(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2013-0143089
(43) 공개일자 2013년12월30일

(51) 국제특허분류(Int. Cl.)
G06F 9/445 (2006.01)
(21) 출원번호 10-2013-7015751
(22) 출원일자(국제) 2011년11월18일
심사청구일자 없음
(85) 번역문제출일자 2013년06월18일
(86) 국제출원번호 PCT/US2011/061560
(87) 국제공개번호 WO 2012/068570
국제공개일자 2012년05월24일
(30) 우선권주장
61/415,243 2010년11월18일 미국(US)

(71) 출원인
구글 인코포레이티드
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(72) 발명자
지트코프 존 니콜라스
미국 캘리포니아주 94301 팔로 알토 엠바카데로 로드 505
리우 칸
미국 캘리포니아주 94040 마운틴 뷰 링컨 스트리트 927
(74) 대리인
박장원, 특허법인태평양

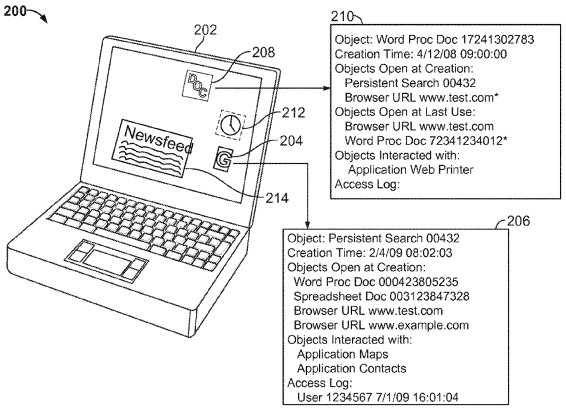
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **컴퓨팅 객체의 콘텍스트 히스토리**

(57) 요 약

컴퓨터 운영체제에 대한 다양한 특징들은, 웹 브라우저 형태의 단일 기본 애플리케이션이 운영체제에 대해 존재하고 다른 모든 애플리케이션들은 그 브라우저 애플리케이션 상에서 웹 앱으로서 실행되는 경우에, 운영을 위한 메커니즘들을 포함한다. 컴퓨터로 구현되는 객체 추적 방법은, 초기에, 컴퓨팅 디바이스 상의 운영체제 객체를 인스턴스화(instantiating) 하는 단계; 상기 운영체제 객체가 인스턴스화될 때, 상기 인스턴스화된 운영체제 객체 외에, 컴퓨팅 디바이스 상에 오픈되어 있는 객체들의 상태를 정의하는 콘텍스트 메타 데이터(contextual meta data)를 자동으로 식별하는 단계; 및 상기 식별된 콘텍스트 메타 데이터를 상기 운영체제 객체와 연관시켜 저장하는 단계를 포함하고, 상기 콘텍스트 메타 데이터는 상기 운영체제 객체가 인스턴스화될 때 상기 운영체제 내에서 활성화된 하나 이상의 객체들을 식별한다.

대 표 도



특허청구의 범위

청구항 1

컴퓨터로 구현되는 객체 추적 방법에 있어서,

초기에, 컴퓨팅 디바이스 상의 운영체제 객체를 인스턴스화(instantiating) 하는 단계;

상기 운영체제 객체가 인스턴스화될 때, 상기 인스턴스화된 운영체제 객체 외에, 컴퓨팅 디바이스 상에 오픈되어 있는 객체들의 상태를 정의하는 콘텍스트 메타 데이터(contextual meta data)를 자동으로 식별하는 단계; 및

상기 식별된 콘텍스트 메타 데이터를 상기 운영체제 객체와 연관시켜 저장하는 단계를 포함하고,

상기 콘텍스트 메타 데이터는 상기 운영체제 객체가 인스턴스화될 때 상기 운영체제 내에서 활성화된 하나 이상의 객체들을 식별하는 것을 특징으로 하는 방법.

청구항 2

청구항 1에 있어서,

상기 운영체제가 클로즈 되었을 때 상기 운영체제를 인스턴스화하는 후속 요청을 수신하고, 콘텍스트 메타 데이터에서 식별된 2차 객체들을 호출하는 단계를 더 포함하는 방법.

청구항 3

청구항 2에 있어서,

상기 2차 객체들을 호출하는 단계는, 상기 운영체제 객체가 최초로 인스턴스화될 때 오픈되었던 하나 이상의 애플리케이션들을 실행하는 단계를 포함하는 방법.

청구항 4

청구항 1에 있어서,

상기 저장된 콘텍스트 메타 데이터에, 상기 후속 요청이 수신될 때 활성화된 객체들에 대응하는 정보를 부가하는 단계를 더 포함하는 방법.

청구항 5

청구항 1에 있어서,

상기 운영체제 객체가 인스턴스화되는 동안 상기 운영체제 객체 외의 객체들과 사용자 상호작용을 식별하고, 상기 사용자 상호작용에 대한 정보를 상기 저장된 콘텍스트 메타 데이터에 부가하는 단계를 더 포함하는 방법.

청구항 6

청구항 1에 있어서,

상기 컴퓨팅 디바이스 상에 오픈된 상기 객체들의 상태를 정의하는 상기 콘텍스트 메타 데이터는, 상기 운영체제 객체가 인스턴스화 되는 동안 상기 컴퓨팅 디바이스를 사용하는 하나 이상의 사람을 식별하는 것을 특징으로 하는 방법.

청구항 7

청구항 6에 있어서,

상기 운영체제가 클로즈 되었을 때 상기 운용체제를 인스턴스화하는 후속 요청을 수신하고, 콘텍스트 메타 데이터에서 식별된 2차 객체들을 호출하는 단계를 더 포함하는 방법.

청구항 8

청구항 6에 있어서,

상기 운영체제 객체가 클로즈 되었을 때 상기 콘텍스트 메타 데이터에서 식별된 하나 이상의 사람이 존재하는 것의 표시(indication)를 수신하고, 상기 표시의 수신에 응답하여 상기 운영체제 객체를 인스턴스화하는 단계를 더 포함하는 방법.

청구항 9

청구항 6에 있어서,

상기 운영체제 객체가 클로즈 되었을 때 상기 콘텍스트 메타 데이터에서 식별된 둘 이상의 사람이 존재하는 것의 표시(indication)를 수신하고, 상기 표시의 수신에 응답하여 상기 운영체제 객체를 인스턴스화 하는 단계를 더 포함하는 방법.

청구항 10

컴퓨터로 구현되는 객체 추적 시스템에 있어서, 상기 시스템은,

컴퓨터 운영체제;

상기 컴퓨터 운영체제에 의해 정의되고 상기 운영체제를 실행하는 디바이스의 사용자에게 접근 가능한 복수개의 객체들; 및

각각의 상기 객체들에 대한 콘텍스트 메타 데이터의 복수개의 인스턴스(instances)를 포함하는 콘텍스트 정보 데이터 스토어를 포함하고,

상기 콘텍스트 메타 데이터의 각 인스턴스는 하나 이상의 상기 객체들과 연관되고,

상기 콘텍스트 메타 데이터의 각 인스턴스는 상기 콘텍스트 메타 데이터와 연관되어 있는 객체가 상기 운영체제에서 최초로 인스턴스화 될 때 상기 운영체제에서 활성화 되었던 하나 이상의 객체들을 식별하는 시스템.

청구항 11

청구항 10에 있어서,

상기 콘텍스트 메타 데이터의 하나 이상의 인스턴스들은, 상기 콘텍스트 메타 데이터의 상기 하나 이상의 인스턴스들과 연관된 상기 객체들이 최초 인스턴스화 후에 상기 운영체제에서 인스턴스화 되었을 때 활성화되었던 하나 이상의 객체들을 식별하는 것을 특징으로 하는 시스템.

청구항 12

청구항 10에 있어서,

상기 콘텍스트 메타 데이터의 하나 이상의 인스턴스들은, 상기 콘텍스트 메타 데이터의 상기 하나 이상의 인스턴스들과 연관된 상기 객체가 활성화되었던 기간 동안 상기 운영체제에서, 상기 콘텍스트 데이터의 상기 하나 이상의 인스턴스들과 연관된 객체 외의 객체에 대하여 취해진 액션을 식별하는 것을 특징으로 하는 시스템.

청구항 13

컴퓨터로 구현되는 객체 추적 시스템에 있어서, 상기 시스템은,

컴퓨터 운영체제;

상기 컴퓨터 운영체제에 의해 정의되고 상기 운영체제를 실행하는 디바이스의 사용자에게 접근 가능한 복수개의 객체들; 및

콘텍스트 메타 데이터의 인스턴스를 각각의 복수개의 객체들과 연관시키기 위한 수단을 포함하고,

상기 콘텍스트 메타 데이터의 각 인스턴스는 상기 콘텍스트 메타 데이터와 연관되어 있는 객체가 상기 운영체제에서 최초로 인스턴스화 될 때 상기 운영체제에서 활성화 되었던 하나 이상의 객체들을 식별하는 시스템.

청구항 14

컴퓨터로 구현된 객체 추적 방법에 있어서,

컴퓨팅 디바이스 상에서 오픈된 제1 운영체제 객체 상에서 제1 미리 정의된 액션의 제1 발생을 기록하는 단계;

상기 제1 운영체제 객체와 동시에 상기 컴퓨팅 디바이스 상에서 오픈된 제2 운영체제 객체 상에서 제2 미리 정의된 액션의 제1 발생을 기록하는 단계;

동시에 발생한 상기 제1 및 제2 미리 정의된 액션들을 연관시키는 단계;

상기 제1 및 제2 운영체제 객체와 관련된 상기 연관된 액션들을 포함하는 연관된 상태를 저장하는 단계; 및

연관된 상태들을 저장하기 위한 데이터베이스를 유지하는 단계를 포함하는 방법.

청구항 15

청구항 14에 있어서,

상기 제1 운영체제 객체 상에서 상기 제1 미리 정의된 액션의 제2 발생을 식별하는 것을 포함하는 상기 컴퓨팅 디바이스 상의 액션들을 모니터링 하는 단계; 및

상기 데이터베이스에서 상기 저장된 연관된 상태를 검색하는 단계; 및

상기 제1 미리 정의된 액션의 상기 제2 발생을 식별하는 단계에 응답하여 상기 제2 운영체제 객체를 인스턴스화 하는 단계를 더 포함하는 방법.

청구항 16

청구항 14에 있어서,

상기 제1 운영체제 객체가 클로즈될 때 상기 제1 운영체제 객체를 인스턴스화 하는 후속 요청을 수신하는 단계; 및

상기 제1 운영체제 객체를 인스턴스화 하는 후속 요청을 상기 데이터베이스 내의 연관된 상태들과 비교하는 단계를 더 포함하는 방법.

청구항 17

청구항 16에 있어서,

상기 비교에 기초하여, 상기 제2 운영체제 객체를 상기 제1 운영체제 객체에 관련된 것으로 식별하는 단계; 및

상기 제2 운영체제 객체를 인스턴스화 하는 단계를 더 포함하는 방법.

청구항 18

청구항 14에 있어서,

상기 제2 운영체제 객체 상의 상기 제2 미리 정의된 액션은 상기 제2 운영체제 객체를 클로즈 하기 위한 요청인 것을 특징으로 하는 방법.

청구항 19

청구항 18에 있어서,

상기 제1 운영체제 상에서 상기 제1 미리 정의된 액션의 제2 발생을 식별하는 것을 포함하는 상기 컴퓨팅 디바이스 상의 액션들을 모니터링 하는 단계;

상기 데이터베이스에서 상기 저장된 연관된 상태를 검색하는 단계; 및

만약 상기 제2 운영체제 객체가 오픈되어 있으면, 상기 제2 운영체제 객체를 클로즈하는 단계를 더 포함하는 방법.

청구항 20

청구항 15에 있어서,

상기 제2 운영체제 객체를 인스턴스화 하는 단계 이전에, 상기 제2 운영체제 객체를 인스턴스화 하기 위한 사용

자 허가(permission)를 요청하는 단계를 더 포함하는 방법.

명 세 서

기 술 분 야

[0001] 본 출원은 2010년 11월 18일자로 출원된 미국 가출원 US 61/415,243 "컴퓨터 운영체제에서의 사용자 상호작용(User Interaction in a Computer Operating System)"에 대하여 우선권을 주장하며, 그 내용은 여기에 모두 참조로서 통합된다.

[0002] 본 발명은 컴퓨터 운영 체제의 사용자들과 상호작용하기 위한 시스템 및 기술과 관련된다.

배 경 기 술

[0003] 컴퓨터(예를 들어, 데스크탑 PC, 랩탑, 넷북, 태블릿, 또는 스마트폰)를 작동시키는 핵심 구조(core structures)는 기본적인 입출력 시스템, 또는 바이오스(BIOS), 운영체제, 및 컴퓨터를 작동시키기 위한 기본 기능을 제공하는 소프트웨어 "스택(stack)"을 사용하는 것으로 참조될 수 있는 다른 구성요소(components)들을 포함할 수 있다. BIOS는 컴퓨터가 처음 켜지고 부팅될 때 접근(access) 될 수 있고, 컴퓨터를 부팅하기 위해 추가적으로 필요한 코드를 저장하는 저장 디바이스를 포함하여, 시스템 디바이스들을 식별, 테스트, 및 초기화하는 것과 같은 기본 기능을 제공할 수 있다. 운영체제를 위한 코드는 추가 디바이스 등에 저장될 수 있고, 부팅이 완료되어 동작할 준비가 될 때까지 컴퓨터의 추가적인 부팅을 제공할 수 있다.

[0004] 운영체제는 일반적으로 컴퓨터 하드웨어와 컴퓨터의 사용자, 및 하드웨어와 컴퓨터 상에 사용자가 로딩하는 애플리케이션 사이의 인터페이스로서 기능한다. 운영체제는 다양한 기능을 제공할 수 있다. 예를 들어, 운영체제는 컴퓨터의 사용자가 컴퓨터로부터 출력을 수신하고 컴퓨터로 입력을 제공할 수 있는 그래픽 사용자 인터페이스(graphical user interface; GUI)를 제공할 수 있다. 또한, 운영체제는 다양한 서드파티(third-party) 애플리케이션들이 실행되는, 운영체제가 이 애플리케이션에 필요한 서비스들을 제공하고 또한 애플리케이션들이 다른 애플리케이션, 주변 장치(예를 들어, 프린터, 카메라, 등)와 같은 다른 리소스 및 운영체제 자체에서 제공되는 서비스와 통신할 수 있는 메커니즘을 제공하는, 플랫폼을 제공할 수도 있다.

발명의 내용

해결하려는 과제

[0005] 본 명세서는 하나의 컴퓨팅 디바이스 또는 다수의 다른 컴퓨팅 디바이스를 포함하는 시스템의 운영체제의 일부로서 구현될 수 있는 시스템 및 기술을 설명한다. 예를 들어, 다양한 메커니즘이 컴퓨팅 디바이스를 클라우드 기반 시스템(cloud-based system)에 저장된 데이터와 동기화하기 위해 이용될 수 있는데, 호스트 컴퓨터 서버 시스템(a hosted computer server system)이 일반 구성원들이 그 시스템에 접근하는 것을 이용 가능하게 하고, 그 시스템은 데이터 저장 및 백업, 문서 저장, 이메일 라우팅 및 핸들링, 및 다른 유용한 서비스와 같은 다양한 서비스들을 제공한다. 컴퓨팅 디바이스는 상대적으로 로컬 스토리지(local storage)를 거의 갖지 않고, 따라서 사용자 데이터를 호스트 서버 시스템에 저장하는 형태일 수 있다. 부가하여, 상기 디바이스는 본질적으로 지속해서 네트워크(예를 들어, 무선 네트워크) 및 네트워크를 통해 인터넷에 연결되도록 구성될 수 있다. 결과적으로, 디바이스 상의 다양한 구성요소들은 거의 항상 인터넷에 접속된 접근(a nearly always-on approach)에 따라 작동하도록 마련될 수 있다.

[0006] 예를 들어, 이하에서 설명되는 어떤 실시예에서는, 컴퓨터는 어떤 객체(objects)(예를 들어, 애플리케이션, 프로그램, 파일)가 동시에 사용되고 있는지를 기록하기 위해서 제시간에 다양한 포인트에서 "스냅샷(snapshots)"을 찍을 수 있다. 이 "스냅샷"으로부터, 사용자는 특정 객체들이 그 또는 그녀의 컴퓨터 상에서 동시에 실행되는 것을 선호한다는 추론(inference)이 이루어질 수 있다. 예를 들어, 사용자는 워드 프로세스 문서(word processing document)에 대한 업무 중에 음악을 듣는 것을 선호할 수 있다. 이러한 경우에, 사용자가 워드 프로세스 문서를 열었을 때, 컴퓨터는 음악 프로그램과 워드 프로세스 프로그램(word processing program) 모두 동시에 열린다는 다수의 "스냅샷"이 촬영되었다는 것을 인식할 것이다. 또 다른 예시에서, 사용자는 특정 워드 프로세스 문서에 대한 업무 중일 수 있다. 이전에 기록된 "스냅샷"은 워드 프로세스 문서가 생성되었을 때, 컴퓨

터상에서 지속성 검색 객체(a persistent search object) 또한 활성화 된다는 것을 지시할 수 있다. 이러한 동시발생(concurrence)은 상기 지속성 검색에 의해 생성된 정보를 담고 있는 상기 문서가 사용자에 의해 생성되었다는 것을 지시할 수 있다. 이것과 관련하여 더욱 강력한 추론이, 예를 들어, 만약 사용자가 지속성 검색 결과로부터 상기 정보를 클립보드에 복사하고, 문서를 생성하고, 그러고 나서 검색 결과 또는 다른 복사된 데이터를 상기 문서에 붙여 넣기 한다면, 형성될 수 있다. 이 정보는 또한 워드 프로세스 문서와 연관되어 메타 데이터로서 저장될 수 있다.

