터들을 포함할 수 있다. 그러한 예에서는, 도 8과 관련하여 아래에서 더 상세히 설명하겠지만, 클라이언트 컴퓨팅 디바이스의 메모리 모듈로부터 PID를 판독하고 획득된 PID를 대응하는 PID 서버 플러그인(829)에 전달하기 위한 인가 플러그인(826)이 클라이언트에 설치될 수 있다. 본 명세서에서 PID 밸리데이터(validator)(829)라고도 불리는 대응하는 PID 플러그인은 하나 이상의 방법을 이용하여 수신된 PID가 유효한지를 판정한다. 일단 클라이언트 컴퓨팅 디바이스(110) 상에 저장된 PID가 유효한 것으로 판정되면, 서버는 클라이언트 컴퓨팅 디바이스(110)가 유효한 PID를 갖는 타깃 그룹의 멤버임을 나타내는 서버 쿠키를 생성한다. 다른 실시예에서, 타깃 그룹은 베타 테스트 컴퓨터로 지정되어 있는 클라이언트 컴퓨팅 디바이스들을 포함할 수 있다.

[0070] 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트 서비스(120)의 인가 서버(122)는, 그 인가 서버가 인식할 클라이언트 컴퓨팅 디바이스들의 타깃 그룹들의 세트를 정의하는 다수의 서버 인가 플러그인들을 포함한다. 각각의 서버 인가 플러그인은 클라이언트 컴퓨팅 디바이스(110) 상에 저장된 대응하는 클라이언트 인가 플러그인과 데이터 통신하기 위한 컴포넌트들을 포함한다. 마찬가지로, 각각의 클라이언트 컴퓨팅 디바이스(110)는 그

클라이언트가 속하는 타깃 그룹을 식별하는 하나 이상의 클라이언트 인가 플러그인들을 포함한다. 본 발명의 예시적인 실시예에서, 클라이언트 인가 플러그인들은 예컨대 운영 시스템의 설치 또는 업그레이드와 같은 소프트웨어 애플리케이션의 설치 또는 업그레이드 동안에 각각의 클라이언트 컴퓨팅 디바이스에 설치될 수 있다. 또한, 서버 인가 플러그인들은 소프트웨어 업데이트들에의 액세스를 제어하기를 원하는 관리자에 의해 동적으로 설치 또는 제거될 수 있다. 클라이언트 컴퓨팅 디바이스(110) 및 인가 서버(122) 상에 저장된 인가 플러그인들은 실제 소프트웨어 플러그인일 수 있고, 또는 그 인가 플러그인들은 동적으로 링크된 라이브러리에 하드 코드될 수 있다.

[0071] 도 8에 도시된 바와 같이, 인가 서버(122)는 3개의 예시적인 서버 인가 플러그인: (1) 모든 컴퓨터들을 포함하는 타깃 그룹(이하에서는 "모든 컴퓨터 타깃 그룹"이라 한다)을 정의하는 제1 서버 인가 플러그인(828)과; (2) 유효한 PID를 갖는 컴퓨터들을 포함하는 타깃 그룹(이하에서는 "PID 타깃 그룹"이라 한다)을 정의하는 제2 서버 인가 플러그인(829)과; (3) 베타 테스트 컴퓨터들을 포함하는 타깃 그룹(이하에서는 "베타 타깃 그룹"이라 한다)을 정의하는 제3 서버 인가 플러그인(830)을 포함한다. 또한 도 8에 도시된 바와 같이, 클라이언트 컴퓨팅 디바이스(110)는 2개의 클라이언트 인가 플러그인: (1) 클라이언트 컴퓨팅 디바이스(110)가 모든 컴퓨터 타깃 그룹의 멤버임을 나타내는 제1 클라이언트 인가 플러그인(825)과; (2) 클라이언트 컴퓨팅 디바이스(110)가 PID 타깃 그룹의 멤버임을 나타내는 제2 클라이언트 인가 플러그인(826)을 포함한다. 본 예에서는, 클라이언트 컴퓨팅 디바이스(110)는 그것이 베타 타깃 그룹의 멤버임을 나타내는 인가 플러그인은 포함하지 않는다. 숙련된 당업자라면 인식하겠지만, 각각의 인가 플러그인(825 및 826)은 클라이언트 컴퓨팅 디바이스(110) 상에서 유효성 확인 처리를 지원하기 위한 하나 이상의 기능들을 수행하도록 구성될 수 있다. 이를테면, 제2 클라이언트 인가 플러그인(826)은 설치된 소프트웨어 애플리케이션에 대한 PID를 검증 및 획득하기 위해 클라이언트 컴퓨팅 디바이스(110)의 메모리를 검사하도록 구성될 수 있다.

[0072] 도 8에 도시된 바와 같이, 인가 서브루틴(702)은 클라이언트 컴퓨팅 디바이스(110)가 인가 서버(122)에 구성 요구(803)를 통신할 때 시작된다. 본 발명의 예시적인 실시예에서, 구성 요구(803)는 인가 서버(122) 상에 저장된 인가 플러그인들을 설명하는 정보를 획득하도록 구성된 임의의 적당한 소프트웨어 컴포넌트로 형성된다. 숙련된 당업자라면 인식하겠지만, 구성 요구(803)는 "GetConfig"로 불리는 공지의 방법을 이용할 수 있다. 구성 요구(803)를 수신하는 것에 응답하여, 인가 서버(122)는 구성 응답(804)을 통신하고, 이 구성 응답은 인가 서버(122) 상에 저장된 모든 인가 플러그인들을 식별하는 정보를 포함한다. 일 실시예에서, 구성 요구(804)는 인가 서버(122) 상에 저장된 모든 인가 플러그인들을 식별하고 설명하는 문자열들(strings)의 배열을 포함한다. 본 예에서는, 구성 요구(804)는 제1 서버 인가 플러그인(828), 제2 서버 인가 플러그인(829), 및 제3 서버 인가 플러그인(830)을 식별하는 정보를 포함한다.

[0073] 블록(805)에서, 클라이언트 컴퓨팅 디바이스(110)는 구성 응답(804)을 수신하는 것에 응답하여 하나 이상의 인가 쿠키들을 생성한다. 블록(805)의 처리 시에, 클라이언트 컴퓨팅 디바이스(110)는 클라이언트 및 서버 인가 플러그인들을 매칭시키는 각각의 쌍에 대한 인가 쿠키를 생성한다. 따라서, 본 예에서는, 제1 클라이언트 인가 플러그인(825)은 모든 컴퓨터 타깃 그룹과 관련된 제1 인가 쿠키를 생성한다. 왜냐하면 제1 클라이언트 인가 플러그인(825) 및 제1 서버 인가 플러그인(828)은 둘 다 모든 컴퓨터 타깃 그룹과 관련되어 있기 때문이다. 또한, 제2 클라이언트 인가 플러그인(826)은 PID 타깃 그룹과 관련된 제2 인가 플러그인을 생성한다. 왜냐하면 제2 클라이언트 인가 플러그인(826) 및 제2 서버 인가 플러그인(829)은 둘 다 PID 타깃 그룹과 관련되어 있기 때문이다. 제3 인가 쿠키는 생성되지 않는다. 왜냐하면 클라이언트 컴퓨팅 디바이스(110)는 그것이 베타 타깃 그룹의 멤버임을 나타내는 인가 플러그인을 갖고 있지 않기 때문이다.

[0074] 숙련된 당업자라면 인식하겠지만, 블록(805)의 처리의 일 구현예는 이 기술 분야에서 "GetAuthCookie"로 불리는 일반적으로 공지된 소프트웨어 방법의 이용을 포함할 수 있다. 또한 각각의 인가 쿠키의 생성은 부가적인 처리를 포함할 수 있다는 것을 인식할 것이다. 이를테면, 제2 클라이언트 인가 플러그인(826)은 PID를 검색하고 그 PID를 인가 쿠키에 포함시키기 위해 클라이언트의 시스템 레지스트리에 저장된 정보를 검사하도록 구성될 수 있다. 다른 예들에서, 블록(805)의 처리는 다른 컴퓨터 또는 디바이스와 통신하기 위한 처리들을 포함할 수 있다. 이를테면, 클라이언트 인가 플러그인은 사운드 카드, 스캐너, 비디오 카드 등과 같은 디바이스와 통신하여 그 디바이스의 메이크(make) 및 모델을 획득할 수 있다. 다른 비한정적인 예들에서, 클라이언트 인가 플러그인은 지문 판독기와 같은 보안 디바이스와 통신하여 사용자를 설명하는 정보를 획득할 수 있다.

[0075] 일반적으로, 클라이언트 인가 플러그인은 클라이언트 컴퓨팅 디바이스(110)의 임의의 컴포넌트 또는 클라이언트 컴퓨팅 디바이스(110)에 통신적으로 접속된 임의의 다른 컴퓨팅 디바이스로부터 구성 정보를 판독할 수 있다. 다른 예들에서, 클라이언트 인가 플러그인은 하나 이상의 공용 또는 전용(public or private) 애플리케이션 프

로그래밍 인터페이스들(API들)을 이용하여 대응하는 서버 플러그인에 의해 유효성 확인될 클라이언트로부터 정보를 수집 및 암호화하도록 구성될 수 있다. 그러한 예들에서, PID 밸리데이터 플러그인(826)은 전용 API를 이용하여 클라이언트의 PID를 암호화하여 이 암호화된 PID를 암호 해독 및 유효성 확인을 위해 서버에 전달한다. 다른 실시예들에서는, 다른 클라이언트 인가 플러그인들이 지문 판독기 또는 성문(voice prints)과 같은 생체 측정을 이용하여 유효성 확인을 위해 서버에 전달될 인가 쿠키를 구성할 수 있다. 또 다른 예에서는, 클라이언트 인가 플러그인이 웹 서버 또는 임의의 다른 서비스를 호출하여 인가 자격 증명(authorization credentials) 또는 임의의 다른 타입의 데이터를 인가 서버(122)에 통신할 수 있다.

[0076] 본 발명이 예시적인 실시예에서, 각각의 인가 쿠키는 관련 타깃 그룹을 식별하는 문자열을 포함한다. 예를 들면, 문자열은 특정 인가 쿠키가 PID 타깃 그룹과 관련되어 있음을 나타낼 수 있다. 각각의 인가 쿠키는 또한 클라이언트와 서버 사이의 데이터 통신을 위한 데이터 부분을 포함한다. 예를 들면, PID 타깃 그룹과 관련된 인가 쿠키는 실제 PID를 포함하는 데이터 부분을 가질 수 있다. 숙련된 당업자라면 인식하겠지만, 데이터 부분은 바이트 배열과 같은 임의의 포맷으로 저장되어 있는 임의의 타입의 데이터를 포함할 수 있다. 예를 들면, 클라이언트 및 서버 상의 플러그인들이 공개 및 개인 키들의 통신을 요구할 경우에는, 그러한 데이터는 하나 이상의 인가 쿠키들의 데이터 부분에 암호화될 수 있다.

[0077] 일단 클라이언트 컴퓨팅 디바이스(110)가 대응하는 클라이언트 및 서버 인가 플러그인들의 각 쌍에 대한 인가 쿠키를 생성하면, 클라이언트 컴퓨팅 디바이스는 생성된 인가 쿠키들을 인가 서버(122)에 통신한다. 도 8에 도시된 바와 같이, 클라이언트 컴퓨팅 디바이스(110)는 쿠키 요구(806)에서 인가 쿠키들을 통신한다. 쿠키 요구(806)는 블록(805)의 처리 중에 생성된 인가 쿠키들의 배열을 통신하기 위한 임의의 적당한 포맷을 포함한다. 인가 방법(702)의 이 부분의 일 구현예는 이 기술 분야에서 "GetCookie"로 불리는 일반적으로 공지된 소프트웨어 방법의 이용을 포함할 수 있다.

[0078] 일 실시예에서, 쿠키 요구(806)는 또한 클라이언트 컴퓨팅 디바이스(110)의 메모리에 저장된 다른 인가 서버 쿠키들을 포함한다. 이하의 설명으로부터 더 쉽게 이해되겠지만, 클라이언트 컴퓨팅 디바이스(110)의 메모리는 인가 루틴(700)의 이전 실행들에서 생성된 오래된 인가 서버 쿠키들을 저장할 수 있다. 저장된 인가 서버 쿠키들을 쿠키 요구(806) 내에 제공함으로써, 클라이언트 컴퓨팅 디바이스(110)는 인가 서브루틴(702)의 이전 실행들에서 승인된 그것의 액세스 특권들을 유지할 수 있을 것이다. 본 예에서는, 클라이언트에 어떠한 인가 서버 쿠키도 저장되어 있지 않기 때문에, 쿠키 요구(806)는 모든 컴퓨터 타깃 그룹과 관련된 제1 인가 쿠키 및 PID 타깃 그룹과 관련된 제2 인가 쿠키를 포함한다.

[0079] 다음으로, 블록(807)에 도시된 바와 같이, 쿠키 요구(806)를 수신하는 것에 응답하여, 인가 서버(122)는 서버 쿠키를 생성한다. 일 실시예에서, 수신된 인가 쿠키들 각각에 대해, 서버 쿠키 데이터를 생성하는 적당한 서버 인가 플러그인에 대해 호출이 행해진다. 각각의 서버 인가 플러그인에 의해 생성된 서버 쿠키 데이터는 수신된 인가 쿠키들에서 식별된 각각의 타깃 그룹에 대한 식별자를 포함한다. 본 예에서는, 쿠키 요구(806)가 모든 컴퓨터 타깃 그룹과 관련된 제1 인가 쿠키 및 PID 타깃 그룹과 관련된 제2 인가 쿠키를 포함하기 때문에, 인가 서버(122)는 이들 각각의 타깃 그룹들에 대한 식별자를 포함하는 서버 쿠키 데이터를 생성한다. 인가 서버(122) 상에서, 서버 쿠키 데이터는, 쿠키 요구(806)에서 오래된 서버 쿠키가 수신되면, 오래된 서버 쿠키들의 데이터와 결합되어, 새로운 서버 쿠키를 생성한다. 일 실시예에서, 새로운 서버 쿠키는 Triple DES와 같은 공개적으로 이용 가능한 암호화 방법을 이용하여 암호화된다.