[0007] 보다 구체적으로, 전자 레코더(an electronic recorder)는 컴퓨팅 디바이스 운영체제의 일부로서 구현될 수 있다. 레코더는 디바이스 상에서 프로세스들을 모니터 할 수 있고, 디바이스 상에 로그온 하기 위해 관련된 활동들(relevant activities)로 고려되는 활동 리스트를 참고하기 위해(to consult) 프로그램 될 수 있다. 예를 들어, 디바이스 상의 어떤 애플리케이션의 인스턴스화(instantiation), 미디어 파일의 로딩(예를 들어, 노래 또는 영화의 플레이), 또는 채팅 세션(chat session)과 같은 커뮤니케이션 세션의 시작, 및 다른 그러한 이벤트들이 레코더에 의해 체크될 수 있다. 레코더가 관련 있는 이벤트를 인식하면, 어느 다른 애플리케이션이 열렸는지, 어떤 웹 페이지가 현재 로드 되는지, 어떤 다른 사용자와 현재 통신중인지와 같은 것을 기록하는 것과 같이, 디바이스의 현재 상태의 스냅샷을 취할 수 있다. 레코더는 현재 상태에 대하여 그러한 포인트들을 기록(log)할 수 있고, 시스템 상에서 그와 같은 다른 이벤트들이 인식되면 반복적으로 수행할 수도 있다. 또한 실질적으로 동시에, 그와 같은 이벤트가 발생하면, 시스템은 발생한 동일 또는 유사 이벤트의 이전 인스턴스(previous instances)에 대한 기록(log)을 검색할 수 있고, 그 이벤트들이 발생했을 때 디바이스의 세부 상태를 식별할 수 있다. 시스템은 이제 자동으로 어떤 세부사항들을 인스턴스화 할(instantiate) 수 있다. (예를 들어, 이전에 실행되었던 애플리케이션을 실행(open), 이전에 오픈된 웹 페이지 또는 다른 콘텐트 소스(content sources)를 오픈, 이전에 대화가 발생했던 사용자들로 연락할 수 있다.) 대안으로, 운영체제(operating system)는 이전 상태에 대한 그러한 특징들(such aspects of prior state) 리스트를 디스플레이 할 수 있고, 사용자는 그 특징들 중 하나 이상을 선택할 수 있고, 시스템은 이제 선택된 특징들을 인스턴스화 할 수 있다.

과제의 해결 수단

[0008] 일부 실시예에서, 컴퓨터로 구현되는 객체 추적 방법은, 초기에, 컴퓨팅 디바이스 상의 운영체제 객체를 인스턴스화(instantiating) 하는 단계; 상기 운영체제 객체가 인스턴스화될 때, 상기 인스턴스화된 운영체제 객체 외에, 컴퓨팅 디바이스 상에 오픈되어 있는 객체들의 상태를 정의하는 콘텍스트 메타 데이터(contextual meta data)를 자동으로 식별하는 단계; 및 상기 식별된 콘텍스트 메타 데이터를 상기 운영체제 객체와 연관시켜 저장하는 단계를 포함하고, 상기 콘텍스트 메타 데이터는 상기 운영체제 객체가 인스턴스화될 때 상기 운영체제 내에서 활성화된 하나 이상의 객체들을 식별한다. 일부 실시예들은 부가적으로 상기 운영체제가 클로즈 되었을 때 상기 운용체제를 인스턴스화하는 후속 요청을 수신하고, 콘텍스트 메타 데이터에서 식별된 2차 객체들을 호출하는 단계를 더 포함한다.

[0009] 일부 경우에, 2차 객체들을 호출하는 단계는, 상기 운영체제 객체가 최초로 인스턴스화될 때 오픈되었던 하나 이상의 애플리케이션들을 실행하는 단계를 포함한다. 상기 방법은 상기 저장된 콘텍스트 메타 데이터에, 상기 후속 요청이 수신될 때 활성화된 객체들에 대응하는 정보를 부가하는 단계를 더 포함할 수 있다. 부가적으로, 상기 방법은 상기 운영체제 객체가 인스턴스화되는 동안 상기 운영체제 객체 외의 객체들과 사용자 상호작용을 식별하는 단계 및, 상기 사용자 상호작용에 대한 정보를 상기 저장된 콘텍스트 메타 데이터에 부가하는 단계를 더 포함할 수 있다.

발명의 효과

[0010] 특정 실시예에서, 이러한 시스템 및 기술은 하나 이상의 이점을 제공할 수 있다. 예를 들어, 컴퓨터 시스템과의 사용자 상호작용은 컴퓨터 시스템을 사용할 때 사용자에게 그 또는 그녀의 선호를 제출할 것을 명확하게 요구하지 않고 사용자의 개인적인 선호에 관한 더 많은 정보를 제공할 수 있다. 대신에, 그러한 선호는 행동적 경향을 통해 추론될 수 있다. 부가적으로, 컴퓨터가 사용자가 동시에 특정 객체들이 열리는 것을 선호한다는 것을 인식할 수 있고, 그녀가 첫 번째 객체를 오픈한 후에 자동으로 사용자를 위해 다른 객체들을 오픈할 것이기 때문에, 사용자는 다수의 객체들 대신 하나의 객체를 열기 위한 클릭만을 함으로써 시간을 절약할 수 있다.

[0011] 하나 이상의 실시예들의 세부 사항이 첨부된 도면 및 이하의 설명에서 개시된다. 다른 특징 및 이점들은 명세서 및 도면, 그리고 청구항으로부터 명백해질 것이다.

도면의 간단한 설명

[0012] 도 1은 컴퓨팅 디바이스의 지연 잠금(delayed locking)을 제공하기 위한 시스템의 개념도이다.

도 2a는 콘텍스트 객체를 사용하는 운영체제의 개념 다이어그램이다.

도 2b는 콘텍스트 객체를 사용하는 운영 체제의 대안적인 실시예의 개념 다이어그램이다.

도 3은 컴퓨팅 디바이스 상에서 메모리 제어(memory control)를 유지하기 위한 시스템의 개념 다이어그램이다.

도 4는 컴퓨터 프로세스 사이의 메시지 패싱(message passing)에 쓰레드 관련성(thread affinity)을 제공하는 시스템의 개념 다이어그램이다.

도 5는 상태 비 보존형 환경에서(in a stateless environment) 상태 정보를 제공하는 시스템의 개념 다이어그램이다.

도 6은 네트워크를 통해 컴퓨팅 디바이스에 대한 이미지를 제공하는 시스템의 개념 다이어그램이다.

도 7은 컴퓨팅 디바이스의 원격 모니터링 및 제어를 제공하는 시스템(700)의 개념 다이어그램이다.

도 8은 호스트 컴퓨터 시스템 상에 집중적으로 저장된 컴퓨팅 디바이스의 데이터에 대한 캐싱(caching)을 제공하기 위한 시스템의 개념 다이어그램이다.

도 9는 컴퓨팅 디바이스의 지연 잠금을 제공하기 위한 프로세스의 흐름도이다.

도 10a는 운영체제에서 콘텍스트 객체를 관리하기 위한 프로세스의 흐름도이다.

도 10b는 운영체제에서 콘텍스트 객체를 관리하기 위한 프로세스의 흐름도이다.

도 11은 컴퓨팅 디바이스 상에서 메모리 제어를 유지하기 위한 프로세스의 흐름도이다.

도 12는 컴퓨터 프로세스 사이의 메시지 패싱(message passing)에 쓰레드 관련성을 제공하기 위한 프로세스의 흐름도이다.

도 13은 상태 비 보존형 환경에서 상태 정보를 제공하기 위한 프로세스의 흐름도이다.

도 14는 네트워크를 통해 컴퓨팅 디바이스에 대한 이미지를 제공하는 프로세스의 흐름도이다.

도 15는 컴퓨팅 디바이스의 제어 및 원격 모니터링을 제공하기 위한 프로세스의 흐름도이다.

도 16은 호스트 컴퓨터 시스템 상에 집중적으로 저장된 컴퓨팅 디바이스의 데이터에 대한 캐싱(caching)을 제공하기 위한 프로세스의 흐름도이다.

도 17은, 여기에 설명된 기술이 이용될 수 있는 컴퓨터 디바이스 및 모바일 컴퓨터 디바이스의 예시를 나타낸다.

다양한 도면에서 동일한 기호는 동일한 구성을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0013] 본 명세서는 컴퓨팅 디바이스 상에서 실행되는, 예를 들어 운영체제의 구성요소를 이용하는 모바일 스마트폰과 같은, 디바이스의 사용자와의 상호작용을 제공하기 위한 시스템 및 기술을 설명한다. 상기 시스템 및 기술은 하나 이상의 호스팅 서버 시스템들과 통신하고 그로부터 제공될 수 있는 다양한 운영체제 컴포넌트(components)를 제공할 수 있다. 특히, 운영체제는 실행되는 대부분의 시간에 인터넷 연결과 함께 구동되도록 설계될 수 있다. 결과적으로, 운영체제에 의하여 수행되는 다수의 동작들은 네트워크 연결이 가능하다는 것을 전제하고, 네트워크 연결이 복구될 때까지 캐싱 기술(caching techniques) 또는 다른 브리지 접근(bridging approaches)에 의존하도록 설계될 수 있다. 특히, 여기에서 기술되는 디바이스들은 인터넷에 연결하기 위해 하나 이상의 셀룰러 전화 네트워크(cellular telephone networks)의 데이터 부분(data portion)과 통신하는 거의 항상 연결된(nearly always-connected) 무선 데이터 인터페이스를 가질 수 있다.

[0014] 도 1은 컴퓨팅 디바이스의 지연 잠금(delayed locking)을 제공하기 위한 시스템의 개념도이다. 일반적으로, 상기 시스템은 휴대 컴퓨팅 디바이스의 사용자가, 즉시 그 디바이스를 셧다운 하지 않고도, 휴대 컴퓨팅 디바이스를 셧다운(shut down)하기 위한 액션을 취할 수 있도록 한다. 결과적으로, 만약 사용자가 그러한 액션을 취한 후에 곧 그 또는 그녀의 마음을 바꾼다면, 사용자는 만약 그 디바이스가 사용자가 그러한 액션을 취하자마자 즉시 셧다운 되었다면 요구되었을 다른 단계들을 취할 필요 없이 그 디바이스를 재-활성화(re-activate)할 수 있다

[0015] 이 특징에서, 휴대 컴퓨팅 디바이스의 3가지 다른 상태, 이 경우에는 클램쉘 디자인(clamshell design)을 갖는 랩탑 컴퓨터가 도시된다. 제 1 단계(102)에서, 컴퓨터는 오픈되어 작동 중이고, 시간은 12시 정오이다. 컴퓨터 상의 디스플레이(108)는 잠금 타이머(lock timer)가 10초로 설정되어 있는 것을 나타낸다. 여기서 디스플레이(108)는 예시의 형태로 제공되고, 실제 사용에 있어서 디바이스는 타이머 상에 남은 양을 디스플레이 하지 않을 수 있고, 타이머는 상태(102)에 도시된 포인트에서 동작을 시작하지 않을 수 있다.

[0016] 상태(104)에서, 5초 후에, 디스플레이는 컴퓨터의 아래 부분으로 닫혀-컴퓨터로 하여금 즉시 동면(hibernate) 또는 다른 형태의 비 활성화 모드를 시작하도록 하는 액션-있다. 이러한 변화는 컴퓨터 상의 마이크로프로세서(microprocessor)의 전원을 차단하는 것(powering down), 디스플레이를 끄는 것(turning off), 냉각 팬(cooling fan) 또는 다른 관련 메커니즘을 끄는 것, GPU(graphical processing unit)의 전원을 차단하는 것, 및 디바이스 상의 다른 전력 절감 기술의 수행을 포함할 것이다. 상태(104)에 의해 도시되지는 않았으나, 단지 몇 초 동안 닫혀 있었을 때 디바이스는 완전히 전원이 차단되지 않았거나 전원 차단을 시작하지 않았을 수 있는데, 이는 이 예시에서 디바이스가 셧다운 시퀀스를 시작하기 전에 10초의 빌트인 딜레이(built-in delay)를 가지고 있기 때문이다. (또한 상기 시퀀스가 시작된 후 잠금 해제 비밀번호의 입력과 같은 실질적인 사용자 개입 없이 디바이스가 재활성화 될 수 있는 시점(point)을 통과할 때까지의 어떤 정해지지 않은 시간이 있을 수도 있다.) 대신에, 사용자에게 디바이스가 전원 차단 중이라는 인상을 주기 위하여, 디스플레이(108)를 전환하는 것 등에 의하여 제한된 수의 특징들이 전력 차단될 수 있다. 다만 이 시점에서, 디바이스를 재-활성화 하는 것은 디바이스를 다시 여는 것 이상의 무엇을 요구하지 않을 수 있다.

[0017] 14초 후 및 디바이스를 닫고 9초 후, 상태(106)에서, 예를 들면 디바이스의 사용자가 디바이스의 사용 중단을 원치 않고 대신에 디바이스를 이용하여 추가 작업을 수행하는 것을 필요로 하기 때문에, 디바이스는 오픈된다. 타이머가 10초 후에 만료되도록 설정되어 있었기 때문에, 디바이스는 아직 동면 또는 다른 전력 차단 상태(powered down state)로 이행되지 않았다. 대신에, 파워 백업을 위해 실질적인 시간을 소모하는 디바이스 상의 모든 시스템들은 전원 공급 상태를 유지하고, 디스플레이(108)와 같은 오직 제한된 시스템들만 전력이 차단된다. 또한 이 예시에서, 타이머는 다시 10초로 재설정되는데, 이는 만약 사용자가 디바이스를 다시 닫으면(closes), 상기 클로징(closing) 후 10초 동안 동면 또는 다른 비활성 모드로 진입하지 않도록 하기 위함이다. 특정 실시예에서는 또한, 디바이스는 하나의 입력에는 응답하여 즉시 슬립(sleep) 상태로 진입할 수 있고 다른 입력에는 그렇지 않을 수 있다. 예를 들어, 컨트롤 키 조합의 입력(entry)은 디바이스를 즉시 슬립 상태로 진입하도록 할 수 있고, 그러한 조합의 입력은 사용자에 의해 더 의도된 것으로 추정될 수 있다. 반면에, 클램쉘 디바이스의 클로징(closing of a clamshell device)은 미리 정해진 지연(delay)을 구현할 수 있고, 이는 그와 같은 액션은 일시적인 액션(예를 들어, 사용자가 디바이스를 한 장소에서 다음 장소로 이동시킴)으로 의도될 개연성이 더 있기 때문이다.

[0018] 디바이스의 전력 차단 프로세스를 시작하기 위한 특정 지연 시간(delay time)은 디바이스의 사용자에 의해 설정될 수 있다. 예를 들어, 만약 사용자가 배터리 절약을 최대화 하기를 원치 않고, 컴퓨터를 빈번하게 닫고 즉시 또는 곧 컴퓨터 상에서 다시 작업을 시작하고 싶다고 결정한다면, 사용자는 슬립 또는 동면 모드로 진입하기 위해서 디바이스가 그 다양한 시스템들의 전력 차단을 시작하기 전에 상대적으로 긴 타이머가 카운트다운 하는 기간을 설정할 수 있다.

[0019] 여기에서 디바이스는, 컴퓨터의 힌지(hinge) 근처에 스위치를 포함하며 클램쉘이 닫힐 때 스위치가 눌려지고 컴퓨터는 전력 차단 모드로 진입할 것을 결정하도록 컴퓨터의 힌지(hinge) 근처에 스위치를 포함하는 클램쉘 배열(arrangement)을 갖는 것으로 도시되었지만, 다른 실시예들도 또한 이용될 수 있다. 예를 들어, 평면 터치스크린 태블릿(flat touchscreen tablet) 또는 슬레이트 컴퓨터(slate computer)는 사용자가 디바이스의 전면(front surface), 후면, 또는 주변 모서리 상의 파워 버튼을 눌러서 그것의 전력을 차단하는 것을 허용할 수 있다. 그와 같은 액션은 사용자에게 디바이스가 완전히 전원 차단되는 인상(impression)을 주기 위하여 디바이스 상의 디스플레이를 즉시 끄도록(to turn off) 할 수 있다. 그러나, 마이크로프로세서, 메모리 컨트롤러, GPU, 및 다른 서브 시스템들과 같은 디바이스 상의 다른 서브 시스템들은 전술한 바와 같이, 타임아웃 딜레이

(timeout delay)의 기간 동안 계속 유지될 수 있다.

[0020] 시스템의 전력을 차단하는 것 외에도, 타이머는 도시된 것과 같은 컴퓨팅 디바이스 상에서 리셋되는 보안 장치의 시작(onset)을 지연시키기 위해 사용될 수 있다. 예를 들어, 특정 컴퓨팅 디바이스들은 슬립 모드(sleep mode) 또는 다른 비활성 모드에 있을 때, 그 디바이스를 다시 완전히 활성화된 모드로 되돌리기 위해서 사용자에 의해 비밀번호 또는 다른 보안 메커니즘이 입력되도록 준비될 수 있다. 만약 사용자가 디바이스를 닫거나 비활성화하고 나서 그 디바이스를 다시 사용할 필요가 있다는 것이 급히 기억나면, 여기서 논의된 타이머 없는 경우, 사용자는 비밀번호를 재입력(reenter)하는 것이 강제될 수 있다. 여기서 논의된 타이머 도구와 같은 타이머가 있는 경우, 사용자는 디바이스를 비활성화하는 어떤 행동을 한 후 바로 버튼을 누르거나 디바이스를 열 수 있고, 디바이스를 잠금 해제(unlock) 하기 위한 패스워드 또는 다른 인증 정보(credentialing information)를 재입력 할 필요 없이 디바이스에 대한 홈페이지 또는 데스크탑이 그들에게 거의 즉시 디스플레이 되도록 할 수 있다. 요약하면, 디바이스 상의 잠금 해제 상태에서 잠금 상태로의 이행은 결정된, 및 사용자 선택 가능한 시간 기간 동안 의도적으로 지연될 수 있다.

[0021] 이 방식에서, 여기서 기술된 시스템 및 방법은 컴퓨팅 디바이스에 대한 더욱 편리한 사용자 경험(user experience)을 제공할 수 있다. 디바이스는 그 전력 차단(powering down)을 사용자가 전력을 차단하는 명령을 지시한 조금 후로 지연시킬 수 있고, 그러한 지연은 만약 사용자가 그 또는 그녀의 마음을 급히 바꾼다면 디바이스가 빠르게 전력 공급하는 것을 허용하기 위해 이용될 수 있다. 동시에, 상기 전력 차단 지연(power down delay)은, 디바이스에 대해 추가 배터리 전력이 소모되지 않도록, 상대적으로 짧을 수 있다.

[0022] 도 2a는 콘텍스트 객체를 사용하는 운영체제의 개념 다이어그램이다. 일반적으로, 콘텍스트 객체는 운영체제에 그 객체가 이미 존재하였던 콘텍스트에 대한 정보를, 그것과 함께 또는 그것을 위해 저장하는 운영체제 객체인데, 최초 객체가 그 운영체제 내에서 생성되고 또는 활성화 되었을 때 운영체제에 존재하고 활성화되었던 다른 객체들을 기술하는 것을 포함한다. 예를 들어, 이하에서 기술되는 특정 실시예에서는, 어떤 객체들(예를 들어, 애플리케이션, 프로그램, 파일)이 동시에 사용되고 있는지 기록하기 위하여 다양한 포인트에서 제시간에 “스냅샷”을 찍을 수 있다. 이 “스냅샷”들로부터 사용자가 그 또는 그녀의 컴퓨터 상에서 특정 객체가 동시에 열리도록 하는 것을 선호하는 인터페이스가 구성될 수 있다.

[0023] 도면에서, 컴퓨터(202)는 기본적인 클램쉘 랩탑 컴퓨터의 형태로 도시되어 있지만, 컴퓨터(202)는 스마트폰, 터치-스크린 태블릿, 또는 슬레이트 디바이스와 같은 다른 형태를 취할 수도 있다. 객체들에 대한 다수의 시각적 표현들은 컴퓨터(202)의 스크린 상에 나타나고, 컴퓨터(202) 상에 로드된 애플리케이션들의 아이콘 및 표현(representations)을 포함한다. 예를 들어, 아이콘(208)은, 컴퓨터(202)의 지속성 저장장치(persistent storage) 상에 저장되어 있거나 컴퓨터(202)로부터 접근 가능한 서버 시스템에 저장되어 있어서, 컴퓨터(202)에서 접근 가능한 특정 워드 프로세스 문서를 나타낸다. 예를 들어, 컴퓨터(202)는 서버 시스템에 대해 자신을 식별할 수 있는, 서버 시스템에 컴퓨터(202)의 사용자로 등록된 계정(account)을 지시하는 쿠키(cookie) 또는 다른 메커니즘을 저장할 수 있다. 이러한 메커니즘은 컴퓨터(202)에 의해서, 아이콘(208)의 문서를 나타내는 데이터 등과 같은 정보를 서버 시스템으로부터 획득하기 위해 사용될 수 있다. 그러면 사용자는 아이콘(208)을 선택하여 문서를 불러올(call up) 수 있다.

[0024] 위젯(widget) 또는 가젯(gadget)(212)은 컴퓨터(202)의 디스플레이 상에 시계로서 나타나고, 컴퓨터(202) 상에 또한 디스플레이 될 수도 있는 객체의 타입을 지시한다. 위젯 또는 가젯(212)은 다양한 친숙한 형태를 취할 수 있고, 디바이스(202) 상에 로드된 운영체제 상에서 이용 가능한 기능을 제공하기 위해 애플리케이션을 제작한(draft) 서드파티에서의 코드로 제공될 수 있다. 뉴스 피드(214)는, 컴퓨터(202)의 사용자에게 최근 이벤트 업데이트를 보여주는 뉴스 제공 사이트(news aggregator)의 형태로, 디바이스(202) 상에서 실행중인 활성 애플리케이션을 나타낸다.

[0025] 아이콘(204)은 컴퓨터(202)에 의해 수행되는 지속 검색(persistent search)의 형태의 객체를 나타낸다. 지속 검색은 컴퓨터(202)와 같은 디바이스에 의해 자동적으로 반복되는 검색이다. 예를 들어, 유럽으로 휴가를 계획 중인 사용자가, 만약 항공편이 특정 가격에서 이용가능해지면 즉시 고지받기 위해서, 유럽향 항공편의 지속성 검색을 설정할 수 있다.

[0026] 박스들(206, 210)은 객체들(208 및 204) 중 하나와 연관되어 저장될 수 있는 콘텍스트 데이터(contextual data)를 나타낸다. 예를 들어, 박스(210)는 하나의 문서 또는 문서(208)와 같은 워드 프로세스 문서에 대한 콘텍스트 정보를 나타낸다. 다양한 필드들이 상기 객체와 함께 저장될 수 있는 콘텍스트 정보의 타입을 나타내기 위해 박스(210)에 도시된다. 예를 들어, 상기 객체는 객체의 타입을 설명하는 이름, 및 컴퓨터(202)에 저장된

모든 다른 객체들에 대하여 상기 객체를 고유하게 식별하는 식별번호를 포함한다.

[0027] 박스(210)는 또한 객체가 최초 생성된 시간과 객체(208)가 생성되었을 때 컴퓨터(202)상에 역시 오픈되었던 객체들의 리스트를 보여준다. 부가하여, 박스(210)는 객체(208)가 마지막으로 사용된 시간에 오픈되었던 다른 객체들의 리스트를 포함한다. 이 예시에서, 사용자는 워드 프로세스 객체(208)가 생성되고 마지막으로 사용된 시간에 웹사이트 www.test.com을 검토하고 있었다. 또한, 워드 프로세스 문서가 생성되었을 때, 지속성 검색 객체(204)가 컴퓨터(202) 상에서 활성화 상태였다. 이와 같은 동시발생(concurrence)은 상기 문서가 지속성 검색에 의해 생성된 정보를 담기 위해 사용자에 의해 생성된 것을 지시할 수도 있다. 이 점과 관련하여 추가적인 강력한 추론이 형성될 수 있는데, 예를 들어, 만약 사용자가 상기 지속성 검색 결과들로부터 클립보드(clipboard)로 정보를 카피했다면, 상기 문서를 생성하고, 그 검색 결과들 또는 다른 카피된 데이터를 상기 문서로 붙여 넣는 것이다. 그러한 정보는 또한 박스(210)에 도시된 바와 같이, 객체(208)와 연관되어 저장될 수 있다. (예를 들어, 언제 그 문서가 마지막으로 저장되었는지, 문서 내의 무슨 정보가 그때 오픈되었던 다른 애플리케이션에 의해 공유되는지, 그렇게 하여 문서와 객체 사이에서 카피된 콘텐트를 지시하는 것을 판단함에 의해서)

[0028] 박스(210)는 또한 문서 객체(208)가 상호작용하는 객체들을 나타낸다. 이 예시에서, 문서(208)는 웹 프린터 애플리케이션과 상호작용 하였고, 이는 문서(208)가 그 프린터 상에서 출력되었다는 것을 나타낼 수 있다. 객체들 사이의 특정 직접적인 상호작용은, 그 객체와 다른 객체들 사이의 특별히 강한 연결(connections)의 표시(indication)를 제공할 수 있기 때문에, 저장될 수 있다. 그러한 연결은 특정 상황에서 객체의 의도된 사용자를 식별하기 위해 사용될 수 있다.

[0029] 박스(206)는 객체(204)와 연관되어 저장된 콘텍스트 정보를 나타낸다. 다시, 상기 객체의 이름은, 그 객체가 생성된 시간과 함께 고유 식별 번호를 포함한다. 박스(206)에 나타난 바와 같이, 지속성 검색 객체(204)가 생성되었을 때 4개의 다른 객체들이 활성화되었다. 이 다른 객체들은 워드 프로세스 문서, 스프레드시트(spreadsheet) 문서, 및 상기 지속성 검색 객체가 생성되었을 때 컴퓨터(202) 상에 보여지던 두 개의 다른 웹 페이지를 포함한다. 또한, 박스(206)는 지속성 검색 객체가 컴퓨터(202) 상의 지도 애플리케이션 및 연락처 애플리케이션과 상호작용 하였다는 것을 나타낸다. 또한, 접속 기록(access log)은 상기 객체가 사용자에 의해 엑세스된 시간을 지시하고 또한 사용자가 그 객체에 대하여 무엇을 하였는지를 지시하는 정보를 포함할 수도 있다.

[0030] 여기에 도시된 정보를 이용하여, 다양한 서비스들이 컴퓨터(202)의 사용자에게 제공될 수 있다. 예를 들어, 객체(208)가 컴퓨터(202)에서 다음에 실행될 때, 컴퓨터(202)는 특정 문서 또한 워드 작업을 위해 열려 있을 때 디바이스(202)의 사용자가 빈번하게 브라우저(browser)에서 www.test.com 웹 페이지를 열었다는 것을 판단하기 위해 정보 박스(210)(실제로는 컴퓨터(202) 또는 다른 디바이스에 저장되어 있는 데이터, 박스(210)는 이 도면의 거와 같은 데이터를 나타냄)를 참조할 수 있다. 이와 같은 판단은, 문서가 열렸을 때 자동으로 브라우저 상에서 활성화된 웹 페이지와 함께 브라우저를 실행하기 위해 적절한 상황에서 사용될 수 있다. 이러한 방법으로, 사용자는 다수의 추론에 의해-관련된(inferentially-related) 애플리케이션들이, 오직 그 애플리케이션들 중 하나에 대한 하나의 아이콘을 선택하여, 활성화되도록 할 수 있다. 일부 실시예에서, 컴퓨터(202)는 저장된 메타데이터(meta data)에 응답하여 하나의 객체를 인스턴스화(instantiating) 하기 전에 사용자의 허가를 요청할 수 있다. 만약 사용자가 허가(permission)을 제공하면, 상기 객체는 인스턴스화된다. 만약, 사용자가 허가를 거부하면, 프로그램은 인스턴스화되지 않는다. 컴퓨터(202)는 그러한 허가의 제공 또는 거부에 관한 사용자의 히스토리(history)를 메타 데이터에 더 저장할 수 있다.

[0031] 도 2b는 콘텍스트 객체를 사용하는 운영 체제의 개념 다이어그램이다. 이 도면에서, 아이콘(228)은 컴퓨터(202)에서 접근 가능한, 예를 들어 컴퓨터(202) 상의 지속성 저장장치, 또는 컴퓨터(202)에서 접근 가능한 서버 시스템에 저장된 특정 이미지를 나타낸다. 예를 들어, 컴퓨터(202)는 서버 시스템에 대해 자신을 식별할 수 있는, 서버 시스템에 컴퓨터(202)의 사용자로 등록된 계정(account)을 지시하는 쿠키(cookie) 또는 다른 메커니즘을 저장할 수 있다. 이러한 메커니즘은 컴퓨터(202)에 의해서, 아이콘(208)의 문서를 나타내는 데이터 등과 같은 정보를 서버 시스템으로부터 획득하기 위해 사용될 수 있다. 그러면 사용자는 아이콘(228)을 선택하여 문서를 불러올(call up) 수 있다.

[0032] 아이콘(224)은 채팅 윈도우(chat window) 형태의 객체를 나타낸다. 채팅 윈도우는 사용자가 컴퓨팅 디바이스, 예를 들어 컴퓨터 또는 스마트폰의 또 다른 사용자와 통신하는 것을 가능하게 한다. 특정 상황에서, 채팅 프로그램(chat programs)은 둘 이상의 사람들이 타이핑 메시지를 주고받으면서 통신하는 것을 가능하게 한다. 다른 상황에서, 채팅 프로그램은 채팅을 위한 오디오 또는 비디오의 송신 및 수신을 용이하게 한다.

[0033] 박스들(226, 230)은 객체들(228 및 224) 중 하나와 연관되어 저장될 수 있는 콘텍스트 데이터를 나타낸다. 예를

들어, 박스(230)는 하나의 문서 또는 이미지 파일(228)과 같은 워드 프로세스 문서에 대한 콘텍스트 정보를 나타낸다. 다양한 필드들이 상기 객체와 함께 저장될 수 있는 콘텍스트 정보의 타입을 나타내기 위해 박스(230)에 도시된다. 예를 들어, 상기 객체는 객체의 타입을 설명하는 이름, 및 컴퓨터(202)에 저장된 모든 다른 객체들에 대하여 상기 객체를 고유하게 식별하는 식별번호를 포함한다.

[0034] 박스(230)는 또한 객체가 최초 생성된 시간과 객체(228)가 생성되었을 때 컴퓨터(202)상에 역시 오픈되었던 객체들의 리스트를 보여준다. 부가하여, 박스(230)는 객체(228)가 마지막으로 사용된 시간에 오픈되었던 다른 객체들의 리스트 및 상기 객체가 생성되고 마지막으로 사용되었을 때 있었던 사람들 리스트를 포함한다. 다른 방법들이 상기 객체가 생성되고 사용되었을 때 있었던 당사자들(parties)을 판단하기 위해 사용될 수 있다. 예를 들어, 그 객체를 실행하는 프로그램은 존재하는 당사자들에 대한 사용자 입력을 요청할 수 있다. 또 다른 예시에서, 컴퓨터(202)는 저장된 데이터베이스에 기초하여 사용자의 음성을 감지(sensing)하거나 저장된 데이터베이스에 기초하여 특정 개인과 연관된 디바이스를 탐지(detecting)하여, 어떤 당사자가 존재하는지를 자동으로 판단할 수 있다.

[0035] 도 2b를 참조하면, 박스(230)에 나타난 바와 같이, 객체 생성 당시 사용자 X는 사용자 Y 및 사용자 Z와 채팅 중이었다. 사용자 X는 객체가 마지막으로 사용된 시간에 사용자 Z와 채팅 중이었다. 박스(230)는 객체 생성 시에 사용자 X가 존재했었고 객체가 마지막으로 사용된 시점에 사용자 X 및 사용자 Y가 모두 존재했었다는 것을 추가로 나타낸다. 존재했던 사용자들은, 심지어 오직 사용자들 중 한 명만 실제로 “드라이빙(driving)” 중 이었다고 하더라도, 마우스나 터치패드로 선택을 하고 키보드나 다른 입력 장치로 텍스트를 입력함으로써 컴퓨터(202)를 사용하는 것으로 식별될 수 있다. 이렇게 하여, 근처(즉, 컴퓨터(202)와 같은 방 또는 컴퓨터(202)에서 특정 거리-예를 들어 15피트- 이내)에 있는 개개인이 존재하는 당사자들을 자동으로 판단하는 컴퓨터(202)를 이용하여 포함될 수 있다. 다른 상황에서, 존재하는 사용자들은 컴퓨터(202)에 의해 유도된 때에 또는 자발적으로 컴퓨터(202)를 사용하는 자신들을 식별할 수 있다.