[0080] 본 발명의 예시적인 실시예에서, 서버 쿠키는 하나 이상의 관련 타깃 그룹을 식별하는 암호화된 정보를 포함할 수 있다. 또한, 서버 쿠키는 명백한 텍스트 포맷 및 암호화된 포맷 양쪽 모두로 저장되는 만료 데이터(expiration data)를 포함할 수 있다. 명백한 텍스트 포맷으로 저장된 만료 데이터는 클라이언트 컴퓨팅 디바이스(110)에 의해 서버 쿠키의 만료를 모니터하는 데 이용된다. 암호화된 포맷으로 저장된 만료 데이터는 소프트웨어 업데이트 서비스(120)에 의해 클라이언트 컴퓨팅 디바이스(110)가 특정 타깃 그룹과 관련된 업데이트들을 수신하도록 인가되어 있는지를 판정하는 데 이용된다. 일 실시예에서, 서버 쿠키의 만료 데이터는 서버 쿠키에서 식별된 모든 타깃 그룹들에 적용된다. 대안적으로, 또는 전체 서버 쿠키에 적용되는 만료 시간에 더하여, 서버 쿠키는 개개의 타깃 그룹들에 각각 적용될 수 있는 복수의 만료 데이터를 포함할 수 있다. 숙련된 당업자라면 인식하겠지만, 각각의 서버 쿠키는 부가적인 데이터를 포함할 수 있다. 이를테면, 서버 쿠키는 인가 서브루틴(702)의 최종 실행의 타임 스탬프와 같은 클라이언트 상태 정보를 저장하도록 구성될 수 있다.

[0081] 인가 서버 쿠키(809)는, 일단 생성되면, 인가 서버(122)로부터 클라이언트 컴퓨팅 디바이스(110)로 통신된다. 다음으로, 블록(811)에 도시된 바와 같이, 서버 쿠키는 그 후에 클라이언트 컴퓨팅 디바이스(110)의 메모리에

저장된다. 클라이언트 컴퓨팅 디바이스(110)가 서버 쿠키의 적어도 하나의 컴포넌트가 만료된 것으로 판정할 경우, 클라이언트 컴퓨팅 디바이스는 새로운 서버 쿠키를 획득하기 위해 인가 방법(702)을 다시 시작한다. 상술한 바와 같이, 인가 방법(702)의 각각의 후속 실행 중에, 클라이언트 컴퓨팅 디바이스(110)는 그것의 저장된 서버 쿠키들을 쿠키 요구(806)에서 인가 서버(122)에 통신할 수 있다. 일 실시예에서, 클라이언트는, 서버가 클라이언트에게 서버 구성이 변경되었음을, 즉 새로운 인가 플러그인이 추가되었음을 통지하지 않는 한 요구(803)를 송신할 필요가 없다.

[0082] 본 발명의 다른 특징에 따르면, 소프트웨어 업데이트 서비스(120)는 메타데이터 서버(123)와 클라이언트 컴퓨팅 디바이스(110) 사이에 업데이트 정보를 동기화하기 위한 동기화 서브루틴을 제공할 수 있다. 고유의 소프트웨어 업데이트 계층 구조를 이용함으로써, 동기화 서브루틴은 특정 클라이언트 컴퓨팅 디바이스에 적용되는 특정 업데이트들을 효율적으로 식별할 수 있다. 게다가, 인증 서브루틴(702)에서 생성된 서버 쿠키를 이용함으로써, 동기화 서브루틴은 특정 타깃 그룹들과 관련된 업데이트들에의 액세스를 선택적으로 승인할 수 있다.

[0083] 본 발명의 예시적인 실시예에 따르면, 각각의 소프트웨어 업데이트는 3개의 컴포넌트: (1) 명령 컴포넌트와; (2) 로컬화된(localized) 데이터 컴포넌트와; (3) 데이터 컴포넌트를 포함한다. 숙련된 당업자라면 인식하겠지만, 각각의 업데이트는 상술한 컴포넌트들 중 하나 이상을 가질 수 있다. 예를 들면, 업데이트는 명령 컴포넌트와, 로컬화된 데이터 컴포넌트와, 데이트 스트림 컴포넌트를 포함할 수 있다. 다른 예에서는, 업데이트는 클라이언트 컴퓨팅 디바이스의 하나 이상의 조건들을 테스트하기 위한 명령 컴포넌트만을 포함할 수 있다. 소프트웨어 업데이트들의 갖가지 컴포넌트들에 대해서는 아래에서 더 상세히 설명한다.

[0084] 일반적으로, 명령 컴포넌트는 2개의 서브컴포넌트: (1) 클라이언트 컴퓨팅 디바이스(110)에 의해 테스트될 하나 이상의 조건들을 정의하는 적용 가능성 규칙(applicability rule)과; (2) 개개의 업데이트의 적절한 설치를 위해 요구되는 하나 이상의 업데이트들을 식별하는 필요 조건들의 세트를 포함한다. 후술하는 바와 같이, 적용 가능성 규칙은 컴퓨터와 관련된 다수의 조건들을 정의할 수 있고, 이 조건들 각각은 임의의 논리 연산자들을 이용하여 다른 조건들과 관련될 수 있다. 예를 들면, 명령 컴포넌트는 컴퓨터에 특정 버전의 Windows®가 설치되어 있는지를 판정하는 적용 가능성 규칙을 포함할 수 있다. 또한 후술하는 바와 같이, 필요 조건의 세트는 이전에 설치되어 있어야 할 하나 이상의 업데이트들을 식별할 수 있다. 예를 들면, 도 9와 관련하여 아래에서 더 상세히 설명하는 바와 같이, 개개의 업데이트는 그 개개의 업데이트의 적절한 설치를 위해 요구되는 다른 업데이트들을 목록화하는 필요 조건을 포함할 수 있다. 다른 예들에서는, 도 9에도 도시된 바와 같이, 필요 조건들의 세트는 논리 연산자들을 이용하여 더 복잡한 필요 조건 규칙들을 정의하는 것을 포함할 수 있다.

[0085] 명령 컴포넌트는 또한 특정 업데이트에 종속하는 다른 업데이트들이 있는지를 나타내는 부울 플래그(Boolean flag)와 같은 코드를 포함한다. 예시의 목적으로, 특정 업데이트에 종속하는 다른 업데이트들이 없을 경우 업데이트는 LEAF 업데이트로 간주된다. 업데이트가 LEAF인지를 나타내기 위해 이용되는 부울 플래그는 관련 업데이트들이 추가되거나 제거될 때 메타데이터 서버(123)에 의해 동적으로 업데이트된다.

[0086] 각각의 업데이트의 로컬화된 데이터 컴포넌트는 업데이트를 설명하는 일반 정보를 포함한다. 이를테면, 로컬화된 데이터는 업데이트의 특징들 및 이점들을 설명하는 정보를 포함할 수 있다. 로컬화된 데이터 컴포넌트는 또한 업데이트의 설치 절차에 대한 텍스트 설명을 포함할 수 있다. 게다가, 로컬화된 데이터는 업데이트와 관련이 있는 임의의 다른 데이터 또는 정보를 포함할 수 있다. 이를테면, 로컬화된 데이터는 업데이트가 높은 우선순위 업데이트임을 나타낼 수 있다. 다른 예에서, 로컬화된 데이터는 업데이트가 다른 소프트웨어 업데이트들과 함께 설치될 수 없음을 나타내는 메시지와 같은 특별한 설치 메시지들을 제공할 수 있다. 로컬화된 정보는 그것에 포함된 정보를 사용자에게 디스플레이할 수 있게 하는 포맷으로 되어 있을 수 있다.

[0087] 각각의 업데이트의 데이터 컴포넌트는 그 업데이트의 하나 이상의 바이너리 데이터 스트림을 포함한다. 일 실시예에서, 각각의 업데이트의 데이터 컴포넌트는 실행 가능 파일, 문서, 링크된 라이브러리 등과 같은 하나 이상의 데이터 파일들과 관련될 수 있다. 아래에서 더 상세히 설명하는 바와 같이, 각각의 업데이트는 데이터 파일들의 조합과 관련될 수 있고, 각각의 데이터 파일은 클라이언트에 의해 사용되는 소프트웨어의 실제 업그레이드, 설치, 또는 수정을 용이하게 한다. 이를테면, 업데이트의 설치는 선택된 업데이트를 완료하기 위해 요구되는 모든 정보를 포함하는 단일 CAB 파일을 이용함으로써 용이하게 될 수 있다. 대안적으로, 업데이트의 설치는 클라이언트 컴퓨팅 디바이스 상에 저장된 하나 이상의 파일들을 업데이트하기 위해 이용되는 다수의 개개의 업데이트들을 이용함으로써 용이하게 될 수 있다.

[0088] 본 발명의 예시적인 실시예에서, 소프트웨어 업데이트들은 소프트웨어 업데이트들의 배포 제어를 가능케 하는 계층 구조로 배열될 수 있다. 일반적으로, 업데이트들의 계층 구조는 업데이트들 간의 관계를 정의하고 특히

어느 업데이트들이 다른 업데이트들에 종속하는지를 나타낸다. 예시의 목적으로, 업데이트들의 세트 예가 도 9에 제공되고 도시되어 있다. 도시된 바와 같이, 샘플 업데이트들(900)의 계층 구조는 기본 세트의 업데이트들(901)과, 제2 세트의 업데이트들(902)과, 제3 세트의 업데이트들(903)을 포함한다. 일반적으로 기본 세트의 업데이트들(901) 내의 각각의 업데이트는 다른 업데이트들의 설치를 필요로 하는 필요 조건을 갖지 않는다. 그러나, 제6 업데이트(921)는 제1 업데이트(911)와, 제2 업데이트(912)와, 제3 업데이트(913)의 설치를 필요로 하는 필요 조건을 포함한다. 제7 업데이트(922)는 제4 업데이트(914)의 설치를 필요로 하는 필요 조건을 포함한다. 제8 업데이트(931)는 제6 업데이트(921)와 제5 업데이트(915)의 설치를 필요로 하는 필요 조건을 포함한다. 그러므로, 제8 업데이트(931)는 또한 제1 업데이트(911)와, 제2 업데이트(912)와, 제3 업데이트(913)의 설치를 필요로 한다. 본 발명을 예시할 목적으로, 샘플 세트의 업데이트들(900)의 모든 업데이트들이 모두 모든 컴퓨터 타깃 그룹 및 PID 타깃 그룹과 관련되어 있는 것으로 가정한다.

[0089] 또한 도 9에 도시되어 있고, 아래에서 더 상세히 설명하는 바와 같이, 각각의 업데이트는 그 업데이트의 설치를 위한 조건들을 특정하는 적용 가능성 규칙을 포함한다. 예를 들면, 제1 업데이트(911)는 영어 버전의 운영 시스템의 설치를 필요로 한다. 제2 업데이트(912)는 Windows® XP 버전 SP1의 설치를 필요로 한다. 다른 예에서, 제6 업데이트(921)는 XP PATCH1로 불리는 소프트웨어 패치의 설치를 필요로 한다. 따라서, 클라이언트 컴퓨팅 디바이스(110)는 적용 가능성 규칙들이 만족되어 있지 않으면 업데이트를 설치하지 않을 것이다.

[0090] 도 9는 또한 특정 업데이트에 종속하는 다른 업데이트들이 있는지를 나타내는 명령 컴포넌트의 부분을 도시한다. 이 부분은 LEAF로 불리고, 이것은 소프트웨어 업데이트가 일련의 관련 업데이트들 중 최종 업데이트임을 나타내는 부울 값의 형태로 되어 있을 수 있다. 본 발명을 예시할 목적으로, 특정 업데이트를 필요 조건으로 목록화하는 다른 업데이트들이 없다면 그 특정 업데이트는 LEAF 업데이트이다. 도 9에 도시된 바와 같이, 제7 업데이트(922) 및 제8 업데이트(931)는 유일한 2개의 LEAF 업데이트들이다.

[0091] 도 10은 본 발명에 따라서 형성된 동기화 서브루틴(704)(도 7)의 프로토콜 도면이다. 일반적으로, 동기화 서브루틴(704)은 클라이언트 컴퓨팅 디바이스(110)와 메타데이터 서버(123)와 같은 서버 사이에 특정 업데이트들의 명령 컴포넌트들을 선택적으로 통신하여, 클라이언트 컴퓨팅 디바이스(110)에 적용될 수 있는 업데이트들을 식별한다. 도 10에 도시된 바와 같이, 업데이트를 개시하기 위하여, 클라이언트 컴퓨팅 디바이스(110)는 우선 설치된 업데이트들을 처리하고 클라이언트에게 이용될 수 있는 하나 이상의 업데이트들을 요구하는 동기화 요구(1051)를 메타데이터 서버(123)에 통신한다. 동기화 요구(1051)를 수신하는 것에 응답하여, 메타데이터 서버(123)는 다수의 업데이트들을 클라이언트 컴퓨팅 디바이스(110)에 반환(return)한다. 이하의 설명으로부터 더 쉽게 이해되겠지만, 클라이언트 컴퓨팅 디바이스(110)는 동기화 요구(1051)의 통신에 앞서서 로컬로 저장된 데이터를 처리한다.