[0036] 이와 같은 동시 발생은 그 이미지 파일이 사용자 X에 의해 사용자 Y 및 사용자 Z로부터의 입력과 함께 생성되었다는 것을 나타낼 수 있다. 이 점과 관련하여 추가적인 강력한 추론이 형성될 수 있는데, 예를 들어, 만약 사용자가 상기 지속성 검색 결과들로부터 클립보드(clipboard)로 정보를 카피했다면, 상기 문서를 생성하고, 그 검색 결과들 또는 다른 카피된 데이터를 상기 문서로 붙여 넣는 것이다. 상기 객체가 사용자 X, 사용자 Y, 및 사용자 Z 사이의 공동 작업(collaboration)을 통해 생성되었다는 것을 암시하는 이와 같은 정보는 또한 박스(230)에 도시된 바와 같이, 객체(228)와 연관되어 저장될 수 있다.

[0037] 박스(226)는 객체(224)와 연관되어 저장된 콘텍스트 정보를 나타낸다. 상기 객체의 이름은 객체가 생성된 시간을 포함한다. 이 예시에서, 채팅 윈도우는 채팅을 하고 있는 사람, 객체를 고유하게 식별하는 것을 나타낼 수 있다. 상기 객체는 컴퓨터 상의 다른 객체들에 대해서 상기 객체를 고유하게 식별하는 식별 번호를 포함할 수 있다. 박스(226)에 나타난 바와 같이, 채팅 윈도우(224)가 생성되었을 대 두 개의 다른 객체들이 활성화되었다. 이 다른 객체들은 이미지 파일 및 상기 채팅 윈도우 객체가 생성되었을 때 컴퓨터(202) 상에 보여지고 있었던 또 다른 채팅 윈도우를 포함한다.

[0038] 여기에 도시된 정보를 이용하여, 다양한 서비스들이 컴퓨터(202)의 사용자에게 제공될 수 있다. 예를 들어, 객체(228)가 컴퓨터(202)에서 다음에 실행될 때, 컴퓨터(202)는 이 특정 이미지 파일에 대해 작업하고 있을 때 디바이스(202)의 사용자 X가 사용자 Y 및 사용자 Z와 빈번하고 공동 작업한다는 것을 판단하기 위해 정보 박스(230)(실제로는 컴퓨터(202) 또는 다른 디바이스에 저장되어 있는 데이터, 박스(230)는 이 도면의 거와 같은 데이터를 나타냄)를 참조할 수 있다. 이와 같은 판단은, 이미지 파일이 열렸을 때 사용자 Y 및/또는 사용자 Z에 대한 채팅 윈도우들을 실행하기 위해 적절한 상황에서 사용될 수 있다. 이러한 방법으로, 사용자 X는 다수의 추론에 의해-관련된(inferentially-related) 애플리케이션들이, 오직 그 애플리케이션들 중 하나에 대한 하나의 아이콘을 선택하여, 활성화되도록 할 수 있다.

[0039] 유사하게, 객체(224)가 다음에 실행되고, 사용자 X와 사용자 Z 사이의 채팅이 시작되면, 박스 226의 정보를 분석하여 사용자 Y에 대한 추가 채팅 윈도우 및/또는 이미지 파일 객체(228)가 실행될 수 있다.

[0040] 또 다른 실시예에서, 컴퓨터(202)는 상기 컴퓨터를 사용하는 특정 사용자들을 식별할 수 있다. (개개인들은, 예를 들어, 사용자가 그 또는 그녀의 신분(identity)을 컴퓨터(202)에 입력하도록 유도되거나 컴퓨터(202)가 근처 또는 컴퓨터(202)로부터 특정 거리 이내에 있는 하나 이상의 사용자들을 감지할 때, “상기 컴퓨터를 사용하는 것(using the computer)”으로서 식별될 수 있다.) 예를 들어, 도 2b와 관련된 시나리오에서, 컴퓨터는 사용자 X와 함께 있는 사용자 Y 및/또는 사용자 Z의 존재를 감지할 수 있다. 박스(230)는 일반적으로 객체(228)를 사용

자 X, 사용자 Y, 및 사용자 Z에 의해 공동 작업되는 것으로 식별하기 때문에, 이 당사자들 중 하나 이상의 존재는, 컴퓨터(202)에 의해 탐지되었을 때, 컴퓨터(202)로 하여금 객체(228)를 자동으로 로드하도록 할 수 있다. 일부 실시예에서, 운영체제 객체가 클로즈 되었을 때 객체(228)의 콘텍스트 메타 데이터에 정의된 적어도 두 사람의 존재가 있지 않으면, 객체(228)는 컴퓨터(202)에 의해 자동으로 로드되지 않을 것이다. 상기 객체의 콘텍스트 메타 데이터에 정의된 적어도 하나-또는 일부의 경우, 적어도 둘-의 사람의 표시(indication)를 수신하는 것에 응답하여, 컴퓨터(202)는 운영 체제 객체(228)를 인스턴스화 한다.

[0041] 다른 컴퓨터 프로그램 및 애플리케이션들과 관련된 다양한 다른 실시예들이 이용될 수 있다. 예를 들어, 사용자는 워드 프로세스 프로그램을 사용하는 동안 음악을 듣는 것을 선호할 수 있다. 이와 같은 경우, 컴퓨터는 이러한 경향(trend)을 (상기 박스들에 대하여 기록된 메타(meta)를 이용하여) 알아챌 수 있고 사용자가 워드 프로세스 프로그램을 열면 언제라도 음악-재생 프로그램을 자동으로 예시할 수 있다. 또 다른 사용자는 워드 프로세스 프로그램에서 작업 중일 때 특정 앨범 또는 아티스트를 듣는 것을 선호할 수 있다. 이러한 경우, 컴퓨터는 그녀의 프로그램 과거 사용(또는 특정 워드 프로세스 문서들에 대한 과거 접근)으로부터 사용자의 선호도를 인식하고, 자동으로 음악-재생 프로그램을 예시하고 자동으로 사용자가 선호하는 아티스트, 앨범, 또는 노래를 플레이하는 것을 시작할 수 있다.

[0042] 다른 경우에, 컴퓨터는 다른 프로그램들이나 객체들이 인스턴스화 되었을 때 사용자가 특정 프로그램 또는 객체들을 닫거나(close) 꺼버리는(shut down) 경향을 가진다는 것을 메타 데이터에 저장할 수 있다. 예를 들어, 사용자는 채팅 윈도우가 인스턴스화 될 때마다 미디어 플레이어를 닫는 경향을 가질 수 있다. 사용자는 이 시나리오에서 미디어 플레이어에서 플레이 되는 미디어가 채팅 애플리케이션을 이용한 대화를 유지하기 위해 시도하는 동안 산만하게 한다는 것을 발견할 수 있다. 이와 같은 경우, 사용자는 채팅 윈도우에서 대화를 시작한 직후에 미디어 플레이어 프로그램을 꺼버린다. 컴퓨터는 이 경향을 (상기의 박스들에 대하여 기록된 메타를 이용하여) 알아챌 수 있고 채팅 윈도우가 인스턴스화 될 때마다 자동으로 미디어 플레이어(및 가능한 다른 유사한 미디어 플레이어)를 꺼버릴 수 있다. 일부 경우에, 컴퓨터는 어떤 채팅 윈도우가 인스턴스화되더라도 상기 미디어 플레이어를 꺼버릴 수 있다. 다른 예시들에서, 컴퓨터는 만약 인스턴스화된 채팅 윈도우가 특정 개인들과의 채팅을 포함하면 오직 상기 미디어 플레이어를 꺼버릴 수 있다. 이와 같은 판단은, 일반적으로 채팅 애플리케이션에 대하여 저장된 메타 데이터 또는 상기 채팅 애플리케이션을 이용하는 특정 사람들과의 상호작용에 대하여 저장된 메타 데이터에 기초할 수 있다. 사용자는 또한 그 또는 그녀의 개인 설정(settings) 또는 선호도(preferences)를 제공할 수도 있다.

[0043] 도 3은 컴퓨팅 디바이스(305) 상에서 메모리 제어(memory control)를 유지하기 위한 시스템(300)의 개념 다이어그램이다. 예시적인 시스템(300)은 디바이스(305) 상에서 소프트웨어 애플리케이션(310)의 실행을 제어하는 운영체제(330)를 포함할 수 있다. 일 실시예에서, 디바이스(305)는 동시에 여러 소프트웨어 애플리케이션을 실행하는 것이 가능한 운영체제(330)를 탑재한 셀룰러 폰(cellular telephone)일 수 있다. 대안적인 실시예에서, 컴퓨팅 디바이스(305)는 랩탑, PC(personal computer), PDA(personal digital assistant), 또는 다른 적절한 컴퓨팅 디바이스 일 수 있다.

[0044] 디바이스(305)가 텔레폰(telephone)인 일 실시예에서, 텔레폰이 켜진 후, 텔레폰의 운영체제(330)가 지속성 메모리(320)로부터 로드되고 미리 결정된 아이콘들의 디스플레이를 포함하는 그래픽 사용자 인터페이스가 사용자에게 제시될 수 있다. 각 아이콘은 애플리케이션 또는 사용자에게 이용 가능한 애플리케이션에 대한 프록시(proxy)일 수 있다; 선택되었을 때 선택된 아이콘은, 필요하다면, 메모리 내의 연관 애플리케이션의 파라미터들과 파일 위치를, 애플리케이션(310)을 차례로 실행할 수 있는 운영체제(330)로 보낼 수 있다. 각각의 실행된 애플리케이션은 텔레폰의 지속성 메모리(320)의 세그먼트(segment)를 사용한다; 애플리케이션은 계속해서 실행되기 때문에, 그 메모리 요구량은 증가할 수 있다. 사용자가 더 많은 애플리케이션들(310)을 사용하거나 실행중인 애플리케이션들(310)이 추가적인 메모리를 소비한다면, 텔레폰의 메모리(320)는 결국 애플리케이션들(310)의 메모리 요구를 수용하기에 부족해질 수 있다.

[0045] 예시적인 실시예에서, 메모리 관리 시스템(300)은 지속성 메모리(320)가 고갈되었을 때 하나 이상의 애플리케이션(310)을 종료하고 사용자가 그 애플리케이션으로 돌아올 때 그 종료된 애플리케이션을 회복(reviving)시키는 것에 의해 메모리 부족에 대응할 수 있다. 특정 실시예에서, 종료된 애플리케이션 윈도우는 완전히 또는 부분적으로 다른 애플리케이션 윈도우에 의해 보기 어려울 수 있기 때문에, 사용자는 애플리케이션이 종료되었다는 것을 인식하지 못할 수 있다. 사용자가 그 애플리케이션으로 다시 전환(switch back)할 것을 선택하면, 그 애플리케이션은 다시 실행(re-launched)될 수 있고 사용자는 그 애플리케이션을 디스플레이 함에 있어서 느린 반응이

라는 것 외에 그 애플리케이션이 일시적으로 종료되었다는 것을 모를 수 있다.

[0046] 실제적인 실시예에서, 운영체제(330)는 그래픽 사용자 인터페이스를 통한 사용자의 상호작용에 따라 애플리케이션들(310)을 랭크할 수 있고 각 애플리케이션(310)은 운영체제로부터의 신호에 응답하여 현재 상태에 관한 정보를 생성하고 저장할 수 있다. 이 애플리케이션들은 그러한 상태 정보를 스스로 저장하거나 그 정보를 지속성 저장장치(예를 들어, 플래시 메모리)에 차례로 저장할 수 있는 운영체제(330)로 제공할 수 있다.

[0047] 만약 메모리(320)가 고갈되면, 운영체제(330)는 하나 이상의 랭크된 애플리케이션(310)을 종료할 수 있고, 나중에 사용자 요청에 응답하여 그 종료된 애플리케이션을 재생성 할 수 있다. 예를 들어, 운영체제(330)가 로드되고, 사용자는 저장된 문서를 읽기 위해 문서 뷰잉(viewing) 애플리케이션을 선택할 수 있다. 계속해서, 문서 뷰어가 계속 실행되는 동안, 사용자는 웹 브라우저를 열고 인터넷 서핑을 시작할 수 있다. 웹 브라우징 세션 중에, 사용자는 이메일이 도착했다는 알림에 응답하여, 그 새로운 이메일을 검토하기 위해 텔레폰의 이메일 애플리케이션을 선택할 수 있다. 사용자가 그 이메일을 읽는 동안, 사용자는 이메일에서 언급된 이벤트에 대한 리마인더(reminder)를 생성하기 위해 캘린더(calendar) 애플리케이션을 실행하려고 할 수 있다.

[0048] 예시적인 실시예에서, 사용자가 새 애플리케이션을 열면, 운영체제(330)는 애플리케이션들(310)을 하나 이상의 동적 표준(dynamic criteria)에 따라 랭크할 수 있다. 여기서, 운영체제(330)는 실행중인 애플리케이션들(310)을 다음과 같은 방식: 이메일 애플리케이션, 웹 브라우저, 및 문서 뷰어 순으로 중요도에 대해 내림차순으로 랭크시킬 수 있다. 이와 같은 배치(ordering)는 다양한 방법으로 일어날 수 있다. 예를 들어, 상기 애플리케이션들은, 필수 비즈니스 애플리케이션, 엔터테인먼트 애플리케이션, 등과 같은 다양한 카테고리로 구분될 수 있다. 일부 실시예에서, 운영체제(330)는 특정 애플리케이션이 백그라운드에 있을 때 휴면(dormant)이 되는 것을 인식할 수 있고, 따라서 그 애플리케이션을 낮은 우선순위(low priority)로 분류할 수 있다. 그러나 또 다른 애플리케이션(예를 들어, 메시지 프로그램)은 네트워크를 통해 지속적으로 정보에 접근하고 있을 수 있고 따라서 높은 우선순위 애플리케이션으로 랭크될 수 있다. 일부 실시예에서, 애플리케이션들은 2 개의 카테고리: 보이는 애플리케이션과 보이지 않는 애플리케이션들로 나뉘어 질 수 있다. 보이지 않는 애플리케이션들(즉, 그 윈도우가 사용자에게 보이지 않는 애플리케이션)은 보이는 애플리케이션들에 비해 더 낮게 랭크된다. 대안적인 실시예에서, 애플리케이션의 개발자들은 애플리케이션들을 자체 분류하거나 사용자가 애플리케이션을 분류 또는 다른 관점에서 랭크시킬 수 있고, 그와 같은 분류 또는 랭킹은 운영체제(330)로 제공될 수 있다.

[0049] 사용자가 애플리케이션들(310)과 상호작용함에 따라, 각 애플리케이션은 그 애플리케이션의 현재 상태에 관한 정보를 생성하고 저장할 수 있다. 예를 들어, 애플리케이션이 운영체제(330)에 의해 언제라도 종료될 수 있는(could be killed) 상태로 들어가면(예를 들어, 그 애플리케이션이 더 이상 사용자에게 보이지 않으면), 운영체제(330)는 애플리케이션에게 그 현재 상태를 저장하라고 명령할 수 있다.

[0050] 실제적인 예시로 돌아가서, 텔레폰의 메모리(320)가 모든 4개의 애플리케이션을 한번에 실행시키기에는 부족할 수 있기 때문에, 운영체제(330)는 캘린더 애플리케이션을 열었을 때 그것이 메모리를 고갈시키기 때문에 가장 낮게 랭크된 애플리케이션(이 예시에서는, 문서 뷰어)을 종료할 것을 선택할 수 있다. 대안적인 실시예에서, 운영체제(330)는 미정의 메모리 부족(pending memory shortage)을 예측하고 메모리가 고갈되는 것(running out)을 방지하기 위해 하나 이상의 애플리케이션을 종료할 수 있다. 예를 들어, 만약 현재 이용 가능한 메모리의 양이 미리 결정된 임계 값 이하로 떨어지면, 운영체제는 현재 이용 가능한 메모리의 양을 임계 값 위로 가져오기 위해 낮게 랭크된 애플리케이션을 종료(kill)할 수 있다. 대안적인 실시예에서, 운영체제(330)는 애플리케이션(310)에 의한 도래할 메모리 요청을 현재 이용 가능한 메모리의 양과 비교하고; 만약 요청된 양이 현재 이용 가능한 양을 초과하면 운영체제는 하니 이상의 애플리케이션을 종료할 수 있다.

[0051] 이와 같은 상황에서, 운영체제(330)는 가장 낮게 랭크된 애플리케이션 또는 애플리케이션들을 식별하고 그들을 일시적으로 종료시킬 수 있다. 운영체제(330)는 또한 나중에, 예를 들어, 메모리가 해소되면(free up), 자동으로 재실행될(re-launched) 필요가 있는 애플리케이션들의 표시(indication)로서 플래그(flag)를 설정할 수 있다.

[0052] 종료할 애플리케이션의 선택은 다른 기술들에 의해 일어날 수 있다. 예를 들어, 운영체제(330)는 특정 애플리케이션에 의해 요구되는 메모리의 양을 판단하고 나서 메모리의 총 양을 오버헤드 메모리(overhead memory)의 특정 안전 영역(safety zone) 이상으로 해소할 수 있는 다른 작동중인 애플리케이션을 식별할 수 있다. 일 예시로서, 한 애플리케이션이 3000K의 추가 메모리를 요구하고 3개의 다른 애플리케이션들이 각각 2000K, 3000K, 및 35000K의 메모리를 각각 해소할 수 있다. 운영체제(330)는 메모리에 대한 최소 "데미지(damage)", 또는 가용 메모리의 최선의 사용(best use)을 판단할 수 있고, 처음의 두 프로그램을 종료시킴으로써 달성할 수 있는데 이는

그들이 가장 필요한 메모리의 양에 가장 근접해 있기 때문이다. 대안적으로, 운영체제는 가능한 적은 애플리케이션을 종료시키는 것을 선호하도록 프로그램 될 수 있다. 이와 같은 상황에서는, 세 번째 애플리케이션이 이 예시에서 종료될 것이다.

[0053] 사용자가 캘린더 애플리케이션의 사용을 마쳤을 때, 사용자는 문서 뷰잉 애플리케이션으로 돌아가는 것을 선택할 수 있다. 운영체제(330)가 일시적으로 종료되었던 문서 뷰잉 애플리케이션으로 돌아가기 위한 사용자의 시도를 탐지하면, 운영체제(330)는 저장된 상태 정보를 이용하여 상기 애플리케이션을 재생성 할 수 있다.

[0054] 그렇게 함으로써, 운영체제(330)는 우선 문서 뷰잉 애플리케이션에 접근하기 위한 커맨드(command)를 감지하고, 저장된 플래그로부터 그 애플리케이션이 활성화 상태였으나 일시적으로 종료되었다는 것에 주목할 수 있다. 운영체제(330)는 이제 상기 애플리케이션을 실행하고, 상기 애플리케이션이 일시적으로 종료되었을 때의 형태로, 또는 실질적으로 그 때의 형태로 재생성 되도록, 저장된 상태 정보를 상기 애플리케이션으로 보낼 수 있다. 대안적으로, 상기 애플리케이션은 그 자신의 상태 정보를 운영체제와 협력하여 저장했을 수 있고, 스스로 접근하여 그 정보를 구현할 수 있다.

[0055] 도 4는 컴퓨터 프로세스 사이의 메시지 패싱(message passing)에 쓰레드 관련성(thread affinity)을 제공하는 시스템의 개념 다이어그램이다. 일반적으로, 상기 시스템은 컴퓨팅 디바이스 상에서 실행 중이고 프로세스들 사이에서 정보를 보내기 위해 서로 통신중인 프로세스 쌍(pair)을 도시한다. 예를 들어, 하나의 프로세스는 메시지와 함께 제2 프로세스가 상기 메시지에 대해서 어떻게 응답 또는 반응하게 되는지에 대한 정보를 제2 프로세스로 보낼 수 있고, 제2 프로세스는 제1 프로세스로 응답 또는 반응할 때 확인(confirmation)을 제1 프로세스로 제공할 수 있다. 보다 특별하게 도 4를 참조하면, 시스템(400)은 제1 프로세스(402) 및 제2 프로세스(404)를 포함한다. 두 프로세스들(402, 404)은 모두 동시에 또는 실질적으로 동시에 하나의 단일 컴퓨팅 디바이스 상에서 타임-와이즈 오버래핑 방식으로(in a time-wise overlapping manner) 실행 중일 수 있다.

[0056] 제1 프로세스(402)는 예를 들어, 하나 이상의 문서(412)를 디스플레이 하는 문서 관리 프로그램일 수 있다. 문서 관리 프로그램은, 예를 들어, 워드 프로세스 애플리케이션 또는 워드 프로세스 애플리케이션을 웹 브라우저에서 구현하는 웹 앱(Web app)일 수 있다. 특정 경우에, 웹 브라우저는 운영체제의 일부일 수 있고, 여기에서 웹 브라우저만이 유일한 운영체제 기본(native) 애플리케이션이고, 모든 다른 애플리케이션들은 웹 브라우저 내에서 웹 앱으로서 동작한다.

[0057] 제2 프로세스(404)는 지도 애플리케이션(mapping application)의 실행을 나타낸다. 예를 들어, 시스템에서 실행중인 다른 웹 앱은 서버 기반 지도 서비스에 접근할 수 있고 흔히 알려진 방식으로 맵핑된 지리의 그래픽 타일(graphical tile)을 이용하여 지리적 위치의 디스플레이에 관한 정보를 제공할 수 있다. 예를 들어, 상기 프로세스(402, 404)를 실행 중인 디바이스의 사용자 위치는 핀(pin) 또는 다른 아이콘을 이용하여 지도상에 표시될 수 있다. 분리된 프로세스(402, 404)는 또한 하나의 디바이스 상에서 단일 웹 브라우저의 분리된 탭(separate tabs)으로 나타내 질 수 있고, 여기서 상기 프로세스들은, 일반적으로 다른 도메인 사이의 통신이 하나의 웹 브라우저에서 발생하는 것을 방지하는 구현을 포함하여, 서로 샌드박스(sandboxed) 된다.

[0058] 상기 프로세스(402, 404) 사이를 지나가는 화살표 세트는 상기 프로세스 중 하나로부터 다른 하나로, 및 그 역으로 통신될 수 있는 메시지 및 정보를 나타낸다. 예를 들어, 메시지(406)는 최초에 프로세스(402)로부터 프로세스(404)로 보내질 수 있다. 예를 들어, 상기 메시지는 프로세스(404)에 대한 디스플레이가 변경될 수 있도록 하는 정보를 지시할 수 있다. 이 예시에서, 예를 들어, 메시지(406)는 지도 애플리케이션(414)에 디스플레이 된 지도의 다른 영역이 보여지도록 하기 위해 사용될 수 있는 위도와 경도 또는 주소 설명을 포함할 수 있다.

[0059] 부가하여, 프로세스(402)는 프로세스(404)로 메시지(406)와 관련된 커맨드 또는 커맨드들이 프로세스(404)에 의해 수행되어야 하는 방식에 대한 정보를 보낼 수 있다. 이 예시에서, 상기 정보는 우선순위 커맨드(priority command)(408)의 형태이다. 우선순위 커맨드(408)는 메시지(406)에 관련된 그것의 실행에 주어지는 우선순위에 관하여 통지할 수 있다. 예를 들어, 만약 프로세스(402)가 시간-민감(time-sensitive) 형태의 프로세스가 아니라면, 우선순위 커맨드(408)는 메시지(406)에 대한 응답은 프로세스(404)에 의해 높은 우선순위로 처리되지는 않을 것을 나타낼 수 있다.

[0060] 메시지(406)와 우선순위 커맨드(408)는 특정 상황에 따라, 별개로 또는 함께, 및 직접 또는 간접적으로 프로세스(404)로 보내질 수 있다. 예를 들어, 메시지(406)와 우선순위 커맨드(408)가 더 큰 단일 메시지 내에서 함께 보내질 때, 프로세스(404)는 상기 메시지의 메시지 부분, 및 메시지 내에 삽입된(embedded) 우선순위 커맨드를

식별하기 위해 상기 더 큰 메시지를 파싱할 수 있다. 다른 예시들에서, 부가 정보가 메시지(406)에 포함될 수 있고, 프로세스(402) 및 프로세스(404)가 동작 및 상호운용 되도록 프로그램 된 적절한 방식으로 식별 및 처리될 수 있다. 프로세스(402)와 프로세스(404)의 상호운용(interoperability)은 상기 두 프로세스들이 API(애플리케이션 프로그래밍 인터페이스) 또는 프로세스들 사이의 통신을 포맷팅(formatting)하는 다른 유사한 표준을 결합(adhering)함으로써 유지될 수 있다.

[0061] 특정 실시예에서, 프로세스(404)는 메시지를 프로세서(402)로 돌려 보낼 수 있다. 그러한 일 메시지는 프로세스(402)에게 프로세스(404)가 메시지(406)에 완전하게 반응했다는 것을 지시하는 확인(confirmation)(410)이다. 특정 실시예에서, 확인(410)은 단순히 메시지(406)의 수신에 응답하여 프로세스(402)로 정보를 다시 제공하는 프로세스(404)에 의해 일어날 수 있다. 이 특정 예시에서, 그러한 정보는 애플리케이션(404)에 의해 결정된 맵핑된 영역의 이미지를 포함할 수 있고, 이는 자동으로 프로세스(402)에 의해 관리되고 있는 문서(412)로 통합될 수 있다.

[0062] 이 방식에서, 프로세스들이 하나의 웹 브라우저 애플리케이션의 샌드박스된 환경에서 실행되는 개별 윈도우 또는 탭의 일부인 때를 포함하여, 인터-프로세스 커뮤니케이션(inter-process communication)이 하나의 프로세스가 부가 정보를 또 다른 프로세스로 통신하는 것을 허용함으로써 향상될 수 있다. 이러한 통신은 제1 프로세스가, 제2 프로세스가 코드의 실행에 부여한 우선순위를 포함하는, 특정 코드를 실행하는 또 다른 프로세스의 방식을 제어하도록 할 수 있다. 결과적으로, 프로세스들 사이의 더 밀접한 상호운용이, 상기 프로세스 중 하나가 다른 프로세스 중 하나를 불법으로 제거하거나 영향을 미치는 것을 방지하는 높은 수준의 보안을 여전히 유지하는 동안, 제공될 수 있다.

[0063] 도 5는 상태 비 보존형 환경에서(in a stateless environment) 상태 정보를 제공하는 시스템의 개념 다이어그램이다. 일반적으로, 시스템(500)은 서버 시스템(502)의 특정 사용자를 대신하여 서버 시스템(502)에 로그인 된 다양한 디바이스 상의 브라우저 애플리케이션들의 상태에 관한 최신 상태 정보를 저장하기 위해 서버 시스템(502)를 이용한다. 예를 들어, 브라우저가 컴퓨팅 디바이스(504)와 같은 디바이스 상에서 운영체제의 유일한 기본 애플리케이션(native application)인 경우에, 상기 상태 정보는 계속적으로 서버 시스템(502)으로 업로드 되고 업데이트될 수 있다. 이러한 방식에서, 만약 사용자가 컴퓨팅 디바이스(504)를 끄거나 그렇지 않으면 또 다른 컴퓨팅 디바이스를 가져와서 그 다른 컴퓨팅 디바이스에서 서버 시스템(502)으로 로그인 하면, 브라우저 애플리케이션에 대한 상태 정보는 그 다른 디바이스 상에서 실행중인 그 애플리케이션으로 복제될 수 있고, 여기서 그 애플리케이션은, 웹 브라우저 애플리케이션이 유일한 기본 애플리케이션이고, 모든 다른 애플리케이션들은 그 브라우저 애플리케이션에서 실행되는 웹 앱인 운영체제에서 또한 실행될 수 있다. 또한, 만약 사용자가 일 디바이스 상의 세션(session)을 닫고 나중에 동일 디바이스를 시작하면, 여기에 설명된 기술은 그 디바이스 자체 보다는 서버 시스템에 저장된 상태 데이터를 이용하여 상기 세션을 재-구축(re-establish)할 수 있다. 결과적으로, 상기 시스템은 그러한 정보를 디바이스 상에 저장하기 위한 초과 메모리를 요구하지 않고 동작할 수 있고, 상태 정보는 디바이스들 사이에서 더욱 쉽게 공유될 수 있다.

[0064] 이 예시에서 시스템(500)은 네트워크(506), 예를 들어 인터넷 및 관련 연결된 네트워크들을 통해 서버 시스템(502)와 통신하는 컴퓨팅 디바이스(504)를 포함한다. 서버 시스템(502)은 시스템(500)의 사용자들에게 다양한 웹 서비스들을 제공하는 시스템 내의 거대한 데이터 센터의 일 부분으로서 수용될 수 있다. 예를 들어, 서버 시스템(502)은 컴퓨팅 디바이스(504) 상의 디스플레이를 위한 문서들을 생성하는 HTML 코드를 제공할 수 있는 하나 이상의 웹 서버들을 포함할 수 있다.

[0065] 컴퓨팅 디바이스(504)는, 컴퓨팅 디바이스(504)의 상태가 나중에 그 디바이스(504) 또는 다른 디바이스 상에 복제될 수 있도록, 일반적으로 상태 비 보존형 애플리케이션(stateless application)일 수 있는 브라우저 애플리케이션에 대한 상태 정보가 저장되는 것을 허용하는 다수의 특정 구성요소들을 포함한다. 예를 들어, 사용자가 디바이스(504)를 슬립 모드로 두거나 디바이스(504)를 끄면, 디바이스(504)의 현재 상태 대부분은 서버 시스템(502)에 이미 저장되었거나(예를 들어, 사용자가 웹 브라우저에 대한 그 또는 그녀의 마지막 액션을 수행했을 때) 운영체제가 디바이스(504)가 완전히 슬립 모드로 진입하도록 하기 전에 서버 시스템(502)로 업로드 될 수 있다. 상태 정보는, 예를 들어, 만약 디바이스(504)가 디바이스(504)의 상태 정보에 변화가 있을 때마다 상태 정보의 변화에 대한 지시자(indicator)를 업로드 하도록 프로그램 되어 있다면, 서버 시스템(502)에 사전에 저장되었을 수 있다. 상태 정보는, 상기 디바이스의 상태를 재생성하기 위해 필요한 다른 관련 정보에 부가하여 디바이스 상에 현재 디스플레이 된 페이지들에 대한 DOMs(document object modes)를 포함할 수 있다.

[0066] 이제 디바이스(504)에서 구현될 수 있는 특정 구성요소들을 참조하면, 브라우저 애플리케이션(510)이 도시되고

이는 디바이스(504)에서 기본적으로(natively) 실행되는 유일한 애플리케이션일 수 있다. 별도로, HTML(508)이 브라우저 애플리케이션(510)에 의해 렌더링될 수 있도록 디바이스(504)에 저장될 수 있다. HTML(508)은 다양한 형태를 취할 수 있고 DOM 트리(DOM tree)로서 하나 이상의 예시에서 표현될 수 있다. 또한 디바이스(504)는 자바스크립트(JavaScript)(512) 및 자바스크립트 변수들(514)를 저장하고 구현할 수 있다. 예를 들어, HTML(508)이 렌더링되면, 상기 HTML은 포인터(pointers) 또는 디바이스(504)에서 실행될 다양한 자바스크립트 프로그램들에 대한 호출(calls)을 포함할 수 있다. 이 프로그램들은, 실행될 때, 그 프로그램의 운영에 필요한 변수들 또는 다른 정보를 수집하거나 생성할 수 있다.

[0067] 또한, 사용자 인터페이스(516)는 디바이스 상에 저장될 수 있고 디바이스(504)의 시각적 디스플레이 또는 스크린 상에 디스플레이 된 것의 현재 표현(representation)을 포함하여, 다양한 파라미터들을 나타낼 수 있다. 예를 들어, 사용자 인터페이스(516)는, 특정 시점에 브라우저의 무슨 탭이 오픈되었는지, 및/또는 특정 상태로서 플로팅 페인(floating pane)과 같이 운영체제에 의해 지원되는 다른 타입의 애플리케이션이 브라우저의 상단에 디스플레이 되었는지에 관한 정보, 및 디바이스(504) 상의 현재 디스플레이 상태를 기술하기 위해 사용될 수 있고 디바이스(504) 상에 현재 디스플레이 된 정보와 매칭되는 디스플레이를 재구성하기 위해 추가로 사용될 수 있는 다른 적절한 정보를 저장할 수 있다.

[0068] 웹 페이지 저장 애플리케이션(518) 역시 디바이스(504) 상에서 구현될 수 있고, 브라우저(510) 및 다른 브라우저들, 또는 디바이스(504) 상에서 실행중인 다른 프로그램들의 현재 상태를 추적할 수 있다. 예를 들어, 웹 페이지 저장 애플리케이션(518)은 브라우저(510)에 의해 표현되는 콘텐트 또는 다른 정보에 관하여 이루어진 변경을 판단하기 위해 디바이스(504) 상의 특정 구성요소에서의 또는 그로부터의 호출(calls)을 가로챌 수 있다. 웹 페이지 저장 애플리케이션(518)은 이제 브라우저(510)의 현재 상태에서 이루어진 변경을 나타내도록 통신(communication)으로 하여금 디바이스(504)에 의해 생성되고 서버 시스템(502)을 향하도록 할 수 있다. 예를 들어, 디바이스(504)는 브라우저 탭이 오픈되었다는 것을 지시하는 정보, 및 그 탭에 대한 URL 또한 제공할 수 있다.