[0092] 클라이언트 컴퓨팅 디바이스(110)는 각각의 수신된 업데이트의 명령 컴포넌트들을 처리하여 적용 가능성 규칙들에서 정의된 조건이 만족될 수 있는지를 판정한다. 만일 개개의 업데이트에서 정의된 조건이 만족되면, 동기화 서브루틴(1050)을 예시할 목적으로, 개개의 업데이트는 "설치되고", 설치된 업데이트는 클라이언트의 업데이트 캐시의 제1 컴포넌트에 저장된다. 다른 한편으로, 만일 개개의 업데이트에서 정의된 조건이 만족되지 않으면, 개개의 업데이트는 "실패"(failed)한 것으로 간주되고, 실패한 업데이트는 클라이언트의 업데이트 캐시의 제2 컴포넌트에 저장된다. 동기화 서브루틴(1050)에 대한 이 설명에서, 업데이트가 설치되면 필요 조건들 및 적용 가능성 규칙들의 조건들이 만족된 것으로 추정될 수 있다. 이 서브루틴을 설명할 목적으로 업데이트의 설치는 업데이트와 관련된 데이터 파일들이 클라이언트 컴퓨팅 디바이스(110)에 실제로 설치되는 것을 반드시 의미하지는 않는다.

[0093] 일 실시예에서, 클라이언트의 업데이트 캐시의 2개의 컴포넌트들은 수신된 업데이트들을 분류하기 위해 사용될 수 있다. 제1 컴포넌트는 설치된 비-LEAF(non-LEAF) 업데이트들을 저장하기 위해 사용되고, 제2 컴포넌트는 클라이언트에 의해 수신된 모든 다른 업데이트들, 즉 설치되지 않은 업데이트들을 저장하기 위해 사용된다. 업데이트 캐시의 제2 컴포넌트는 또한 모든 LEAF 업데이트들의 저장을 포함한다. 아래에서 더 상세히 설명하는 바와 같이, 업데이트 캐시에 저장된 업데이트들은 메타데이터 서버(123)에 통신되어 메타데이터 서버(123)에 의해 클라이언트 컴퓨팅 디바이스(110) 상에 설치하기 위해 이용 가능한 다른 관련 업데이트들을 식별하도록 처리될 수 있다.

[0094] 이제 도 10으로 돌아가서, 항목들(1051, 1055, 및 1060)로 예시되어 있는 동기화 요구의 상세에 대하여 설명하겠다. 숙련된 당업자라면 인식하겠지만, 동기화 요구는 다수의 서로 다른 디바이스들, 처리들, 애플리케이션들, 사용자 개시 커맨드들 중 하나에 의해 개시되어, 업데이트를 요구할 수 있다. 동기화 요구는

업데이트들의 목록을 요구하는 사용자, 클라이언트에 의해 개시된 자동 업데이트, 또는 메타데이터 서버(123) 또는 업데이트 서버(120)로부터 정보를 요구하는 임의의 다른 소프트웨어 컴포넌트에 의해 개시될 수 있다. 일 실시예에서, 동기화 요구는 인가 루틴(702)으로부터 생성된 인가 서버 쿠키와 같은 인가 서버 쿠키를 포함한다. 서버 쿠키의 사용으로 서버는 클라이언트가 하나 이상의 타깃 그룹들의 멤버인지를 판정할 수 있게 된다.

[0095] 각각의 동기화 요구는 또한 클라이언트의 업데이트 캐시에 저장된 각각의 업데이트에 대한 식별자들을 포함할 수 있다. 더 구체적으로, 만일 하나 이상의 업데이트들이 업데이트 캐시에 저장되어 있다면, 동기화 요구는 설치된 비-LEAF 업데이트들에 대한 식별자들을 갖는 제1 컴포넌트와; LEAF 업데이트, 실패한 업데이트, 및 설치되지 않은 다른 업데이트들에 대한 식별자들을 갖는 제2 컴포넌트를 포함한다. 업데이트 식별자들은 정수의 배열을 포함하는(그러나 이것에 한정되지 않는) 임의의 포맷으로 되어 있을 수 있다. 대안적으로, 만일 클라이언트의 업데이트 캐시에 어떠한 업데이트도 저장되어 있지 않다면, 동기화 요구는 업데이트 식별자와 함께 구성되지 않는다. 동기화 요구가 업데이트 식별자와 함께 구성되지 않을 경우, 동기화 요구는 클라이언트 컴퓨팅 디바이스(110)가 어떠한 캐싱된 업데이트도 갖고 있지 않다고 하는 지시(indication)를 제공한다.

[0096] 도 10에 도시된 바와 같이, 제1 동기화 요구(1051)는 클라이언트 컴퓨팅 디바이스(110)로부터 메타데이터 서버(123)로 통신된다. 본 예에서는, 클라이언트의 업데이트 캐시는, 이것이 이 방법의 첫 번째 실행이라고 가정할 때, 어떠한 업데이트도 포함하지 않을 것이다. 따라서, 제1 동기화 요구(1051)는 캐싱된 업데이트에 대한 식별자를 포함하지 않는다. 동기화 요구를 수신하는 것에 응답하여, 블록(1052)에 도시된 바와 같이, 메타데이터 서버(123)는 동기화 요구가 적어도 하나의 업데이트 식별자를 포함하는지를 판정한다. 만일 동기화 요구가 업데이트 식별자를 포함하지 않는 것으로 판정되면, 메타데이터 서버(123)는 클라이언트 컴퓨팅 디바이스(110)에 통신하기 위한 제1 레벨 업데이트들을 선택하는 것으로 응답한다. 상술한 바와 같이, 제1 레벨 업데이트들은 다른 업데이트들을 식별하는 필요 조건을 갖지 않는 임의의 업데이트들을 포함할 수 있다.

[0097] 대안적으로, 만일 동기화 요구가 적어도 하나의 업데이트 식별자를 포함하는 것으로 판정되면, 메타데이터 서버(123)는 서버의 저장된 업데이트들의 필요 조건들을 검사하여 클라이언트에 전달하기 위한 부가적인 업데이트들을 선택한다. 일 실시예에서, 메타데이터 서버(123)는 충족된 필요 조건들을 갖는 업데이트들을 선택한다. 필요 조건들의 검사 시에, 서버는 동기화 요구의 제1 컴포넌트의 업데이트들을 이용하고, 동기화 요구는 클라이언트 상에 설치되어 있는 비-LEAF 업데이트들의 식별자들을 포함한다.

[0098] 충족된 필요 조건들을 갖는 업데이트들을 선택하는 것 외에, 서버는 또한 동기화 요구의 제2 컴포넌트에서 식별된 업데이트들을 이용하여 선택된 업데이트들을 필터링한다. 더 구체적으로, 동기화 요구에서 식별된 비설치된 업데이트들, LEAF 업데이트들 및 실패한 업데이트들이 하나 이상의 선택된 업데이트들을 필터링하는 데 이용된다. 본 발명의 이러한 특징으로 본 발명의 시스템 및 방법이 메타데이터 서버(123) 상에 저장된 업데이트들의 다중 송신을 피할 수 있게 된다.

[0099] 본 예로 돌아가서, 제1 동기화 요구(1051)는 업데이트 식별자를 포함하지 않기 때문에, 메타데이터 서버(123)는 클라이언트 컴퓨팅 디바이스(110)에 통신하기 위한 기본 레벨의 업데이트들(901)을 선택한다. 도 9에 도시된 샘플 세트의 업데이트들과 관련하여, 기본 레벨의 업데이트들(901)은 (911, 912, 913, 914, 및 915)로 참조되는 업데이트들을 포함한다.

[0100] 블록(1052)의 처리 중에, 메타데이터 서버(123)는 또한 동기화 요구(1051)에 포함된 동기화 서버 쿠키를 검사하여 클라이언트 컴퓨팅 시스템(110)과 관련되어 있는 타깃 그룹들을 식별한다. 메타데이터 서버(123)는 또한 블록(1052)의 처리 중에 선택된 업데이트들의 타깃 그룹들을 검사한다. 그 후에 블록(1052)의 처리는 수신된 동기화 서버 쿠키에서 식별된 타깃 그룹과 관련되어 있지 않은 모든 선택된 업데이트들을 필터링 제거한다. 본 예에서는, 선택된 업데이트들(911, 912, 913, 914, 및 915) 전부가 PID 및 모든 컴퓨터 타깃 그룹들과 관련되어 있기 때문에, 선택된 업데이트들 전부가 클라이언트 컴퓨팅 디바이스(110)에 송신된다.

[0101] 그 후에 메타데이터 서버(123)는 선택된 업데이트들을 동기화 응답(1053)에서 클라이언트 컴퓨팅 디바이스(110)에 통신한다. 일반적으로, 각각의 동기화 응답은 서버(120)에 의해 송신된 각각의 업데이트의 명령 컴포넌트를 포함한다. 따라서, 본 예에서는, 제1 동기화 응답(1053)은 (911, 912, 913, 914, 및 915)로 참조되는 업데이트들에 대한 명령 컴포넌트들을 포함한다. 일 실시예에서, 각각의 동기화 응답은 각각의 업데이트의 로컬화된 데이터 컴포넌트 또는 데이터 컴포넌트를 포함하지 않는다.

[0102] 다음으로, 블록(1054)에 도시된 바와 같이, 클라이언트 컴퓨팅 디바이스(110)는 각각의 수신된 업데이트의 명령 컴포넌트들을 처리하여 적용 가능성 규칙들에서 정의된 조건이 충족될 수 있는지를 판정한다. 다시 도 9와 관

련하여, 클라이언트 컴퓨팅 디바이스(110)는 수신된 업데이트들(911-915)의 명령 컴포넌트들을 처리한다. 본 발명을 예시할 목적으로, 이 예에서는 클라이언트 컴퓨팅 디바이스(110)의 운영 시스템은 Windows® 버전 XP SP1의 영어 버전 설치인 것으로 가정한다. 또한 클라이언트 컴퓨팅 디바이스는 델 PC이고 32비트, X86 프로세서를 실행하고 있다고 가정한다. 따라서, 샘플 세트의 업데이트들의 명령 컴포넌트들의 처리 중에, 클라이언트 컴퓨팅 디바이스(110)는 컴퓨터가 영어 버전 OS를 포함하고 있기 때문에 제1 업데이트(911)에서 정의된 조건이 충족될 것으로 판정할 것이다. 운영 시스템은 Windows® 버전 XP SP1이기 때문에 제2 업데이트(912)에서 정의된 조건이 충족될 것이다. 클라이언트 컴퓨팅 디바이스(110)가 X86 프로세서를 실행하고 있기 때문에 제3 업데이트(913)에서 정의된 조건이 충족될 것이다. 클라이언트 컴퓨팅 디바이스(110)가 델 PC이기 때문에 제5 업데이트(915)에서 정의된 조건이 충족될 것이다. 그 결과, 제1 업데이트(911), 제2 업데이트(912), 제3 업데이트(913), 및 제5 업데이트(915)는 모두 클라이언트의 업데이트 캐시의 제1 컴포넌트에 저장된다. 클라이언트 컴퓨팅 디바이스(110)가 64비트 X86 프로세서를 실행하고 있지 않기 때문에 제4 업데이트(914)에서 정의된 조건은 충족되지 않을 것이다. 따라서, 제4 업데이트(914)는 실패한 업데이트로 간주되어 클라이언트의 업데이트 캐시의 제2 컴포넌트에 저장된다.

[0103] 도 10으로 돌아가서, 블록(1054)의 처리 중에, 클라이언트 컴퓨팅 디바이스(110)는 또한 후속 동기화 요구가 필요한지를 판정한다. 일 실시예에서, 만일 수신된 업데이트들 중 적어도 하나가 그것이 LEAF 업데이트가 아님을 나타내면 후속 동기화 요구가 필요한 것으로 판정된다. 본 예에서는, 수신된 업데이트들 전부가 LEAF 업데이트가 아니기 때문에 후속 동기화 요구가 필요한 것으로 판정된다. 따라서, 클라이언트 컴퓨팅 디바이스(110)는 후속 동기화 요구(1055)를 메타데이터 서버(123)에 통신한다.

[0104] 위에서 요약 설명한 바와 같이, 동기화 요구는 클라이언트의 업데이트 캐시에 저장된 각각의 업데이트에 대한 식별자들을 포함한다. 따라서, 본 예에서는, 후속 동기화 요구(855)는 제1 업데이트(911), 제2 업데이트(912), 제3 업데이트(913), 및 제5 업데이트(915)가 클라이언트 상에 설치되어 있음을 나타내는 제1 데이터 컴포넌트를 포함한다. 또한, 후속 동기화 요구(855)는 제4 업데이트(914)가 클라이언트 상에 성공적으로 설치되어 있지 않음을 나타내는 제2 데이터 컴포넌트를 포함한다.

[0105] 후속 동기화 요구(1055)를 수신하는 것에 응답하여, 위에서 요약 설명한 바와 같이, 메타데이터 서버(123)는 후속 동기화 요구가 적어도 하나의 업데이트 식별자를 포함하는지를 판정한다. 만일 후속 동기화 요구가 적어도 하나의 업데이트 식별자를 포함하는 것으로 판정되면, 메타데이터 서버(123)는 모든 저장된 업데이트들의 필요 조건들을 검사하여 클라이언트에 전달하기 위한 부가적인 업데이트들을 선택한다.