[0069] 웹 페이지 저장 애플리케이션(518) 및 디바이스(504) 상의 다른 구성요소들은 또한 이미지(520)의 데이터 저장에 접근권한을 가질 수 있다. 특정 형태에서, 이미지(520)는 단순히 운영체제 수준의 이미지, 예를 들어 일반적인 아이콘 및 디바이스(504)를 실행하고 디바이스(504)의 상태를 재생성 하기 위해 필요한 다른 기본 정보에 대한 이미지이다. 상기 이미지들은 또한 웹 페이지 상의 이미지 또는 디바이스(504) 상의 웹 앱에 의해 사용된 이미지일 수 있고, 웹 페이지 저장 애플리케이션(518)에서 또는 그에 의한 나중 접근을 위해 상기 애플리케이션(518)을 대표하여 저장될 수 있다.

[0070] 여기에 일 예시에 기재된 시스템(500)을 이용하여, 사용자 디바이스의 현재 상태는 캡쳐될 수 있고 서버 시스템(502)으로 업로드 될 수 있다. 사용자는 홈 컴퓨터에서 네트워크 컴퓨터로 이동하는 것과 같이 이제 다른 컴퓨팅 디바이스로 이동할 수 있고, 사용자가 다른 디바이스를 부팅하거나 아마 그 자신을 식별하기 위해 인증정보(credentials)를 서버 시스템(502)으로 제공하면, 디바이스(504)의 전체 상태가 작업 컴퓨터 또는 다른 디바이스에 복제될 수 있다. 결과적으로, 사용자는 하나의 설정에서 다른 하나로 전환할 수 있거나 그들이 떠났던 장소로 더 편리하게 돌아올 수 있고 디바이스(504) 상에서 더욱 효율적으로 애플리케이션들을 사용할 수 있다. 예를 들어, 만약 사용자가 사용자를 대신하여 브라우저(510)에서 실행중인 애플리케이션 또는 웹 앱을 설정하면, 사용자는 이 웹 앱들이 나중에 또는 다른 디바이스에서 실행되는 것을 유지하고자 할 수 있다. 예를 들어, 사용자는 이메일 웹 앱, 문서 편집 앱, 지도 앱, 및 일반적인 웹 브라우징 윈도우가 항상 오픈되어 있는 것을 선호할 수 있고, 상태 복제 기술은 사용자가 하나의 기계에서 다른 하나로 이동할 때마다 사용자가 수동으로 그 상태를 재생성할 필요 없이 사용자로 하여금 그렇게 하는 것을 허용할 수 있다.

[0071] 도 6은 네트워크를 통해 컴퓨팅 디바이스에 대한 이미지(여기에서 이미지는 사진(photograph)과 같은 그래픽 이미지보다는 시스템 이미지이다)를 제공하는 시스템(600)의 개념 다이어그램이다. 일반적으로, 시스템(600)은 인터넷과 같은 네트워크(606)를 통해 호스트 서버 시스템(hosted server system)(602)과 통신하는 모바일 컴퓨팅 디바이스(604)와 같은 다양한 디바이스를 포함한다. 검색 엔진 서비스, 지도 서비스, 이메일 서비스, 문서 관리 서비스, 등과 같은 다양한 호스트 서비스(hosted services)를 제공하는 것에 부가하여, 호스트 서버 시스템(602)은 또한 디바이스(604) 상의 운영체제를 관리하기 위하여 디바이스(604) 상의 운영체제와 협력할 수 있다. 예를 들어, 디바이스(604) 상의 운영체제는 웹 브라우저의 형태의 단일의 기본 애플리케이션을 갖는 운영체제일 수 있고, 디바이스(604) 상에서 실행되는 다른 애플리케이션들은 상기 웹 브라우저 내의 웹 앱들처럼 행동할 수 있다. 이와 같은 접근은 디바이스에 저장되고 관리될 필요가 있는 기본 애플리케이션들의 수를 최소화 할 수 있다. 부가하여, 여기에 기술된 것처럼, 그와 같은 처리방식(arrangement)은 디바이스(604)가 호스트 컴퓨터 시스

템(602)에 의해 관리될 수 있는 방식을 단순화할 수 있다.

[0072] 이 예시에서, 디바이스(604)는 이미지(608)를 저장하는 것으로 도시된다. 상기 이미지는 무슨 구성요소가 지속적으로 디바이스(604) 상에 저장되는지 정의할 수 있다. 예를 들어, 이미지(608)는, 특정 프로그램들에서, 디바이스(604) 상의 펌웨어 뒤의(beyond the firmware) 기본적인 운영체제 구성요소들을 포함할 수 있고, 여기서 하나의 단일 프로그램은 웹 브라우저 형태의 단일 기본(native) 프로그램을 갖는 디바이스 상의 운영체제 이미지의 일부이다. 일반적으로, 이미지들은 다수의 컴퓨터들이, 그와 같은 컴퓨터들의 운용성(operability)을 관리하고 유지하기 위한 능력을 향상시키기 위해, 그들에 있어서 구성요소들의 공동 베이스라인(common baseline)을 갖는 것을 보장하기 위해 사용될 수 있다. 특히, 회사(company)는 그 직원들에게 특정 수의 소프트웨어 구성요소들은 사용 가능하고 다른 구성요소들은 사용 가능하지 않는 것을 원할 수 있고, 따라서 이미지를 정의하고, 그들이 최초로 배치될 때 그 이미지를 직원들의 기계에 설치할 수 있다.

[0073] 이 예시에서, 서버 시스템(602)은 상기 디바이스(604) 및 상기 시스템(602)으로부터 서비스를 수신하는 다른 디바이스들과, 디바이스 상에서 이미지들을 유지 또는 보수하기 위해 협력할 수 있다. 도면에 도시된 바와 같이, 다수의 특정 구성요소들이 그와 같은 서비스를 제공하는데 있어서 서버(602)에 의해 이용될 수 있다. 예를 들어, 이미지 인터페이스(610)가 디바이스(604)와 같은 원격 디바이스들과 상호작용 하기 위해 제공될 수 있다. 이미지 인터페이스(610)는, 예를 들어, 각각의 상기 디바이스들 상의 이미지가 여전히 정확한지를 확인하기 위해 디바이스들과 통신할 수 있다. 예를 들어, 어떤 애플리케이션도 디바이스에 추가되는 것이 허용되지 않는 경우에, 유일한 기본 애플리케이션은 브라우저이고 부가 애플리케이션들은 지속적으로 저장되지 않는 웹 애플리케이션과 같을 때, 해쉬(hash)가 디바이스(604)의 기본 구성요소(native components)로부터 구성될 수 있고, 그 해쉬는 저장되고 디바이스(604)가 부팅되는 매 시간마다 계산된 후속 해쉬(subsequent hash)와 비교될 수 있다. 만약 이 해쉬들이 일치하지 않으면, 이는 디바이스(604) 상의 운영체제에 대한 핵심 구성요소(core components)가 위태로워졌다(have been compromised)는 것을 지시할 수 있다. 이와 같은 상황에서, 디바이스(604)는 인터페이스(610)로 하여금 결국 서버 시스템(602)의 다른 구성요소들이 특정 동작을 수행하도록 하는 메시지를 이미지 인터페이스(610)로 보낼 수 있다.

[0074] 예를 들어, 이미지 재-구성기(image re-constructor)(616)는, 디바이스(604)에 대한 대체 이미지일 수 있는 이미지에 대한 특정 구성요소(components)를 식별하고, 수집하고(gather), 조합하도록(assemble) 프로그램 될 수 있다. 이미지 재구성기(616)는 디바이스(604)에 인스톨 되기로 되어 있는 이미지의 형태를 식별하기 위해 초기에 이미지 인덱스(618)를 참고할 수 있다. 예를 들어, 운용체제에 대한 특정 개정번호(revision number)가 디바이스(604)에 대해 제공될 수 있다. 대안적으로, 다른 제조사 또는 사용자 기반의 디바이스들은 각각 커스텀 이미지를 가질 수 있지만, 단일 제조사에 의해 제공되는 디바이스 라인은 공통의 이미지를 가질 수 있다. 그러므로, 이미지 인덱스(618)는 제조사와 디바이스 또는 그 디바이스를 운용하는 회사의 모델에 대한 식별자를 수신할 수 있고, 그와 같은 디바이스로부터의 요청에 응답하여 빌딩되어야 하는(to be built)이미지를 식별할 수 있다. (여기서 그 이미지는 시스템에 의해 저장된 공통의 교환 가능한 구성요소로부터 빌딩될 수 있다.)

[0075] 이 예시에서, 하나의 이미지는 이미지 조각들(image fragments)로부터 빌드 업(build up) 될 수 있다. 예를 들어, 특정 레벨의 운영체제는 상기 이미지의 일부일 수 있고, 모듈식 운영체제(modular operating system)에서 각 레벨은 이미지의 별개의 조각일 수 있다. 또한, 동일 레벨(common level)에서의 다른 기능들도 또한 별개의 이미지 조각으로서 고려되고 저장될 수 있다. 이미지 재구성기(616) 또는 이미지 업데이터(614)는 이미지 인덱스(618)로부터의 정보를 이용할 수 있고, 예를 들어, 이미지 식별자를 전체 이미지를 구성하는 다양한 조각들 또는 구성요소들에 맵핑할 수 있다.

[0076] 또한, 사용자 디바이스 지도(622)는 특정 사용자들이나 사용자 그룹들, 또는 특정 타입의 기계(machines)를 특정 이미지에 맵핑하는 것을 포함하여, 이미지 인덱스(618)에 대하여 이전에 기술된 것과 같은 기능을 수행할 수 있다. 예를 들어, 디바이스의 사용자는 시스템(602)에 로그인 할 수 있고, 그들이 디바이스(604) 상에 제공하고 싶은 이미지에 대한 특정 구성요소나 조각을 선택할 수 있는 하나 이상의 웹 페이지가 제공될 수 있다. 이와 같은 구성요소들은, 이미지를 재빌딩(rebuild)하기 위한 후속 시도들이 사용자에 의해 이전에 선택되었던 구성요소들을 자동으로 선택하도록, 사용자에 대한 식별자와 연계되어 저장될 수 있다.

[0077] 재구성된 이미지로, 이미지 업데이터(614)는, 일반적인 작동과 여기에 기술된 구성요소들 사이의 조정(coordination)을 제어할 수 있는 작동 인터페이스(operational interface)(612)와 협력하여, 이미지 인터페이스(610) 및 네트워크(606)을 통해 디바이스(604)로 업데이트된 이미지를 공급할 수 있다. 상기 업데이트는 디바이스(604)가 네트워크(606)를 통해서 서버 시스템(602)에 의해 우연히 또는 고의로 지워졌을(wiped) 때와 같은

경우, 디바이스(604)에 대한 전체 이미지를 포함할 수 있다.

[0078] 특정 실시예에서, 서버 시스템(602)은 디바이스(604)에서 이미지를 원격으로 지우고(wipe) 특정 경우에 그 이미지를 새로운 이미지로 대체하기 위해 이용될 수 있다. 예를 들어, 만약 디바이스(604)가 도둑맞으면, 시스템(602)은 디바이스(604) 상의 이미지를 지울 수 있고 디바이스(604)가 이미지를 다시 획득하는 것을 방지하기 위해 디바이스(604) 상의 펌웨어에 통합될 수 있는 식별자를 이용할 수 있다. 대안적으로, 소유자가 그 디바이스가 정당한 사용자 손에 돌아왔다는 것을 입증하면 서버 시스템(602)이 새롭고 업데이트된 이미지를 디바이스(604)에 제공할 수 있도록, 디바이스(604)는 시스템(602)에 의해 원격으로 지워질 수 있고 정당한 소유자(proper owner)에 의해 회복될 수 있다.

[0079] 예를 들어, 디바이스(604)가 초기에 부팅되고 있을 때, 공공키(public key)가 펌웨어의 읽기 전용 세그먼트(read-only segment)에 저장될 수 있고, 그 키는 디바이스(604) 상의 커널(kernel)을 체크하기 위해 사용될 수 있다. 상기 키, 또는 하나의 해쉬된 버전(hashed version)이나 디바이스 상의 하나 이상의 운영체제 구성요소의 다른 해쉬된 버전(상기 키 자체가 하나의 해쉬된 버전인 때)과 같은 다른 식별자는 부팅 시에 디바이스 상의 구성요소들에 대하여, 예를 들어 원본 식별자(original identifier)를 생성했던 동일 해쉬 기능(hash function)을 통해 그 구성요소들을 실행하여, 체크될 수 있다. 만약 매치가 이루어지지 않으면, 이는 디바이스(604)가 (변경되서는 안되는 때에 핵심 코드가 변경되었기 때문에) 제 기능을 발휘하지 못하는 것과, 디바이스(604)를 봇(bot)으로 만들거나 또는 다른 목적으로, 누군가 디바이스(600) 상의 기본 파일(native files)을 변경하려고 시도했다는 것을 가리킬 수 있다.

[0080] 로우레벨의 오퍼레이팅 스택(low-level of the operating stack)에서 통신 인터페이스는 이제 사용자가 부트 시퀀스(boot sequence)를 방해할 수 있기 전에 서버 시스템(602)으로의 네트워크 연결을 만들 수 있고, 디바이스(604) 상에서 상기 이미지의 실패가 발생했다는 것을 지시할 수 있다. 그러면 서버 시스템(602)은 위에서 설명한 디바이스(604)에 대한 새로운 이미지를 빌딩하고 그 새 이미지를 디바이스(604)로 전송할 수 있다. 디바이스(604)는 이제 잠재적으로 변질된 이미지를 새 이미지로 대체하고, 기능 상실 이미지(the compromised image)를 완료하는 것을 포함하여, 새 이미지를 작동시킬 수 있다.

[0081] 디바이스(604)는 또한 상기 기능 상실 이미지, 또는 기능 상실 이미지를 특징화 한 데이터를 서버 시스템(602)으로 돌려보낼 수 있다. 그러면 서버 시스템(602)은 무엇이 그 기능상실을 야기하였는지를 판단하기 위한 시도로서 상기 기능상실 이미지를 분석할 수 있다. 예를 들어, 서버 시스템(602)은 그 기능상실 이미지를 서버 시스템(602)에 의해 동일한 최초 이미지가 할당된 다른 디바이스들로부터의 기능 상실 이미지들과 비교할 수 있다. 만약 다수의 기능 상실 이미지들의 매칭이 식별되면, 그와 같은 식별은 그 디바이스들에 대해 단일 엔티티에 의해서 조직화된 공격(coordinated attack)이 발생했다는 것을 나타낼 수 있다.

[0082] 즉, 여기에 기술된 방법에서, 원격 재-이미지 시스템(a remote re-imaging system)(600)은 디바이스(604)와 같은 디바이스를 편리한 방법으로 업데이트 하기 위한 메커니즘을 제공할 수 있다. 예를 들어, 디바이스(604)는 많은 양의 코드를 지속적으로 저장하지 않기 때문에, 이미지에 대한 빈번한 업데이트는 디바이스(604)의 사용자에게 최소한의 분열(minimal disruptions)을 제공할 수 있다. 또한, 상술한 바와 같이, 클라우드-포스티드(cloud-posted) 이미지 데이터의 사용은, 특정 클라이언트 디바이스를 삭제하고, 서버 시스템(602)으로부터 그것들을 재이미지화(reimaging)하기 위한 편리한 방법을 제공한다. 부가하여, 전체 이미지를 함께 구성하는 변경가능한 구성요소들을 이용하고, 이 구성요소들을 사용자 디바이스 지도 데이터베이스(user device maps database)(622)의 특정 디바이스들에 맵핑함으로써, 모든 디바이스들 또는 특정 거대 그룹 디바이스들에 대해 공통인 빌딩 블록(building blocks)의 상대적으로 고유한 조합으로부터 특정 이미지를 재구성할 수 있도록, 서버 시스템(602)은 많은 수의 다른 사용자 디바이스들에 대한 이미지를 저장하도록 할 수 있다.

[0083] 도 7은 컴퓨팅 디바이스의 원격 모니터링 및 제어를 제공하는 시스템(700)의 개념 다이어그램이다. 일반적으로, 시스템(700)은 스마트폰이나 태블릿 컴퓨터 또는 랩탑 컴퓨터와 같은 휴대 컴퓨팅 디바이스의 형태일 수 있는 컴퓨팅 디바이스(702)를 포함한다. 컴퓨팅 디바이스(702)는, 웹 브라우저 형태의 오직 하나의 기본 애플리케이션을 갖고 다른 애플리케이션들은 그 웹 브라우저 상에서 실행되는 웹 앱의 형태로 제공되는 운영체제와 같은, 매우 가벼운(light) 운영체제로 로드되는 디바이스일 수 있다. 전술한 다른 디바이스들처럼, 디바이스(702)는 데스크탑 컴퓨터(706) 또는 서버 시스템(704)과 통신하기 위하여 인터넷과 같은 네트워크(708)를 통해 상호작용할 수 있다. 데스크탑 컴퓨터(706)는 컴퓨팅 디바이스(702)와 동일한 사람에 의해 소유되는 컴퓨터일 수 있다. 예를 들어, 사용자는 직장이나 집에서 데스크탑 컴퓨터를 가질 수 있고, 길에서 휴대 컴퓨팅 디바이스(702)를 사용할 수 있다. 여기에 기술된 시스템(700)은, 사기꾼이나 유사한 사람들이 디바이스(702)에 대한 운영체제에

간섭하는 것을 어렵게 만들고, 악성 코드 또는 콘텐츠를 체크하는 기회를 제공함으로써, 컴퓨팅 디바이스에서 보다 강력한 보안을 제공하는 것을 지향한다.

[0084] 부팅 프로세스 중 초기 단계에서 보안 위반을 식별하기 위한 일 예시 메커니즘이 도면에서 예시 스택(stack)에 의해 도시된다. 이 예시에서 상기 스택은 비교적 압축되어 있고, 가장 낮은 레벨인 바이오스(BIOS)(716)를 포함하고, 이는 펌웨어에 의해 구현될 수 있다. 바이오스는 전형적인 BIOS 동작을 수행할 수 있고, 또한 서버 시스템(704)과의 제한된 네트워크 연결을 구축하기 위한 코드를 포함할 수도 있다. 이 예시에서 제한된 연결은, (부팅 프로세스가 쉽게 해킹될 수 있는 레벨에 도달하기 전에) 디바이스(702)에서 서버 시스템(704)으로, 또는 서버 시스템(704)에서 디바이스(702)로 범죄적인 활동(nefarious activity)을 보고하기 위해 이용된다. 일 예시로서, 사용자는 디바이스(702)가 도난당했다는 것을 발견할 수 있고, 컴퓨터(706)를 이용하여 컴퓨터 시스템(704)에 로그온 할 수 있다. 사용자는 이제 디바이스(702)가 지워지거나, 종료되거나, 또는 현재 사용자의 사진을 찍고 디바이스(702)의 현재 위치를 보고할 것, 또는 다른 적절한 액션을 요청할 수 있다. 디바이스(702)가 다음에 부팅될 때, 바이오스(716)는, 무선 통신을 수행하기 위한 메커니즘을 포함하는 네트워크 하드웨어(718)를 경유하여 보내진 메시지를 통해 서버 시스템(704)으로 통지할 수 있다. 서버 시스템(704)은 상태 메시지를 응답할 수 있다. 대개 상기 상태 메시지는 모든 것이 정상이라는 것을 나타낼 수 있다. 그러나 사용자가 로그인하고 디바이스(702)가 사라졌다는 것을 보고하였기 때문에, 서버 시스템(704)은 이 예시에서 디바이스(702)에게 문제가 있다는 것을 나타낼 수 있다.

[0085] 서버 시스템(704)은, 네트워크(708)와 디바이스(702) 사이의 리턴 화살표(return arrow)로 도시된 것과 같이, 디바이스(702)의 후속 동작에 대한 명령을 제공할 수 있다. 예를 들어, 서버 시스템(704)은 디바이스(702)에게, 디바이스(702)가 누군가 디바이스(702) 상에서 타이핑 중이라는 것을 감지한 때에, 디바이스(702)의 사용자의 디지털 이미지를 촬영할 것을 명령할 수 있다. 완전히 부팅된 디바이스(702)는 현재 사용자(아마 도둑)에게 알리지 않고 자동으로 그 사진을 업로드 할 수 있다. 부가하여, 그와 같은 이미지는 디바이스(702)의 현재 위치를 나타내는 GPS 데이터를 동반할 수 있다. 이 방식에서, 사법 당국(law-enforcement authorities)은 어디로 가면 디바이스(702)를 되찾을 수 있는지 알 수 있고, 또한 도난 중에 특정 사용자가 그 디바이스(702)를 사용 중이었다는 증거를 가질 수 있다.

[0086] 디바이스(702)의 부팅에 대한 나머지 스택 또한 상대적으로 컴팩트(compact)하다. 예를 들어, 바이오스 코드(716)를 이용하여 기본적인 보안 체크가 이루어진 후에, 운영체제에 대한 코드(714)가 실행되고 운영체제의 기본 구성요소들이 런칭될 수 있다. 운영체제 런치(launch)의 일부 또는 유사하지만 별개인 액션의 일부로서, 서비스들(712)이 디바이스(702) 상에서 오픈되고 이용 가능해지며, 운영체제(714)에 대한 전용 기본 웹 브라우저 애플리케이션과 같은 애플리케이션들(710)이 런칭될 수 있다.

[0087] 보안 검사는 전체 부팅이 일어난 후에도 또한 자동 및 주기적으로 수행될 수 있다. 예를 들어, 감시 타이머(a watchdog timer 720)가 디바이스(702) 상에서 실행될 수 있고, 디바이스(702) 상의 보안이 주기적으로 체크되도록 할 수 있다. 예를 들어, 여기에 도시된 스택은 업데이트 될 때 그 자신의 시스템 파티션 내에 있을 수 있고, 해쉬 값(hash value)을 생성하기 위하여 함수(function)가 그 코드에 적용되도록 할 수 있다. 감시 타이머(720)가 트리거될(triggered) 때, 만약 운영체제가 각각의 이 구성요소들이 항상 스택 내에 있고 업데이트 중을 제외하고는 변경되지 않는다면, 유사한 해쉬가 그 시점에 스택이 포함하는 어떤 것에라도 수행될 수 있다. 상기 새로운 해쉬는 이제 스택에 대하여 저장된 해쉬와 비교될 수 있다. (해쉬들은 또한 스택 내의 모든 구성요소보다 적은 것에 대해 수행될 수 있다.) 만약 그 값이 변경되었으면, 서버 시스템(704)은 통지받을 수 있고, 나중에 디바이스(702)로 디바이스에 제제를 가하거나, 디바이스(702)의 스토리지를 지우거나, 디바이스의 저장 매체를 재포맷하거나, 또는 다른 그러한 동작을 수행하기 위한 신호를 보낼 수 있다. 서버 시스템(704)은 또한 디바이스(702)의 정당 사용자에게 작업 이메일 또는 텍스트 메시지 알림과 같은 백업 채널로 통지할 수 있다.

[0088] 디바이스(702)는 문제들을 서버 시스템(704)으로 보고하는데 있어서, 문제가 발견되자마자 즉시 또는 나중에 보고할 수 있다. 예를 들어, 바이오스(716)는 디바이스(702)에 대한 문제를 나타내는 식별자를 생성할 수 있다. 바이오스(716)는 이제 디바이스(702)가 완전하게 기능하도록(fully featured) 만들기 위하여 스택의 다른 구성요소들이 실행되도록 할 수 있다. 디바이스(702)가 완전하기 부팅되면, 네트워크 하드웨어(718)는, 서버 시스템(704)이 디바이스(702)가 문제를 가지고 있다는 것을 알고 이해하도록, 및 서버 시스템(704)이 재부팅(reboot), 특정 프로세스 또는 프로세스들을 재부팅, 및 디바이스(702) 상의 저장 매체를 삭제하거나, 디바이스(702) 상의 저장 매체를 재포맷(reformat)하기 위한 메시지와 같은 적절한 메시지를 디바이스(702)로 보낼 수 있도록, 서버 시스템으로 식별자가 제공하도록 될 수 있다.

[0089] 도 8은 호스트 컴퓨터 시스템 상에 집중적으로 저장된 컴퓨팅 디바이스의 데이터에 대한 캐싱(caching)을 제공하기 위한 시스템의 개념 다이어그램이다. 일반적으로, 시스템(800)은, 서버 시스템(804) 상의 스토리지(810)에 대한 캐시로 쓰기 위해 컴퓨팅 디바이스(802) 상의 스토리지(808)를 이용한다. 이와 같은 방법에서, 디바이스(802)의 동작은, 캐시된 데이터(808)가 많은 경우 디바이스(802)에 머무르고, 상대적으로 적은 경우에 처리 속도를 향상시키기 위해 네트워크(806)를 통한 왕복(round-trip)을 요구하는 것을 허용함으로써, 더욱 효율적이고 스피디하게 될 수 있다. 여기에서 디바이스(802)는, 웹 브라우저가 유일한 애플리케이션이고 다른 애플리케이션들은 상기 웹 브라우저의 웹 앱들로서 동작하는 운영체제를 갖는, 전술한 다른 디바이스들과 같이 구현될 수 있다. 또한, 도 7에 대해 설명한 스택이 디바이스(802) 상에서 구현될 수 있다.

[0090] 이 도면에서, 디바이스(802)는 3개의 특정 값: A, B, 및 C를 저장하는 것으로 도시된다. 유사하게, 서버 시스템(804)은 대응되는 방식으로, A 및 B의 형식으로 2개의 값을 저장하고 있다. 서버 시스템(804)은 또한 D에 대한 값도 저장하고 있다. 스토리지(810)에 저장된 A 및 B에 대한 값은 그 항목들이 "오염(dirty)"되었다는 것을 나타내기 위해 그 주위에 괄호로 도시되어 있고, 스토리지(810)로의 접근을 시도하는 다른 디바이스들에 의해 신뢰되지 않을 수 있는데, 이는 이들이 스토리지(808)에서 디바이스(802)에 의해 변경되었을 수 있기 때문이다. 근본적으로, 서버 시스템(804)은 디바이스(802)가 그 값들을 체크한 것처럼 동작한다 - 그럼에도 만약 다른 디바이스에 의해 그들에 대한 요청이 있으면, 서버 시스템(804)은 디바이스(802)에게 최신 값(latest values)을 획득하고 그들에 대한 제어를 디바이스(802)로부터 회수하였는지 문의할 수 있다. 또한, 도면에 도시된 바와 같이, 오직 항목 B만이 서버 시스템(804)에서 획득된 항목의 버전으로부터, 스토리지(808) 상의 항목 B에 작은따옴표(apostrophe)로 도시된 것처럼, 디바이스(802)에서 변경되었다.

[0091] 동작하는 동안, 디바이스(802)는 항목 A, B, 및/또는 C에 대한 값이 변경되는 결과를 가져오는 다양한 동작을 수행할 수 있다. 적절한 시점에, 디바이스(802)는 그와 같은 변화를 서버 시스템(804)으로 제출할 수 있고, 이는 스토리지(810)의 항목들에 대한 값들을 차례로 업데이트 할 수 있다. 디바이스(802)는 또한 그 항목들을 사용하는 것이 완료되었다는 것을 나타낼 수 있고, 이에 응답하여, 서버 시스템(804)은 시스템(800)에서 제어되고 있는 타입의 값들이라는 마크를 해제할 수 있다. 이 방식으로, 데이터는 디바이스(802) 상에서 실행중인 웹 앱과 서버 시스템(804)에 대한 데이터 저장 사이에서 편리하게 캐쉬될 수 있다.

[0092] 도 9는 컴퓨팅 디바이스의 지연 잠금을 제공하기 위한 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는, 예를 들어 클램쉘 디바이스의 커버를 닫거나 보안 슬립 모드로 진입하기 위해 태블릿 디바이스 상의 버튼 또는 다른 구성요소를 누름으로써 사용자가 디바이스를 클로즈(close)했을 때, 활성 모드로부터 슬립 모드와 같은 보안 또는 잠금 모드로의 경과(passage)를 지연시키기 위해 마련된 컴퓨팅 디바이스 상에서 실행될 수 있다. 이 예시에서, 보안 슬립 모드는, 그것을 다시 사용하기 위해 디바이스를 언락하기 위해서는 사용자가 패스워드를 입력하거나 또 다른 유사한 동작을 수행하여야 하는 모드이다. 일반적으로, 이와 같은 디바이스는, 재활성화 하기 위해 단순히 디바이스를 다시 키는 것 외에 활성화 입력(active input)을 요구하기 때문에, 잠겨 있는 것으로 참조된다.

[0093] 여기에 기술된 프로세스에 의해 의도적으로 도입되는 지연은, 사용자가 신속하게 마음을 바꾸고 그것이 완전히 슬립 모드로 진입하기 전에 그 디바이스를 다시 사용하는 것을 필요로 하는 경우, 사용자가 디바이스가 슬립 또는 잠금 모드로 진입하고자 한다는 것을 나타낸 직후에(short time period after) 사용자로 하여금 디바이스를 회복하는 것을 허용할 수 있다. 상기 지연이 적절하게 짧은 시간인 경우에, 그와 같은 딜레이는, 침입자가 잠금 모드로 들어가기 전에 디바이스를 취하지 못하도록, 정당한 최초 사용자가 지연 기간에 걸쳐 디바이스 근처에 머무르려 한다는 점에서, 그 프로세스에 대한 최소한의 보안 부담(security burden)을 가져올 수 있다.

[0094] 상기 프로세스는, 사용자가 그것이 특정 액션을 수행하여야 한다는 것을 나타냈는지 여부를 판단하기 위해 컴퓨팅 디바이스의 모니터링을 시작하는, 박스 902에서 시작한다. 박스 904에서, 프로세스는 잠금 시간(locking time)에 관한 사용자 입력을 수신한다. 예를 들어, 사용자는 그들의 컴퓨팅 디바이스가 슬립 모드로 변하기 전에 특정 사용자들에게 다양해질 수 있는 딜레이를 갖도록 설정할 수 있다. 예를 들어, 이 경우에, 사용자는 사용자가 그 또는 그녀의 마음을 바꾸고 디바이스의 활성 상태를 재구축하기 위한 적당한 시간을 제공하기 위한 9초의 지연을 식별할 수 있다. 박스 906에서, 디바이스를 잠그기 위한 지연 시간에 관한 사용자 입력에 응답하여, 프로세스는 그 디바이스에 대한 잠금 시간 파라미터를 조절한다. 이와 같은 파라미터는, 하나의 세션에서 다른 하나로, 디바이스를 잠그기 위한 지연 기간이 동일하도록, 및 만약 상기 시간이 만료되지 않았다면 사용자가 그 디바이스를 다시 동작시키는 것(reanimate)에 편안함을 느끼기 시작할 수 있도록 디바이스 상에 영구적으로 저장될 수 있다.

[0095] 박스 908에서, 프로세서는, 사용자가 디바이스 상의 파워 스위치를 누르도록 디바이스 상의 쉘 커버(shell cover)를 닫거나, 또는 디바이스를 끄거나 슬립 모드로 이동하도록 직접 스위치를 누르는 것에 의해, 디바이스가 오픈 설정(open configuration)에서 잠금 설정(closed configuration)으로 이동되는 것을 식별한다. 여기에서 논의되는 적절한 이행(relevant transition)은 디바이스에 대한 활성 작동 모드(active operable mode)로 돌아가기 위해 실질적인 사용자 입력을 요구하는, 상기 모드 변화에 응답하여 디바이스가 잠기지 않고 실질적인 구성요소들의 전원이 차단되지 않는다는 점에서, 스크린이 꺼질 수 있으나 쉽게 되돌릴 수 있는 디바이스를 잠금에서 오픈 설정으로 단순히 이동시키는 것 이상의 이행이다. 그렇기는 하지만, 사용자 입력이 수신되었다는 피드백을 주기 위해, 비록 그로부터 회복되기 위해 실질적인 사용자 입력을 요구하는 슬립모드로 디바이스가 이동하는 것에 있어 지연이 진행 중이라 하더라도, 디바이스의 스크린은 디바이스가 오픈에서 잠금 설정으로 이동한 직후에는 비어있을 수 있다.

[0096] 박스 910에서, 프로세스는 시간 파라미터의 만료 시에 디바이스의 잠금이 설정된 셧다운 타이머(shutdown timer)를 시작한다. 이 경우 시간 파라미터는 박스 904에서 사용자에 의해 선택되고, 박스 906에서 디바이스에 의해 적용된 파라미터다. 예를 들어, 디바이스는 스크린을 비우고 닫히자 마자 9초 카운트다운 타이머를 시작하지만, 타이머가 만료될 때까지 다른 모드로 이동하지 않을 수 있다. 즉, 912에서, 프로세스는 설정된 시간이 만료되었는지 여부를 판단하기 위해 반복적으로 체크한다. 만약 만료되지 않았다면 프로세스는 박스 914에서 디바이스가 오픈되었는지 여부를 체크한다. 만약 디바이스가 오픈되었다면, 프로세스는 박스 902에서 컴퓨팅 디바이스를 모니터링 하는 단계로 돌아간다. 후속하여, 사용자는 새 잠금 시간을 설정하거나, 또는 다시 디바이스를 오픈에서 잠금 설정으로 이동시킬 수 있고, 따라서 여기에 기술된 일부 또는 전부의 액션을 반복할 수 있다.

[0097] 타이머가 만료되면, 박스 916에서 프로세스는 디바이스를 잠근다. 이와 같은 디바이스 잠금은 그 디바이스를 활성화 상태로 다시 가져오기 위한 잠금 해제 입력(unlocking input)을 사용자에게 요구하는 것에 부가하여, 디바이스 상의 특정 프로세서들을 보여주거나 그들에 대한 전력을 전체적으로 제거하고, 디스플레이 스크린을 끄고, 공기 순환 팬(air circulation fans) 및 전력을 소모하는 다른 아이템들을 끄고, 및 디바이스를 재활성화 및 잠금 해제 하는 사용자를 기다리는 것을 수반할 수 있다.