[0106] 다시 본 예와 관련하여, 블록(1056)의 처리 중에, 메타데이터 서버(123)는 제6 업데이트(921)의 필요 조건이 충족되기 때문에 제6 업데이트(921)를 선택할 것이다. 더 구체적으로, 도 9에 도시된 바와 같이, 제6 업데이트(921)가 클라이언트 컴퓨팅 디바이스(110)에의 통신을 위해 선택된다. 왜냐하면 제1 업데이트(911), 제2 업데이트(912), 및 제3 업데이트(913)의 설치를 필요로 하는 제6 업데이트(921)의 필요 조건이 충족되기 때문이다. 제7 업데이트(922) 및 제8 업데이트(931)는 그것들의 필요 조건이 충족되지 않기 때문에 클라이언트에의 통신을 위해 선택되지 않을 것이다. 더 구체적으로, 동기화 요구(1055)는 제7 업데이트(922)에 대한 필요 조건인, 제4 업데이트(914)에 대한 식별자를 포함하지 않았다. 게다가, 동기화 요구(1055)는 제8 업데이트(931)에 대한 필요 조건인, 제6 업데이트(921)에 대한 식별자를 포함하지 않았다.

[0107] 도 10으로 돌아가서, 동기화 서브루틴(1050)은 후속 응답(1057)에서 메타데이터 서버(123)로부터 클라이언트 컴퓨팅 디바이스(110)로 선택된 업데이트들을 통신함으로써 계속된다. 다시 본 예와 관련하여, 후속 응답(1057)은 제6 업데이트(721)와 관련된 정보를 포함할 것이고, 후속 응답(1057)에서 클라이언트(110)로 통신된다.

[0108] 후속 응답(1057)을 수신하면, 클라이언트 컴퓨팅 디바이스(110)는 후속 응답(1057)의 명령 컴포넌트들을 처리한다. 블록(854)의 처리와 유사하게, 클라이언트 컴퓨팅 디바이스(110)는 각각의 수신된 업데이트의 명령 컴포넌트들을 처리하여 적용 가능성 규칙들에서 정의된 조건이 충족되는지를 판정한다. 본 예에서, 클라이언트 컴퓨팅 디바이스에 XP PATCH1이 설치되어 있다고 가정하면, 제6 업데이트(921)는 설치되는 것으로 간주되고 업데이트는 클라이언트 컴퓨팅 디바이스(110)의 업데이트 캐시에 기입된다. 제6 업데이트(921)는 LEAF 업데이트가 아니므로, 클라이언트 컴퓨팅 디바이스(110)는 클라이언트의 업데이트 캐시의 제1 및 제2 컴포넌트에 저장된 모든 업데이트들을 포함하는 또 다른 동기화 요구(1060)를 송신한다. 동기화 요구(1060)는 또한 인가 서버 쿠키를 포함한다.

[0109] 본 예에서는, 메타데이터 서버(123)의 상술한 처리를 이용하여, 동기화 요구(1060)는 블록(1061)에서 처리되고,

거기서 메타데이터 서버는 제8 업데이트(931)를 선택한다. 제8 업데이트(931)가 선택되는 이유는, 동기화 요구(1060)가, 클라이언트 컴퓨팅 디바이스(110)에 제5 및 제6 업데이트들(915 및 921)이 설치되어 있음을 나타내기 때문이다. 제8 업데이트(931)가 인가 서버 쿠키에서 식별된 동일 타깃 그룹들과 관련되어 있다고 가정하면, 제8 업데이트(931)의 명령 컴포넌트는 또 다른 응답(1062)에서 클라이언트 컴퓨팅 디바이스(110)에 통신된다. 제8 업데이트(931)는 그 후에 블록들(1054 및 1059)의 처리와 유사한 방법으로 블록(1063)에서 처리된다. 응답(1062)의 수신된 업데이트들 모두가 LEAP 업데이트들이므로, 후속 동기화 요구가 메타데이터 서버(123)에 다시 송신되지 않는다.

[0110] 클라이언트 컴퓨팅 디바이스(110)에서, 수신된 업데이트들 모두가 LEAF 업데이트들인 것으로 판정된 후이거나 또는 응답(1062)에서 어떠한 업데이트도 수신되지 않으면, 동기화 서브루틴(1050)은 클라이언트 컴퓨팅 디바이스(110)로부터 메타데이터 서버(123)로 드라이버 동기화 요구(1064)를 통신한다. 숙련된 당업자라면 인식하겠지만, 드라이버 동기화 요구(1064)는 클라이언트 컴퓨팅 디바이스(110)에 설치된 모든 하드웨어를 설명하는 정보 및 설치된 소프트웨어를 설명하는 정보를 포함할 수 있다. 이전의 소프트웨어 동기화 요구들(1051, 1055 및 1060)과 유사하게, 드라이버 동기화 요구(1064)는 설치된 업데이트들을 서버에 통신할 수 있다. 또한, 클라이언트 상에 현재 캐싱되어 있는 드라이버 업데이트들이 있다면, 그것들 전부가 서버에 통신된다.

[0111] 드라이버 동기화 요구(1064)를 수신하는 것에 응답하여, 메타데이터 서버(123)는 클라이언트 컴퓨팅 디바이스(110)에 적용되면서 클라이언트 상에 이미 캐싱되어 있지 않은 드라이버 업데이트들 전부를 송신하는 것으로 응답한다. 드라이버 업데이트는 그것의 필요 조건들 및 조건들이 충족되면 응답(1065)에서 클라이언트 컴퓨팅 디바이스(110)에 송신된다. 드라이버 업데이트들을 통신하는 응답(1065)은 바람직하게는 각각의 업데이트의 명령 컴포넌트를 통신한다. 드라이버 업데이트들은 그 후에 클라이언트 컴퓨팅 디바이스의 업데이트 캐시에 기입된다.

[0112] 드라이버 업데이트들을 포함하는 응답(1065)을 수신한 후에, 동기화 서브루틴(1050)은 수신된 소프트웨어 및 하드웨어 업데이트들 각각의 로컬화된 데이터에 대한 요구(1066)를 송신한다. 위에서 요약 설명한 바와 같이, 각각의 업데이트의 로컬화된 데이터 컴포넌트는 업데이트를 설명하는 일반 정보를 포함한다. 이를테면, 로컬화된 데이터 컴포넌트는 업데이트의 특징들 및 이점들을 설명하는 정보를 포함할 수 있다. 로컬화된 데이터 컴포넌트는 또한 업데이트의 설치 절차에 대한 텍스트 및 설명을 포함할 수 있다. 게다가, 로컬화된 데이터는 업데이트와 관련이 있는 임의의 다른 데이터 또는 정보를 포함할 수 있다.

[0113] 따라서, 수신된 소프트웨어 및 하드웨어 업데이트들 각각의 로컬화된 데이터에 대한 요구(1066)를 수신하면, 메타데이터 서버(123)는 클라이언트의 업데이트 캐시에 저장된 수신된 소프트웨어 업데이트들 및 하드웨어 업데이트들 전부에 대한 로컬화된 데이터 전부를 송신하는 것으로 응답한다. 일단 수신되면, 로컬화된 데이터는 업데이트들 중 어느 것이 설치되어야 할지를 판정하는 소프트웨어 애플리케이션에 의해 처리될 수 있다. 대안적으로, 수신된 로컬화된 데이터는 사용자에게 클라이언트 컴퓨팅 디바이스(110)에게 이용될 수 있는 모든 업데이트들을 통지하도록 사용자에게 디스플레이될 수 있다. 일 실시예에서, 수신된 로컬화된 데이터는 웹 페이지 상에 디스플레이될 수 있다. 본 예에서는, 제6 및 제8 업데이트들(921 및 931)에 대한 로컬화된 데이터가 클라이언트에 의해 수신될 수 있다. 만일 로컬화된 데이터가 기본 업데이트들(911, 912, 913 및 915)에 저장되어 있다면, 이들 업데이트에 대한 로컬화된 데이터도 클라이언트에 의해 수신될 것이다.

[0114] 도 11은 클라이언트에게 이용될 수 있는 업데이트들과 관련된 로컬화된 데이터의 예를 디스플레이하는 웹 페이지(1100)의 일례를 예시한다. 예시의 목적으로, 웹 페이지(1100)는 한 업데이트의 제1 상세 설명(1105) 및 다른 업데이트의 제2 상세 설명(1106)을 포함한다. 또한 도시된 바와 같이, 각각의 업데이트는 업데이트들의 사용자 선택을 수신하기 위한 선택 메커니즘들(1103 및 1104)과 각각 관련되어 있다. 또한 도시된 바와 같이, 웹 페이지(1100)는 사용자가 메타데이터 서버(123) 또는 다운로드 서버(124)와 같은 서버로의 업데이트들의 선택의 통신을 제어할 수 있게 하는 제어 버튼(1101)과 함께 구성될 수 있다.

[0115] 본 발명의 한 특징에서, 클라이언트는 웹 페이지(1100)의 디스플레이를 보강(enhance)하기 위한 다수의 처리를 수행한다. 이를테면, 클라이언트 컴퓨팅 디바이스(110)는 각각의 업데이트의 로컬화된 데이터를 검사하여 특정 업데이트가 높은 우선 순위의 것인지를 판정한다. 그러한 특징은 로컬화된 데이터에 또는 특정 업데이트의 다른 컴포넌트에 그 특정 업데이트가 높은 우선 순위 또는 긴급(emergency) 업데이트임을 나타내는 텍스트를 넣음으로써 용이하게 될 수 있다. 만일 클라이언트 컴퓨팅 디바이스(110)가 높은 우선 순위 또는 긴급 업데이트를 검출하면, 클라이언트는 웹 페이지(1100)의 눈에 띄는 부분, 예컨대 페이지의 상단 부분에 높은 우선 순위 업데이트를 디스플레이한다. 또한, 클라이언트는 업데이트가 높은 우선 순위 업데이트임을 나타내는 전문화된 텍스

트 메시지(1120)와 같은 눈에 띄는 지시자(visual indicator)를 생성할 수 있다.

[0116] 클라이언트 컴퓨팅 디바이스(110)는 또한 각각의 업데이트의 로컬화된 데이터를 검사하여 특정 업데이트가 단독 설치(exclusive installation)를 필요로 하는지, 즉 다른 업데이트의 설치 파일과 동시에 설치될 수 없는 설치 파일을 갖는 업데이트인지를 판정할 수 있다. 그러한 특징은 로컬화된 데이터에 또는 특정 업데이트의 다른 컴포넌트에 그 특정 업데이트가 단독 설치를 필요로 함을 나타내는 텍스트를 넣음으로써 용이하게 될 수 있다. 만일 클라이언트 컴퓨팅 디바이스(110)가 그러한 업데이트를 검출하면, 클라이언트는 예컨대 도 11에 도시된 텍스트 메시지(1122)와 같은 눈의 띄는 지시자를 단독 설치를 필요로 하는 업데이트에 대한 설명과 함께 디스플레이한다.

[0117] 도 7로 돌아가서, 소프트웨어 업데이트 루틴(700)은 블록(708)에서 속행되고, 거기서 클라이언트 컴퓨팅 디바이스(110)는 업데이트들의 선택을 수신한다. 위에서 언급한 바와 같이, 제어 버튼(1101)의 작동에 응답하여, 메타데이터 서버(123) 또는 다운로드 서버(124)에 의해 하나 이상의 업데이트들의 선택이 획득될 수 있다. 일단 하나 이상의 업데이트들의 선택이 수신되면 소프트웨어 업데이트 루틴(700)은 블록(708)에서 속행되고, 거기서 선택된 소프트웨어 업데이트들이 처리된다.

[0118] 본 발명의 또 다른 특징에 따르면, 소프트웨어 업데이트 서비스(120)는 소프트웨어 업데이트 서비스와 클라이언트 컴퓨팅 디바이스(110) 사이에서 정보를 선택하고 송신하는 방법을 제공할 수 있다. 도 12a 및 12b는 본 발명에 따라서 요구된 소프트웨어를 검색 및 설치하도록 클라이언트 컴퓨터 디바이스(110)에 의해 실행되는 소프트웨어 업데이트 처리 서브루틴(1200)을 예시한다. 상술한 바와 같이, 소프트웨어 업데이트 처리 서브루틴(1200)은 일단 소프트웨어 업데이트들의 선택이 생성되거나 수신된 후에 실행될 수 있다. 도 12a와 관련하여, 블록(1202)에서, 업데이트 관리 컴포넌트(111)는 업데이트 에이전트(118)를 인스턴스화한다. 본 발명의 예시적인 실시예에서, 업데이트 에이전트(118)는 요구된 소프트웨어 업데이트를 완료하고, 업데이트 에이전트의 설치 컴포넌트의 요구된 버전을 생성하고, 기존의 파일들을 델타 패치들과 병합하여 업데이트된 파일들을 생성하고, 및/또는 업데이트된 파일들의 설치를 개시하기 위해 어떤 소프트웨어 업데이트 정보가 필요한지를 판정하는 전문화된 소프트웨어 컴포넌트이다. 업데이트 에이전트(118)가 이미 인스턴스화되어 있는 경우에는, 블록(1202)이 생략될 수 있다.

[0119] 블록(1204)에서, 업데이트 에이전트(118)는 업데이트 서비스(120)로부터 소프트웨어 업데이트 정보를 획득한다. 본 발명의 예시적인 실시예에서, 업데이트 서비스(120)에 의해 송신되는 소프트웨어 업데이트 정보는 업데이트 에이전트에 의해 이용될 수 있는 갖가지 데이터를 포함하는, 자동 압축 풀기 파일과 같은, 패키지의 형태로 되어 있다. 본 발명의 한 특징에서, 패키지는 특정 소프트웨어 업데이트에 대응하는 모든 파일들의 목록을 포함할 수 있다. 또한, 패키지는 업데이트될 특정 버전의 파일들을 업데이트 서비스(120) 상의 패치 저장 파일에 저장된 대응하는 소프트웨어 업데이트 델타 패치에 맵핑하는 패치 저장 매니페스트(patch storage manifest)의 적어도 일부의 카피를 포함할 수 있다. 패키지는 또한 설치를 완료하는 데 필요한 설치 컴포넌트의 버전의 ID를 포함할 수 있는 업데이트될 각각의 파일에 대한 설치 정보를 포함할 수 있다. 게다가, 패키지는 또한 업데이트 에이전트(118)에 대한 설치 컴포넌트 또는 클라이언트 컴퓨팅 디바이스(110) 상에 이미 저장되어 있는 설치 컴포넌트의 버전을 업데이트하는 델타 패치를 포함할 수 있다. 더욱이, 패키지는 업데이트 에이전트가 소프트웨어 업데이트가 성공했는지를 판정할 수 있게 하는 검증 정보(verification information)를 포함할 수 있다. 예를 들면, 검증 정보는 비교를 위한 업데이트된 파일에 대한 기준 해시 값들을 포함할 수 있다. 업데이트 에이전트(118)는 또한 패키지의 컨텐츠를 검증할 수 있다.

[0120] 판정 블록(1206)에서, 업데이트 에이전트(118)가 업데이트를 실행하기 위해 설치 컴포넌트의 버전을 업데이트할 필요가 있는지를 판정하는 테스트가 수행된다. 숙련된 당업자라면 자동 압축 풀기 파일 내의 설치 컴포넌트의 완전한 카피의 송신은 각각의 소프트웨어 업데이트에 대해 업데이트 서비스(120)에 의해 송신되는 데이터의 양을 증가시킬 수 있다는 것을 인식할 것이다. 따라서, 본 발명의 예시적인 실시예에서는, 설치 컴포넌트의 기준선 버전(baseline version)이 클라이언트 컴퓨팅 디바이스에 저장되어 설치 컴포넌트 델타 패치를 통하여 현재의 소프트웨어 업데이트의 필요 조건에 대해 특정하게 업데이트될 수 있다. 따라서, 자동 압축 풀기 파일 내의 설치 정보는 임의의 포함된 설치 컴포넌트 업데이트들이 클라이언트 컴퓨팅 디바이스(110) 상의 설치 컴포넌트의 기준선 버전과 병합될 필요가 있는지의 여부를 업데이트 에이전트(118)에게 지시한다. 만일 업데이트가 필요하다면, 블록(1208)에서, 업데이트 에이전트(118)는 기준선 설치 컴포넌트를 업데이트한다. 이에 대해서는 아래에서 도 13과 관련하여 더 상세히 설명하겠다.

[0121] 일단 업데이트 에이전트가 설치 컴포넌트를 업데이트하였거나 또는 설치 컴포넌트가 업데이트를 필요로 하지 않

는다면, 블록(1210)에서, 업데이트 에이전트(118)는 클라이언트 컴퓨팅 디바이스(110) 상에 설치된 파일들의 목록 및 파일의 특정 버전을 수행한다. 본 발명의 예시적인 실시예에서, 업데이트 에이전트(118)는 패키지에서 선택된 업데이트에 대응하는 것으로 식별된 모든 파일들을 얻기 위하여 클라이언트 컴퓨팅 디바이스(110)를 조회(query)할 수 있다. 대안적으로, 만일 업데이트 에이전트(118)가 최근에 목록을 수행하였다면, 목록의 캐싱된 버전이 이용될 수 있다. 블록(1212)에서, 업데이트 에이전트(118)는 요구된 업데이트를 완료하기 위해 어떤 소프트웨어 업데이트 정보가 필요한지를 식별한다. 본 발명의 예시적인 실시예에서, 패치 저장 매니페스트는 요구된 델타 패치로의 설치된 파일들의 버전의 맵핑을 포함한다. 따라서, 델타 패칭이 이용 가능하다면, 업데이트 에이전트(118)는 맵핑을 이용하여 특정 델타 패치 및 패치 저장 파일 내의 그것의 오프셋 위치를 식별할 것이다. 대안적으로, 델타 패칭이 이용 가능하지 않거나 실행될 수 없다면, 업데이트 에이전트(118)는 다운로드를 위한 전체 파일을 식별할 수 있다.

[0122] 이제 도 12b와 관련하여, 블록(1214)에서, 업데이트 에이전트는 식별된 소프트웨어 업데이트 정보에 대한 요구를 송신한다. 본 발명의 예시적인 실시예에서, 업데이트 에이전트(118)는 패치 저장 파일로부터 요구되는 패치들의 특정 범위를 업데이트 서비스(120)의 다운로드 서버(124)에게 지시함으로써 특정 델타 패치들에 대한 요구를 송신할 수 있다. 상술한 바와 같이, 패치 저장 파일은 다수의 적용 가능한 델타 패치들을 포함하고, 각각의 델타 패치는 패치 저장 파일과의 그것의 위치에 의해 식별된다. 패치 저장 파일은 몇몇 실시예들에서 다소 클 수 있기 때문에, 업데이트 에이전트(118)는 패치 저장 매니페스트로부터 지시된 패치 저장 파일 내의 특정 위치들로부터의 데이터만을 요구하는 요구를 이용할 수 있다. 본 발명의 다른 실시예에서, 업데이트 에이전트(118)는 업데이트 파일의 전체 카피 및/또는 패치 저장 파일의 완전한 카피를 요구할 수 있다.

[0123] 본 발명의 다른 실시예에서, 업데이트 서비스(120)와 배타적으로 관련되지 않을 수 있는 다른 다운로드 서버가 업데이트 에이전트(118) 요구를 처리할 수 있다. 이 실시예에서, 요구 패치 저장 파일은, 전부 또는 일부가, 네트워크 상의 임의의 수의 부가적인 다운로드 서버들에 송신될 수 있다. 부가적인 다운로드 서버들은 전용 네트워크(private network) 상의 클라이언트들을 업데이트하는 데 이용되는 전용 네트워크의 일부일 수 있다. 또한, 부가적인 다운로드 서버는 공용 네트워크의 일부일 수 있다. 전용 네트워크 환경에서, 다운로드 서버들은 클라이언트 요구들을 처리하기 위하여 패치 저장 파일의 완전한 카피를 획득할 수 있다. 대안적으로, 다운로드 서버들은 또한 다른 클라이언트들로부터의 이전의 데이터 요구들을 처리할 때 패치 저장 파일의 부분들을 캐싱하고 그 캐시 데이터를 이용하여 다운로드를 충족시킬 수 있다. 따라서, 부가적인 다운로드 서버들은 업데이트 서비스(120)의 다운로드 서버(124)의 통신 부담을 저감시킬 수 있다.

[0124] 블록(1216)에서, 업데이트 에이전트(118)는 요구된 업데이트 정보를 수신한다. 본 발명의 예시적인 실시예에서, 요구된 업데이트 정보는 2가지 방법으로 송신될 수 있다. 수동 업데이트로 불리는 첫 번째 방법에서는, 업데이트 요구는 직접 HTTP 데이터 전달 응답에 대한 요구와 함께 업데이트 서비스(120)에 송신된다. 이 방법에서는, 업데이트 서비스(120)는 요구된 데이터를 업데이트 에이전트(118)에 송신하는 데 이용할 수 있는 전체 대역폭의 전부를 이용할 수 있다. 자동 업데이트로 불리는 두 번째 방법에서는, 업데이트 요구는 간접 HTTP 데이터 전달 응답에 대한 요구와 함께 업데이트 서비스(120)에 송신된다. 이 응답에서는, 업데이트 서비스(120)는 요구된 데이터를 배경 처리로서 송신한다. 배경 처리는 최소량의 이용 가능한 대역폭을 이용하도록 실행될 수 있다. 또한, 배경 처리는 다운로드 처리 중에 중단되고 다음 이용 가능한 시간에 재개될 수 있다. 본 발명에 참고로 통합된, "네트워크를 통해 데이터를 전송하는 시스템 및 방법"(System and Method for Transferring Data Over a Network)이라는 명칭으로 2000년 2월 16일자로 출원된 공동 양도되고 공동 진행중인 미국 특허 출원 번호 제09/505,735호에는 배경 처리에 의하여 요구된 데이터를 송신하는 시스템 및 방법에 대한 설명이 기재되어 있다. 숙련된 당업자라면 전경 또는 배경 데이터 전달이 반드시 선택된 소프트웨어 업데이트의 우선 순위를 반영하지는 않고, 그보다는 업데이트 정보를 획득하기 위해 어떻게 대역폭이 할당되는지를 반영한다는 것을 인식할 것이다.

[0125] 일단 요구된 정보가 업데이트 서비스로부터 수신되면, 블록(1218)에서, 업데이트 에이전트(118)는 델타 패치를 대응하는 설치된 파일들과 병합한다. 본 발명의 예시적인 실시예에서, 업데이트 에이전트(118)는 다운로드 및 병합 처리 중에 선택된 파일들이 변하지 않도록 하기 위해 설치된 파일의 오리지널 버전을 캐싱할 수 있다. 또한, 캐싱된 설치된 파일의 오리지널 버전은 선택된 업데이트를 제거(uninstall)하기 위해 사용될 수 있다.

[0126] 판정 블록(1220)에서, 업데이트 파일이 유효한지를 판정하는 테스트가 수행된다. 본 발명의 예시적인 실시예에서, 업데이트 에이전트(118)는 해싱 알고리즘을 이용하여 업데이트 정보로부터 획득되고 유효 파일 업데이트에 대응하는 기준 해시 값을 현재의 수정된 파일로부터의 해시와 비교할 수 있다. 만일 해시들이 일치하지 않으면, 현재의 수정된 파일은 유효하지 않다. 숙련된 당업자라면 다수의 대안적인 유효성 확인 알고리즘들 중

임의의 하나가 또한 이용될 수 있다는 것을 인식할 것이다. 만일 업데이트된 파일이 유효하지 않으면, 서브루틴(1200)은 블록(1214)으로 복귀하고, 거기서 업데이트 에이전트는 업데이트 정보를 다시 요구할 수 있다. 대안적으로, 만일 업데이트 에이전트(118)가 수회에 걸쳐 업데이트 파일을 생성하려는 시도가 실패하면, 업데이트 에이전트는 수 개의 고장 조치 절차들 중 하나를 실행할 수 있다. 본 발명의 일 실시예에서, 업데이트 에이전트(118)는 패치 저장 파일에 저장되고 패치 저장 매니페스트로부터 식별된 업데이트된 파일의 완전한 카피를 업데이트 서비스(120)로부터 요구할 수 있다. 본 발명의 다른 실시예에서, 업데이트 에이전트(118)는 자체 완비된 파일 내의 업데이트된 파일의 카피를 업데이트 서비스(120)로부터 요구할 수 있다. 본 발명의 또 다른 실시예에서, 서브루틴(1200)은 그렇지 않으면 실패할 수 있다.

[0127] 일단 선택된 파일이 유효하면, 판정 블록(1222)에서, 임의의 부가적인 다운로드들이 필요한지를 판정하는 테스트가 수행된다. 본 발명의 예시적인 실시예에서, 서브루틴(1200)은 이전에 선택된 다운로드의 완료 후에 부가적인 다운로드들을 계속해서 체크하는 반복 루프에 들어간다. 만일 다운로드 중에 파일의 상태가 변하면, 업데이트 에이전트(118)는 새로운 상태 변경을 위해 계속해서 부가적인 다운로드를 요구할 것이다. 만일 부가적인 다운로드가 필요하면, 블록(1224)에서, 업데이트 에이전트(118)는 다른 목록을 수행하고 모든 적용 가능한 델타 패치들을 식별한다. 그 후에 서브루틴(1200)은 블록(1214)으로 복귀한다.

[0128] 일단 모든 요구된 업데이트 다운로드들이 완료되면, 판정 블록(1226)에서, 클라이언트 머신의 상태가 변하였는지를 판정하는 테스트가 수행된다. 본 발명의 예시적인 실시예에서, 업데이트 정보의 다운로드 및 병합과 업데이트된 파일의 실제 설치 사이에 시간이 경과할 수 있다. 따라서, 업데이트된 파일을 설치하기에 앞서서, 업데이트 에이전트는 클라이언트 컴퓨팅 디바이스 상태가 변하였는지를 판정한다. 만일 상태가 변하였다면, 파일 업데이트는 유효하지 않을 수 있고 블록(1228)에서 업데이트가 실패한다. 대안적으로, 만일 어떠한 상태 변화도 일어나지 않았다면, 업데이트 에이전트(118)는 블록(1230)에서 업데이트된 파일을 설치하고 서브루틴(1200)은 블록(1232)에서 복귀한다.

[0129] 이제 도 13과 관련하여, 블록(1208)(도 12a)에 대응하는 기준선 설치 컴포넌트를 업데이트하기 위해 클라이언트 컴퓨팅 디바이스(110)에 의해 실행되는 서브루틴(1300)에 대해 설명하겠다. 판정 블록(1302)에서, 업데이트 서비스(120)로부터 업데이트 에이전트(118)에 송신된 자동 압축 풀기 파일 내에 새로운 기준선 설치 컴포넌트가 포함되어 있는지를 판정하는 테스트가 수행된다. 본 발명의 예시적인 실시예에서, 만일 기준선 설치 관리자(baseline installer)를 업데이트하는 데 필요한 델타 패치들이 사이즈에서 업데이트된 설치 컴포넌트의 송신에 필적하면, 새로운 기준선 설치 컴포넌트가 송신될 것이다. 만일 업데이트된 설치 컴포넌트가 포함되어 있다면, 블록(1304)에서, 업데이트 에이전트는 업데이트된 기준선 설치 컴포넌트를 새로운 설치 컴포넌트로서 설치한다. 게다가, 새로운 업데이트된 설치 컴포넌트는 부가적인 업데이트를 위한 기준선 설치 관리자로서 역할을 하도록 클라이언트 컴퓨팅 디바이스(110) 메모리에 저장될 수 있다. 블록(1306)에서, 서브루틴은 복귀한다.

[0130] 만일 업데이트된 기준선 설치 컴포넌트가 자동 압축 풀기 파일에 포함되어 있지 않다면, 블록(1308)에서, 업데이트 에이전트(118)는 자동 압축 풀기 파일로부터 기준선 설치 컴포넌트 델타 패치를 획득한다. 본 발명의 예시적인 실시예에서, 기준선 설치 컴포넌트 델타 패치는 업데이트된 기준선 설치 컴포넌트를 생성하기 위해 기준선 설치 컴포넌트와 병합될 수 있는 소프트웨어 코드에 대응한다. 따라서, 블록(1310)에서, 업데이트 에이전트는 기준선 설치 컴포넌트 델타 패치를 기준선 설치 컴포넌트와 병합한다. 그 후에 블록(1312)에서, 업데이트 에이전트(118)는 업데이트된 기준선 설치 컴포넌트를 현재의 설치 컴포넌트로 지정한다. 본 발명의 예시적인 실시예에서, 업데이트된 설치 컴포넌트는 설치가 완료된 후에 저장되지 않을 것이다. 이 실시예에 따르면, 업데이트 에이전트(118)는 클라이언트 컴퓨팅 디바이스(110) 메모리에 제한된 수의 기준선 설치 컴포넌트들만을 유지한다. 따라서, 업데이트 에이전트는 각각의 설치 시에 임시의 업데이트된 설치 컴포넌트를 생성한다. 각각의 클라이언트 컴퓨팅 디바이스(110)는 제한된 수의 기준선 설치 컴포넌트들에만 대응할 수 있기 때문에, 업데이트 서비스(120)는 각각의 클라이언트 컴퓨팅 디바이스에 대한 단일 기준선 설치 컴포넌트 델타 패치를 송신하는 데에만 필요하다. 블록(1314)에서, 서브루틴(1300)은 복귀한다.

[0131] 본 발명의 바람직한 실시예를 예시하고 설명하였지만, 발명의 사상 및 범위를 벗어나지 않고 갖가지 변경이 행해질 수 있다는 것을 인식할 것이다. 이를테면, 비록 본 명세서에서 설명된 예들은 소프트웨어 업데이트에 적용되지만, 본 발명의 범위는 소프트웨어 업데이트와 관련된 정보의 배포 및 통신을 넘어서 다른 용도들을 포함한다. 따라서, 이 개시 내용에서 특정 주제 내용이 명백히 배제되지 않는 한은, 본 발명의 범위는 소프트웨어 업데이트 이외의, 또는 그것에 더하여, 임의의 타입의 데이터의 배포 및 통신에 적용된다는 것을 인식해야 할 것이다.

도면의 간단한 설명

[0022] 본 발명의 전술한 특징들 및 다수의 부수적인 이점들은 이하의 상세한 설명을 첨부 도면들과 관련하여 참조함으로써 더 쉽게 인식되고 더 잘 이해될 것이다.

[0023] 도 1은 본 발명에 따라서 업데이트 소프트웨어를 제공하는 업데이트 서비스 및 클라이언트 컴퓨터를 포함하는 소프트웨어 업데이트 시스템의 블록도이다.

[0024] 도 2는 본 발명에 따라서 업데이트 서비스에 의한 클라이언트 컴퓨팅 디바이스의 인증을 예시하는 도 1의 소프트웨어 업데이트 시스템의 블록도이다.

[0025] 도 3은 본 발명에 따라서 클라이언트 컴퓨팅 디바이스와 업데이트 서비스 사이의 이용 가능한 업데이트들의 동기화를 예시하는 도 1의 소프트웨어 업데이트 시스템의 블록도이다.

[0026] 도 4는 본 발명에 따라서 업데이트 서비스로부터 클라이언트 컴퓨팅 디바이스로의 소프트웨어 업데이트 정보의 송신을 예시하는 도 1의 소프트웨어 업데이트 시스템의 블록도이다.

[0027] 도 5는 본 발명에 따라서 클라이언트 컴퓨팅 디바이스에 의한 업데이트 정보의 처리 및 선택을 예시하는 도 1의 소프트웨어 업데이트 시스템의 블록도이다.

[0028] 도 6은 본 발명에 따라서 클라이언트 컴퓨팅 디바이스에 의한 업데이트된 파일들의 설치 및 델타 패치들의 병합을 예시하는 도 1의 소프트웨어 업데이트 시스템의 블록도이다.

[0029] 도 7은 본 발명에 따라서 클라이언트 컴퓨팅 디바이스 상에 설치하기 위해 이용 가능한 소프트웨어 업데이트들을 식별하기 위해 클라이언트 컴퓨팅 디바이스와 업데이트 서비스에 의해 실행되는 소프트웨어 업데이트 루틴을 예시하는 흐름도이다.

[0030] 도 8은 본 발명에 따라서 업데이트 서비스 상에 저장된 업데이트들에의 선택적 액세스를 제공하기 위한 인가 루틴의 프로토콜 도면이다.

[0031] 도 9는 본 발명에 따라서 인가 루틴을 예시하는 소프트웨어 업데이트들의 세트 예의 블록도이다.

[0032] 도 10은 본 발명에 따라서 소프트웨어 업데이트 서비스로부터 클라이언트 컴퓨팅 디바이스로 소프트웨어 업데이트들의 선택 그룹을 통신하기 위한 동기화 루틴의 프로토콜 도면이다.

[0033] 도 11은 본 발명에 따라서 개개의 클라이언트 컴퓨팅 디바이스에게 이용될 수 있는 소프트웨어 업데이트들의 리스트를 디스플레이하기 위한 그래픽 사용자 인터페이스의 전형적인 부분을 예시하는 블록도이다.

[0034] 도 12a 및 12b는 본 발명에 따라서 요구된 소프트웨어를 검색 및 설치하기 위해 클라이언트 컴퓨팅 디바이스(110)에 의해 실행되는 소프트웨어 업데이트 처리 서브루틴(1200)을 예시하는 흐름도이다.

[0035] 도 13은 본 발명에 따라서 기준선(baseline) 설치 컴포넌트를 업데이트하기 위해 클라이언트 컴퓨팅 디바이스에 의해 실행되는 서브루틴을 예시하는 흐름도이다.

[0036] 〈도면의 주요 부분에 대한 부호의 설명〉

[0037] 120 : 업데이트 서비스

[0038] 121 : 업데이트 서버

[0039] 122 : 인가 서버

[0040] 123 : 메타데이터 서버

[0041] 124 : 다운로드 서버

[0042] 101 : 네트워크

[0043] 110 : 클라이언트

[0044] 112 : 업데이트 관리 컴포넌트

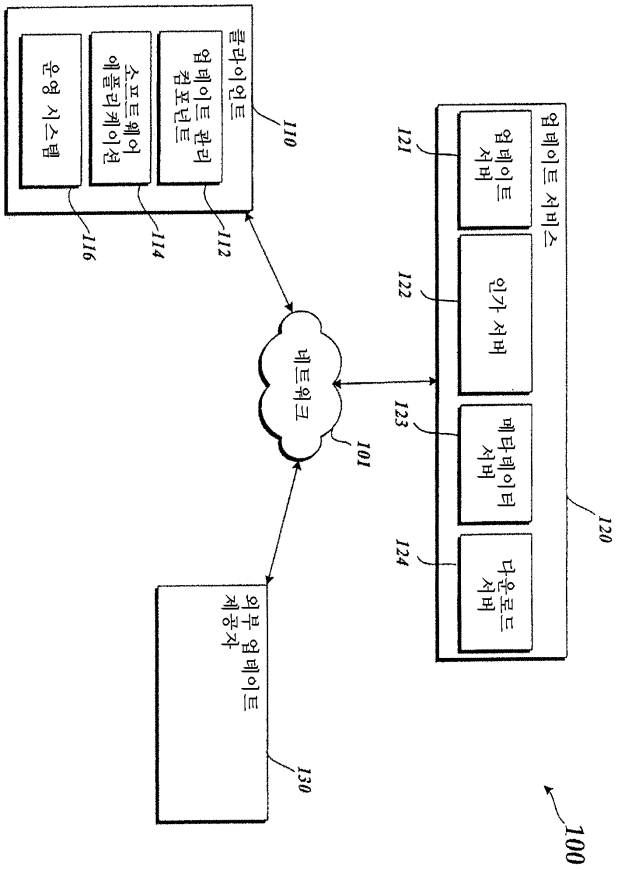
[0045] 114 : 소프트웨어 애플리케이션

[0046] 116 : 운영 시스템

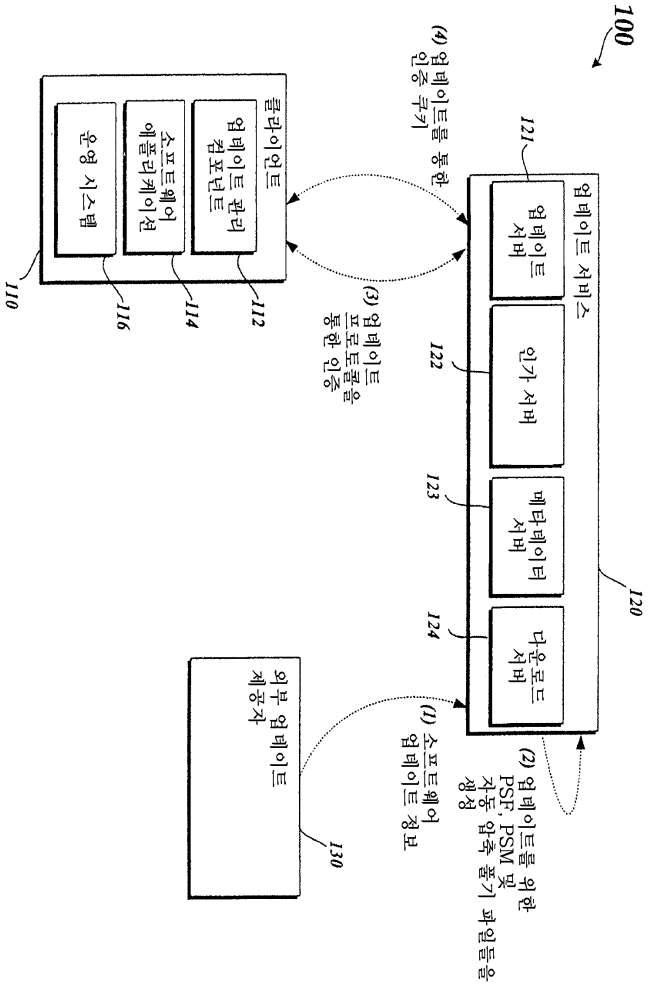
[0047] 130 : 외부 업데이트 제공자

도면

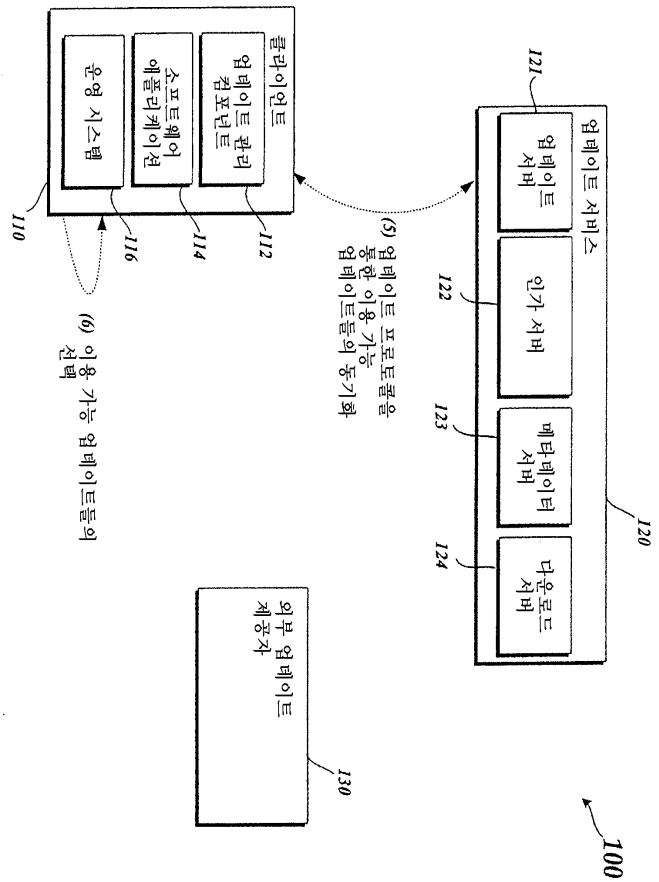
도면1



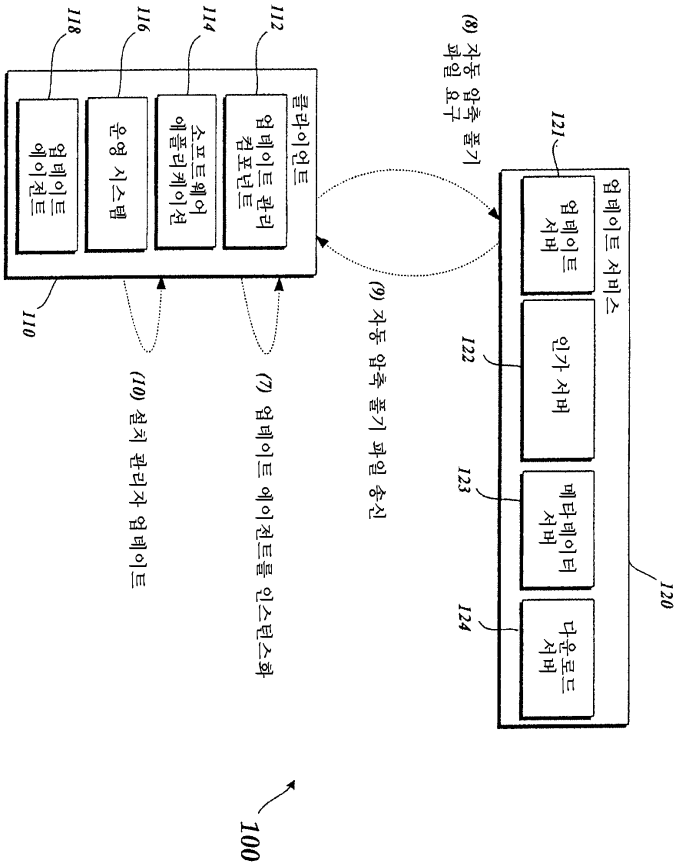
도면2


100
업데이트 서비스
121
업데이트 처버
인가 서버
122
메타데이터 서버
123
다운로드 서버
124
120
(1) 소프트웨어 업데이트 정보
(2) 업데이트를 위한 PSP, PSM 및 압축 풀기 파일들을 자동 생성
(3) 업데이트 프로토콜을 통한 인증
(4) 업데이트를 통한 인증 쿠키
클라이언트
업데이트 관리 컴포넌트
112
소프트웨어 애플리케이션
114
운영 시스템
116
110
외부 업데이트 제공자
130

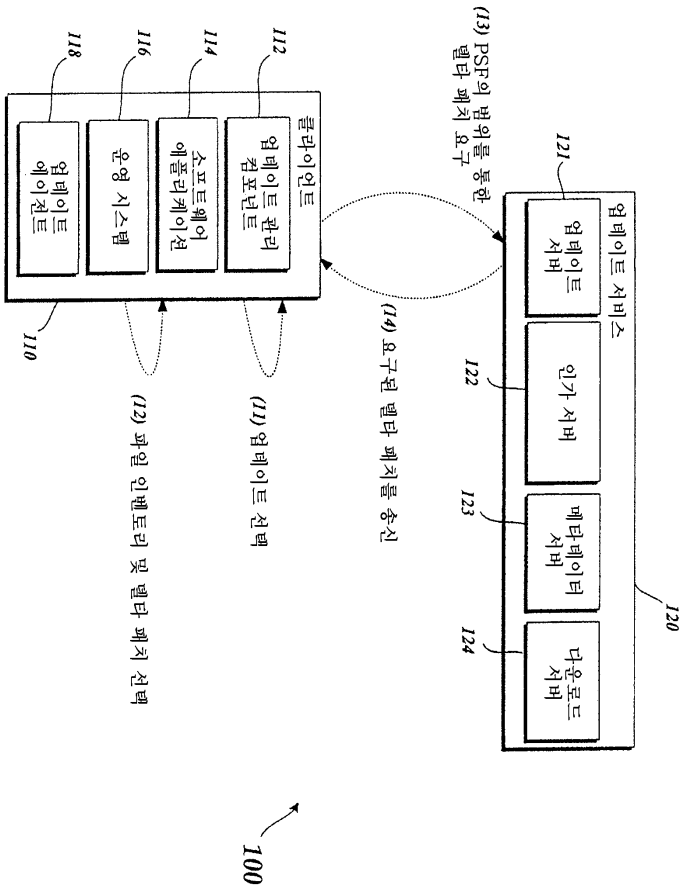
도면3


업데이트 서비스
업데이트 처버
인가 서버
메타데이터 처버
다운로드 처버
121
122
123
124
120
100
클라이언트
업데이트 관리 컴포넌트
소프트웨어 애플리케이션
운영 시스템
112
114
116
110
(5) 업데이트 프로토콜을 통한 이용 가능 업데이트들의 동기화
(6) 이용 가능 업데이트들의 선택
외부 업데이트 제공자
130

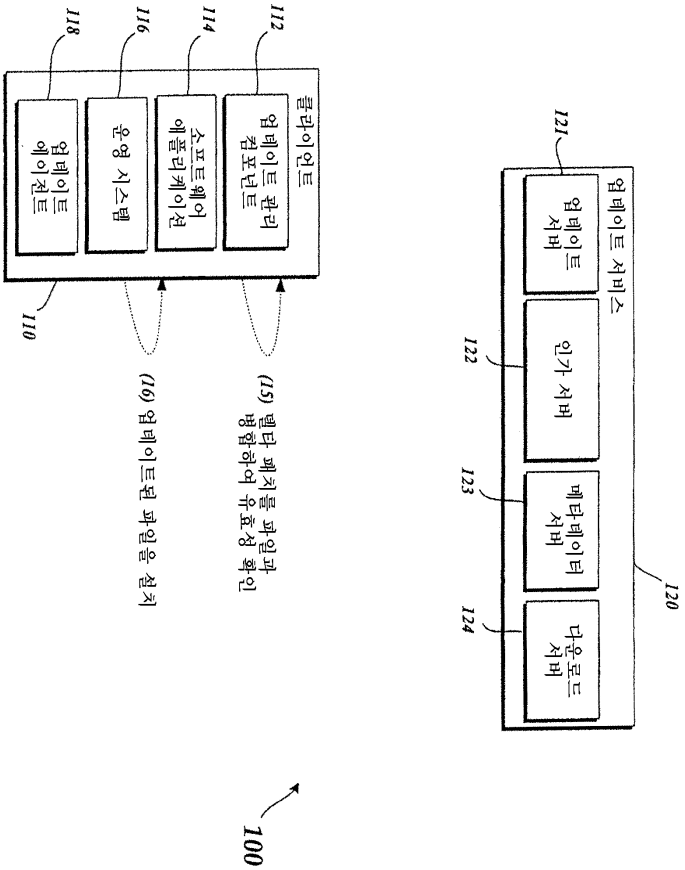
도면4


100
110
클라이언트
112
업데이트 관리 컴포넌트
114
소프트웨어 애플리케이션
116
운영 시스템
118
업데이트 에이전트
(7) 업데이트 에이전트를 인스턴스화
(8) 자동 압축 풀기 파일 요구
(9) 자동 압축 풀기 파일 송신
(10) 설치 관리자 업데이트
업데이트 서비스
120
121
업데이트 서버
122
인가 서버
123
메타데이터 서버
124
다운로드 서버

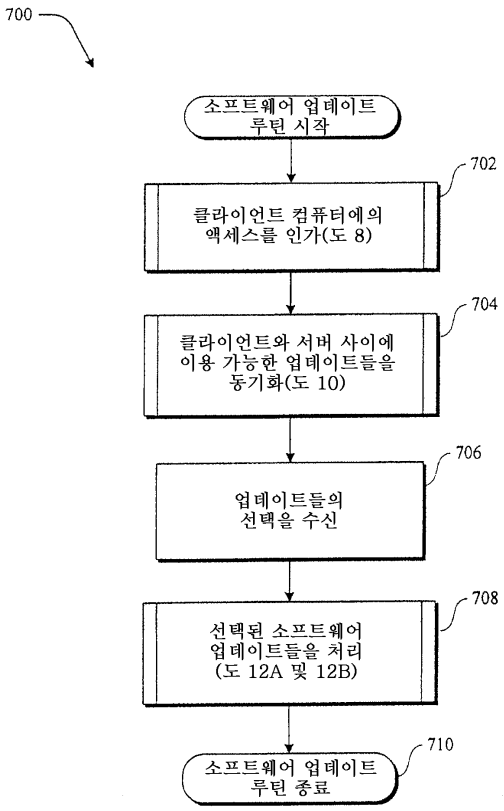
도면5


100
110
112
114
116
118
클라이언트
업데이트 관리 컴포넌트
소프트웨어 애플리케이션
운영 시스템
업데이트 에이전트
120
121
122
123
124
업데이트 서버
인가 서버
메타데이터 서버
다운로드 서버
(11) 업데이트 선택
(12) 파일 인벤토리 및 델타 패치 선택
(13) PSP의 범위를 통한 델타 패치 요구
(14) 요구된 델타 패치를 송신

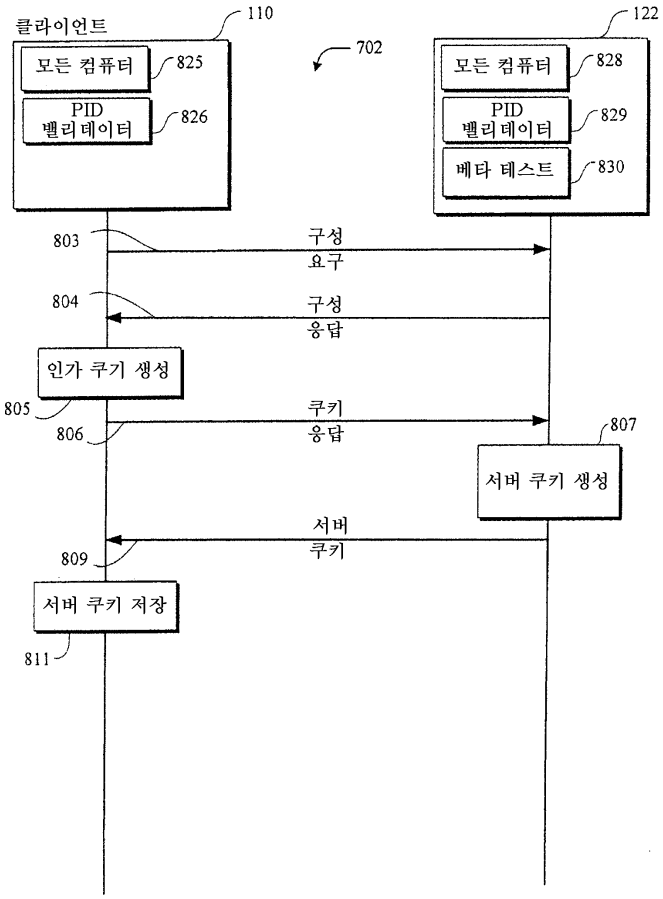
도면6


클라이언트
업데이트 관리 컴포넌트
112
소프트웨어 애플리케이션
114
운영 시스템
116
업데이트 에이전트
118
110
(15) 델타 패치를 파일과 병합하여 유효성 확인
(16) 업데이트된 파일을 설치
100
업데이트 서비스
121
업데이트 서버
인가 서버
122
메타데이터 서버
123
다운로드 서버
124
120

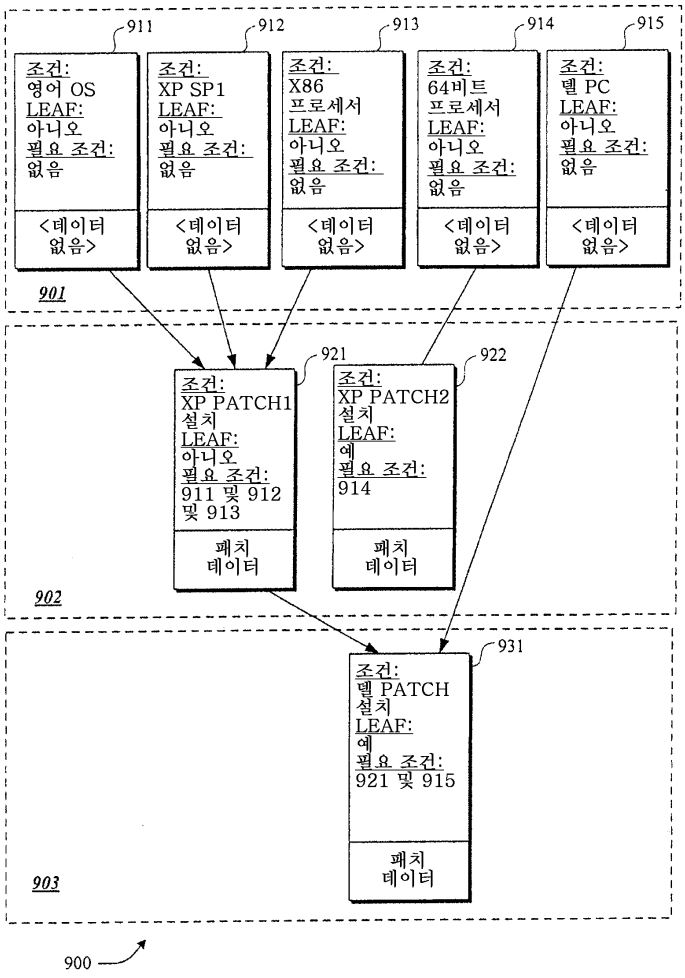
도면7


700
소프트웨어 업데이트 루틴 시작
702
클라이언트 컴퓨터에의 액세스를 인가(도 8)
704
클라이언트와 서버 사이에 이용 가능한 업데이트들을 동기화(도 10)
706
업데이트들의 선택을 수신
708
선택된 소프트웨어 업데이트들을 처리 (도 12A 및 12B)
710
소프트웨어 업데이트 루틴 종료

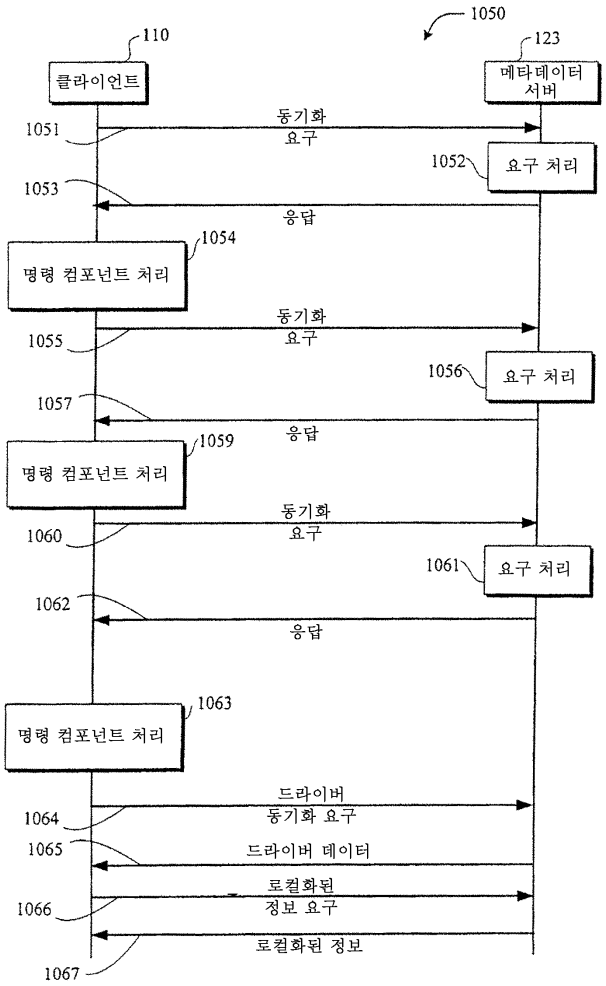
도면8


클라이언트
110
모든 컴퓨터
825
PID
밸리데이터
826
702
122
모든 컴퓨터
828
PID
밸리데이터
829
베타 테스트
830
803
구성
요구
804
구성
응답
인가 쿠기 생성
805
806
쿠키
응답
807
서버 쿠키 생성
서버
쿠키
809
서버 쿠키 저장
811

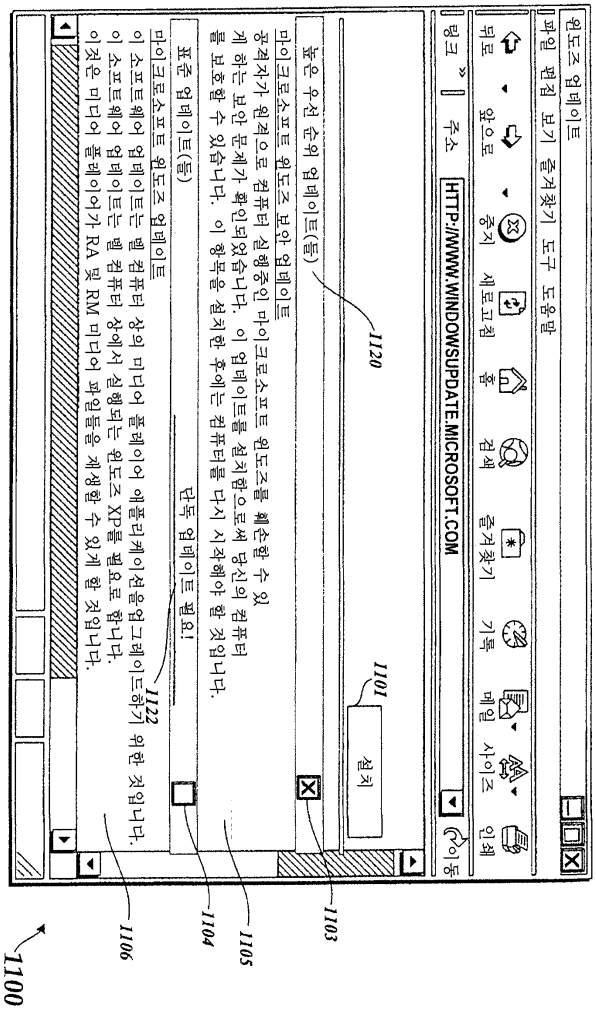
도면9


911
912
913
914
915
조건:
영어 OS
LEAF:
아니오
필요 조건:
없음
조건:
XP SP1
LEAF:
아니오
필요 조건:
없음
조건:
X86
프로세서
LEAF:
아니오
필요 조건:
없음
조건:
64비트
프로세서
LEAF:
아니오
필요 조건:
없음
조건:
델 PC
LEAF:
아니오
필요 조건:
없음
<데이터 없음>
<데이터 없음>
<데이터 없음>
<데이터 없음>
<데이터 없음>
901
921
922
조건:
XP PATCH1
설치
LEAF:
아니오
필요 조건:
911 및 912 및 913
조건:
XP PATCH2
설치
LEAF:
예
필요 조건:
914
패치 데이터
패치 데이터
902
931
조건:
델 PATCH
설치
LEAF:
예
필요 조건:
921 및 915
패치 데이터
903
900

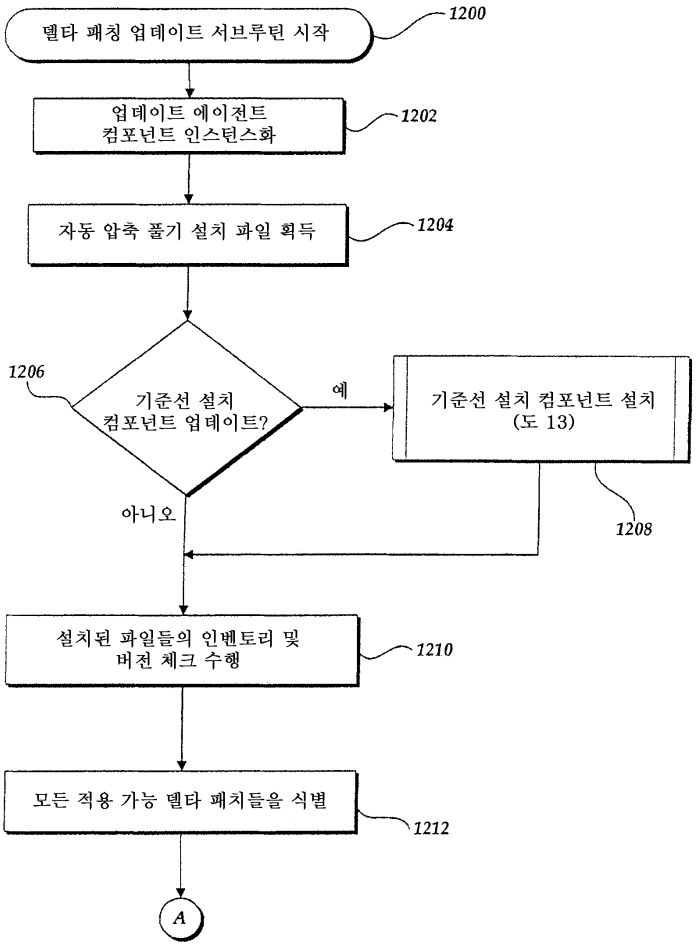
도면10


1050
110
클라이언트
123
메타데이터 서버
1051
동기화 요구
1052
요구 처리
1053
응답
1054
명령 컴포넌트 처리
1055
동기화 요구
1056
요구 처리
1057
응답
1059
명령 컴포넌트 처리
1060
동기화 요구
1061
요구 처리
1062
응답
1063
명령 컴포넌트 처리
1064
드라이버 동기화 요구
1065
드라이버 데이터
1066
로컬화된 정보 요구
1067
로컬화된 정보

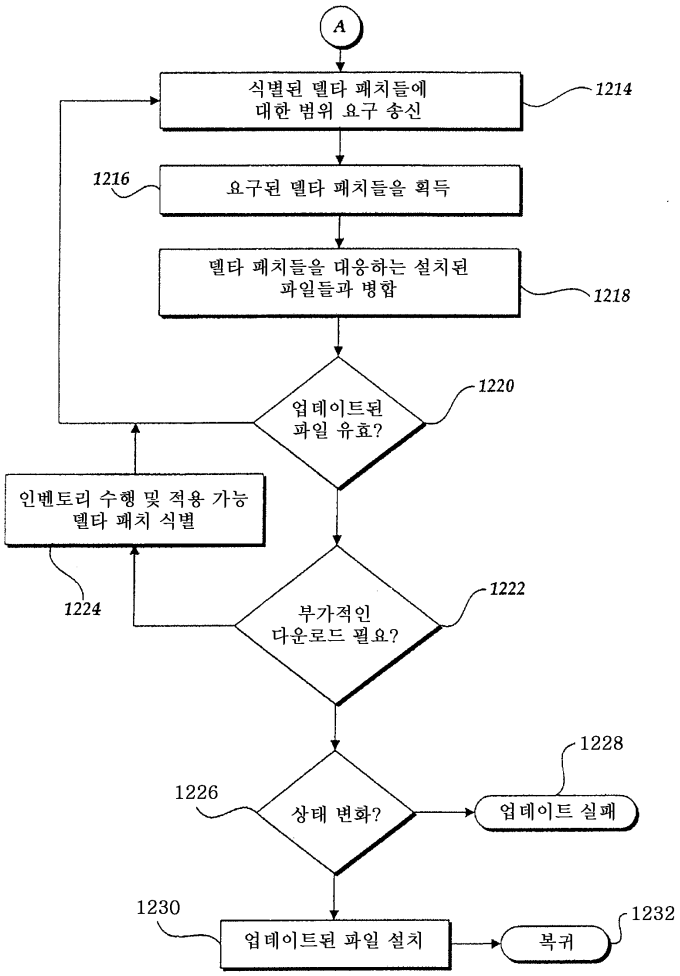
도면11


윈도즈 업데이트
파일 편집 보기 즐겨찾기 도구 도움말
뒤로 앞으로 중지 새로고침 홈 검색 즐겨찾기 기록 메일 사이즈 인쇄
링크 주소 HTTP://WWW.WINDOWSUPDATE.MICROSOFT.COM 이동
1120
1101
설치
높은 우선 순위 업데이트(들)
마이크로소프트 윈도즈 보안 업데이트
공격자가 원격으로 컴퓨터 실행중인 마이크로소프트 윈도즈를 훼손할 수 있
게 하는 보안 문제가 확인되었습니다. 이 업데이트를 설치함으로써 당신의 컴퓨터
를 보호할 수 있습니다. 이 항목을 설치한 후에는 컴퓨터를 다시 시작해야 할 것입니다.
표준 업데이트(들)
단독 업데이트 필요!
1122
마이크로소프트 윈도즈 업데이트
이 소프트웨어 업데이트는 델 컴퓨터 상의 미디어 플레이어 애플리케이션을업그레이드하기 위한 것입니다.
이 소프트웨어 업데이트는 델 컴퓨터 상에서 실행되는 윈도즈 XP를 필요로 합니다.
이것은 미디어 플레이어가 RA 및 RM 미디어 파일들을 재생할 수 있게 할 것입니다.
1103
1104
1105
1106
1100

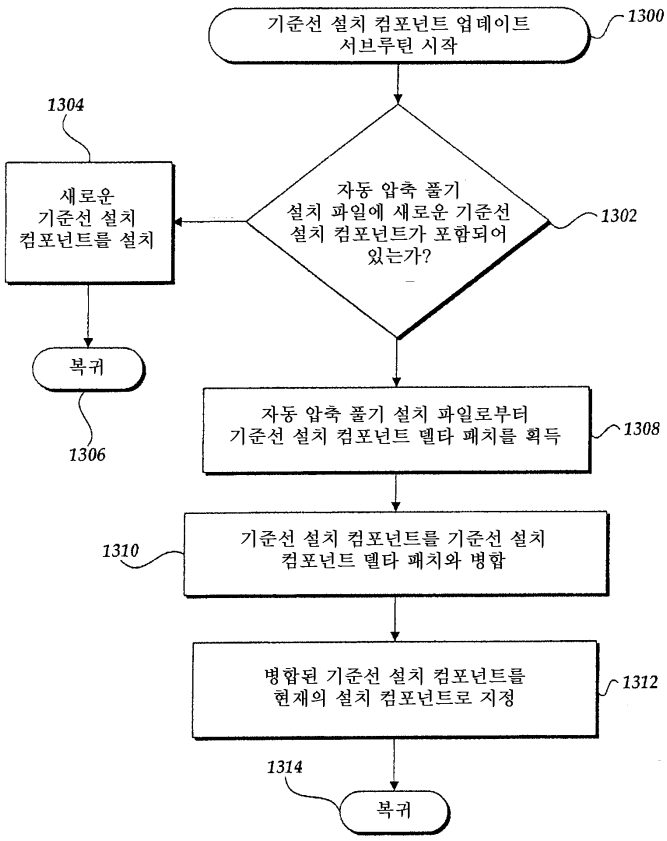
도면12a


델타 패칭 업데이트 서브루틴 시작
1200
업데이트 에이전트 컴포넌트 인스턴스화
1202
자동 압축 풀기 설치 파일 획득
1204
1206
기준선 설치 컴포넌트 업데이트?
예
기준선 설치 컴포넌트 설치 (도 13)
1208
아니오
설치된 파일들의 인벤토리 및 버전 체크 수행
1210
모든 적용 가능 델타 패치들을 식별
1212
A

도면12b


A
식별된 델타 패치들에 대한 범위 요구 송신
1214
1216
요구된 델타 패치들을 획득
델타 패치들을 대응하는 설치된 파일들과 병합
1218
업데이트된 파일 유효?
1220
인벤토리 수행 및 적용 가능 델타 패치 식별
1224
부가적인 다운로드 필요?
1222
1228
1226
상태 변화?
업데이트 실패
1230
업데이트된 파일 설치
복귀
1232

도면13


기준선 설치 컴포넌트 업데이트 서브루틴 시작
1300
자동 압축 풀기 설치 파일에 새로운 기준선 설치 컴포넌트가 포함되어 있는가?
1302
1304
새로운 기준선 설치 컴포넌트를 설치
복귀
1306
자동 압축 풀기 설치 파일로부터 기준선 설치 컴포넌트 델타 패치를 획득
1308
기준선 설치 컴포넌트를 기준선 설치 컴포넌트 델타 패치와 병합
1310
병합된 기준선 설치 컴포넌트를 현재의 설치 컴포넌트로 지정
1312
1314
복귀