[0098] 도 10a는 운영체제에서 콘텍스트 객체를 관리하기 위한 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는 운영체제 내의 객체들을, 그것들이 생성되었을 때, 및/또는 변경되거나 다른 경우 디바이스 상에서 조작되었을 때, 그 객체들 주위의 콘텍스트(contexts)를 기술하는 메타데이터와 연계시키는 것을 수반한다.

[0099] 상기 프로세스는 객체가 운영체제에서 인스턴스화되는 1002에서 시작한다. 상기 객체는 다양한 형태를 취할 수 있고, 여기서는 실례의 목적으로, 워드 프로세스 애플리케이션에서 편집될 수 있는 워드 프로세스 문서다. 박스 1004에서, 프로세스는 초기화 시점에 다른 오픈된 객체들의 상태를 정의하는 콘텍스트 메타데이터를 식별한다. 예를 들어, 사용자는 웹 브라우저가 특정 URL을 열도록 할 수 있고, 상기 프로세스는 그 웹 브라우저 및 URL에 대한 식별자를 저장할 수 있다. 이러한 콘텍스트 메타데이터는 만약 그것이 사용자가 웹 페이지를 보고 있었고, 그 웹 페이지에 관하여 워드 프로세스 프로그램에 기록하기로 결정하였는지를 지시하는 것과 관련이 있을 수 있다. 즉, 브라우저의 웹 페이지 URL과 워드 프로세스 애플리케이션의 워드 프로세스 문서 사이의 연관성을 생성하고 저장하는 것은 유리할 수 있다. 그리고 결과적으로, 콘텍스트 메타데이터는 박스 1006에서 저장되고, 인스턴스화된 객체에 관한 정보는 박스 1008에서 저장된다.

[0100] 박스 1010에서, 상기 객체는 잠시 후에, 예를 들어 사용자가 워드 프로세스 문서에 정보를 타이핑 하고 그 문서를 편집하는 것을 끝마친 후에 클로즈될 수 있고, 후속하여 그 객체를 오픈하기 위하여, 예를 들어 사용자가 그 문서를 조금 더 편집하고자 할 때, 요청이 수신될 수 있다. 객체가 다시 오픈되면 그 시점에 다양한 다른 애플리케이션이 실행될 수 있고, 그것들은 이 문서가 사용자에 의해 열렸던 이유와 다시 관련될 수 있다. 다시, 예를 들어, 브라우저들이 사용자에 관련된, 더 나아가 상기 객체인 그 문서에 관련된 토픽에 대해서 오픈될 수 있다.

[0101] 결과적으로, 그리고 그러한 정보를 캡쳐하기 위해서, 박스 1012에서, 프로세스는 메타데이터를 인스턴스화된 데이터 외에 다른 오픈된 객체들에 대한 정보로 업데이트한다. 즉, 객체 그 자체를 나타내는 도메인 파일의 일부일 수 있는, 상기 객체에 대한 메타데이터 저장소(repository)가, 사용자가 계속해서 그 객체를 열고, 닫고, 및 조작함에 따라 생성되고, 추가되고, 업데이트 될 수 있다. - 여기서 부가되는 데이터는, 동시에 실행 중이던 다른 애플리케이션들에 의해 정의되는 것을 포함하여, 각 상황에 존재하는 객체 상의 콘텍스트(context), 및 그 다른 애플리케이션들이 무엇을 하고 있었는지에 관한 정보를 나타낸다. 따라서, 박스 1014에서, 사용자는 객체의 닫고 엶(closing and opening)의 다른 사이클(cycles)을 경험하고, 메타데이터는 업데이트된다.

[0102] 이러한 콘텍스트 메타 데이터는, 사용자의 의도를 충분히 나타내는 것으로 판단될 때, 자동 액션을 수행하기 위해 최종적으로 사용될 수 있다. 예를 들어, 사용자가 문서를 열 때, 시스템은 동시에 브라우저 내의 검색엔진에 대한 검색을 수행할 수 있고, 검색 결과를 그 문서와 함께 디스플레이 할 수 있다 - 만약 콘텍스트 메타데이터의 분석이 사용자가 전형적으로 문서를 열면 검색을 수행했다는 것을 나타내는 경우. (예를 들어, 그 문서는 회사에 대한 주가(stock prices)를 추적하고, 그 또는 그녀가 그 문서로 잘라내기-및-붙여 넣기를 할 수 있도록 그 회사에 대한 현재 주가에 대해 검색하는 경우)

[0103] 도 10b는 운영체제에서 콘텍스트 객체를 관리하기 위한 프로세스의 흐름도이다. 도 10a에 도시된 프로세스와 같이, 상기 프로세스는 운영체제 내의 객체들을, 그것들이 생성되었을 때, 및/또는 변경되거나 다른 경우 디바이스 상에서 조작되었을 때, 그 객체들 주위의 콘텍스트(contexts)를 기술하는 메타데이터와 연계시키는 것을 수반한다.

[0104] 프로세스는 컴퓨팅 디바이스에서 오픈된 제1 운영체제 객체 상에서 제1 미리 정의된 액션의 제1 발생이 기록되는 1022에서 시작한다. 박스 1024에서, 제1 운영체제 객체와 동시에 컴퓨팅 디바이스 상에 오픈된 제2 운영체제 객체 상에서 제2 미리 정의된 액션의 제1 발생이 기록된다. 상기 객체는 워드 프로세스 문서, 이미지 파일, 특정 URL을 디스플레이 하는 웹 브라우저 등과 같은, 다양한 형태를 취할 수 있다.

[0105] 다음에, 박스 1026에서, 제1 운영체제 객체 및 제2 운영체제 객체에 각각 관련된 제1 및 제2 미리 정의된 액션이 연관된다. 박스 1028에서 제1 및 제2 운영체제 객체들에 관련된 상기 연관된 액션들을 포함하는 연관된 상태가 저장된다. 상기 연관된 상태들을 저장하는 데이터베이스가 유지된다. 이러한 상태들을 저장하는 것은, 특정 시점에서 다양한 객체들에 대하여 취해진 액션의 히스토리, 즉 객체 연관성(object correlations)의 히스토리를 저장하는 것을 제공한다. 그렇게 하는 것은 다양한 객체들 사이의 관계가 결정되도록 하기 위하여 컴퓨터가 특정 객체를 포함하는 모든 연관된 상태를 검색하기 위해 데이터베이스에 접근하는 것을 허용한다.

[0106] 박스 1030에서, 제1 운영체제 객체 상에서 제1 미리 정의된 액션의 제2 발생이 식별된다. 박스 1032에서, 제2 발생은 이제 제1 운영체제 객체 상에서의 제1 미리 정의된 액션을 포함하는 데이터베이스 내의 연관된 상태들에 비교된다. 상기 비교는, 박스 1034에 도시된 바와 같이, 제2 운영체제 객체 상에서의 제2 미리 정의된 액션이 제1 운영체제 객체 상에서의 제1 미리 정의된 액션과 관련되는 판단을 산출해낸다. 따라서, 박스 1036에서, 제2 운영체제 객체는 인스턴스화된다. 연관성은 어느 주어진 시간에 컴퓨터 상에서 오픈된 객체들의 "스냅샷(snapshot)"을 찍음으로써 구출될 수 있다. 연관성은 또한 개별 객체들에 관련된 메타 데이터를 저장함으로써 결정될 수 있고, 여기서 메타 상태(meta state)는 도 2a 및 2b에 대하여 전술한 바와 같이, 다른 객체들 또는 사용자들에 관한 정보를 포함한다.

[0107] 도 11은 컴퓨팅 디바이스 상에서 메모리 제어를 유지하기 위한 프로세스의 흐름도이다. 프로세스 1100은 예를 들어, 시스템(400)과 같은 시스템에 의해 수행될 수 있다. 그러나, 다른 시스템, 또는 시스템들의 조합이 프로세스 1100을 수행하기 위해 사용될 수 있다.

[0108] 도 11을 참조하면, 프로세스 1100은 컴퓨팅 디바이스 상에서 잠재적인 메모리 부족을 관리하기 위한 예시적인 방법에서 애플리케이션 라이프사이클(lifecycle)을 나타낸다. 상기 방법은 애플리케이션이 실행되는 단계 1105에서 시작한다. 예를 들어, 운영체제는 사용자 요청에 응답하거나 다른 애플리케이션의 명령(behest)에 대해 지속성 메모리로부터 애플리케이션을 로드할 수 있다. 다음으로, 단계 1110에서, 시스템(400)은 애플리케이션의 상태가 변경되었는지 여부를 판단한다. 일부 실시예에서, 애플리케이션 모니터(220)는 언제 애플리케이션이 포커스를 맞추는지 언제 새로운 애플리케이션이 생성되는지, 또는 언제 애플리케이션이 종료되었는지 주목할 수 있다. 대안적인 실시예에서, 전술한 변경들에 부가하여, 또는 그 대신에, 애플리케이션 매니저는 언제 애플리케이션이 상태를 변경하였는지 판단하기 위해 시스템에 대한 사용자 입력(예를 들어, 키 입력, 마우스 클릭, 스타일러스 또는 손가락 탭 등)을 모니터 할 수 있다.

[0109] 만약 상태 변화가 탐지되지 않으면, 상기 방법은 시스템이 메모리 부족이 존재하는지 여부를 판단하는 단계 1115로 진행한다. 만약 시스템이 메모리 부족이 존재한다고 판단하면, 상기 방법은 이하에서 설명되는 단계 1130으로 진행한다. 그러나 만약 메모리 부족이 존재하지 않으면, 단계 1110은 반복된다.

[0110] 상태 변경이 탐지되면, 상기 방법은 단계 1120으로 진행한다. 단계 1120에서, 상태정보는 생성되고 저장된다. 특정 상태에 있는 애플리케이션은 언제라도 종료될 수 있기 때문에, 애플리케이션 모니터가 상태 변경을 탐지한 후에, 애플리케이션 모니터는 애플리케이션에게 상태정보를 생성하고 지속성 메모리에 저장할 것을 명령할 수 있다. 실제적인 실시예에서, 상태 정보는 그 애플리케이션을 종료되기 전에 존재했던 애플리케이션으로 재생성

하기 위해 사용되는 정보를 포함할 수 있다. 예를 들어, 상태정보는 디스플레이 상의 애플리케이션 윈도우의 위치, 사용자에 의해 생성된 애플리케이션 파일에 대한 어떤 변경, 및 사용자가 선호하는 뷰 모드(view mode)를 포함할 수 있으나, 이에 제한되지는 않는다. 상태정보가 생성되면, 애플리케이션은 그 상태정보를 지속성 메모리에 저장할 수 있다.

[0111] 상태정보가 생성되고 저장된 후, 상기 방법은 시스템이 메모리 부족이 존재하는지 여부를 판단하는 단계 1115로 진행한다. 만약 시스템이 메모리가 고갈되었다고 판단하면, 커널은 애플리케이션 종료기(application terminator)에게 하나 이상의 애플리케이션을 종료하여 메모리를 이용 가능하게 만들 것을 명령할 수 있다. 대안적인 실시예에서, 커널은 메모리가 부족해지고 있는지 여부를 판단할 수 있다. 커널은 애플리케이션으로부터의 메모리 요청을 현재 이용 가능한 메모리와 비교하여 임박한(imminent) 메모리 부족을 탐지할 수 있다. 커널은 또한 이용 가능한 메모리의 양이 미리 결정된 임계 값보다 낮은지 여부를 판단할 수 있다. 만약 부족이 탐지되면, 커널은 선택된 애플리케이션이 일반적인 동작을 계속할 수 있도록 충분한 메모리를 갖는 것을 보장하기 위해 충분한 메모리를 마련하기 위한 단계들을 취할 수 있다.

[0112] 만약 시스템이 메모리가 필요하다고 판단하면, 상기 방법은 애플리케이션 종료기가 애플리케이션이 가장 낮게 랭크된 애플리케이션인지, 즉 그 애플리케이션이 애플리케이션 계층(application hierarchy)에서 최하위(bottom)에 있는지 여부를 판단하는 단계 1130으로 나아간다. 만약 애플리케이션이 계층(224)의 최하위에 있지 않으면, 상기 방법은 시스템이 애플리케이션을 상태 변경에 대하여 모니터 하는 단계 1110으로 돌아간다. 만약 애플리케이션 계층의 최하위에 있으면, 상기 방법은 그 애플리케이션이 종료되는 단계 1140으로 진행한다. 일부 실시예에서, 애플리케이션 종료기는 그 애플리케이션의 프로그램 스택에서 마지막 애플리케이션인지 여부를 판단한다. 만약 그렇다면, 그 애플리케이션 및 어떤 연관된 스레드(thread)라도 종료된다.

[0113] 단계 1150에서, 시스템은 애플리케이션이 회복(revive)되어야 하는지 판단한다. 실제적인 실시예에서, 만약 시스템이 종료된 애플리케이션을 되돌리려는 사용자 시도를 탐지하면, 상기 방법은 단계 1105로 돌아가고, 여기서 애플리케이션이 저장된 상태 정보를 이용하여 회복된다. 예를 들어, 시스템은 종료된 애플리케이션에 오버레이 되어 있는 윈도우를 닫거나 최소화 하기 위한 사용자의 시도를 탐지한다. 이에 대응하여, 시스템(400)은 지속성 메모리에서 그 애플리케이션에 대한 상태정보를 불러오고 그 애플리케이션을 회복하기 위해 상태정보를 이용한다.

[0114] 만약 시스템(400)이 종료된 애플리케이션과 상호작용하는 시도를 탐지하지 않으면, 상기 방법은 단계 1150으로 돌아가고 여기서 시스템(400)은 그 애플리케이션을 회복시킬지 여부를 다시 판단한다.

[0115] 도 12는 컴퓨터 프로세스 사이의 메시지 패싱(message passing)에 쓰레드 관련성(thread affinity)을 제공하기 위한 프로세스의 흐름도이다. 여기에 도시된 프로세스는 일반적으로 단일 컴퓨터 상에서 실행 중인 프로세스들 사이에서 일어나고 있다. 예를 들어, 상기 2개의 프로세스들은 하나의 디바이스 상에서 실행중인 두 개의 다른 애플리케이션을 나타낼 수 있고, 다른 애플리케이션에 대해 디바이스 상의 메모리를 보호하기 위하여 각자로부터 샌드박스 되도록 마련된다. 프로세스는 각 프로세스가 인스턴스화되는 박스 1202에서 시작한다.

[0116] 박스 106에서, 및 프로세스가 잠시 동안 실행된 후에, 프로세스 A는 프로세스 B에 의한 액션 또는 프로세스 B로부터 정보를 다시 수신하는 것이 필요하다는 것을 판단할 수 있다. 그리고 결과적으로, 박스 106에서, 프로세스 A는 메시지와 함께 프로세스 B가 프로세스 A를 대신하여 얼마나 빨리 메시지에 대한 작업을 필요로 하는 지를 정의하는 정보를 보낼 수 있다. 따라서, Paul Blake 박스에서, 프로세스 B는 그 메시지를 수신하고, 박스 110에서 우선순위 정보로부터 메시지의 메인 바디를 분리한다. 프로세스 B는 이제 우선순위 정보 또는 프로세스 A로부터 그 메시지를 어떻게 처리할 지 판단하기 위해 메시지와 함께 보내졌을 수 있는 다른 정보를 사용한다. 예를 들어, 프로세스 A는 그 작업을 완료하기 위해 컴퓨터 디바이스의 현재 상태에 대한 보고를 필요로 할 수 있고, 만약 실질적으로 느리고 프로세스로부터 이야기하는 정보를 기다리는 것이 사용자에 의해 즉시 필요로 한다면, 프로세스 A는 모든 다른 것들보다 그 동작을 우선시 할 것을 프로세스 B에게 전하는 정보를 제공할 수 있다. 따라서, 박스 Paul 112에서, 프로세스 B는 그 우선순위 정보에 기초하여 그 포커스를 변경할 수 있다. 예를 들어, 만약 프로세스 B가 이전에 오래 걸리지만 시간에 민감하지 않은 작업 중이었다면, 그것은 그러한 활동에 대한 상태 정보를 저장하라고 진술할 수 있고, 프로세스 B 상의 그 작업을 한쪽으로 치워두라고 프로세스 A에 대해 현명하게 응답했을 수 있다. 이 방식으로, 프로세스들은 프로세스들을 통해 이루어지는 요청에 대해 더욱 집중된 응답을 제공하기 위하여 편리한 방법으로 서로 통신을 주고받을 수 있다.

[0117] 도 13은 상태 비 보존형 환경에서(in a stateless environment) 상태 정보를 제공하기 위한 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는 웹 브라우저 또는 웹 브라우저에서 실행중인 하나 이상의 웹 앱의 판(pan

e)과 같은 애플리케이션의 상태에서의 변경을 추적(tracking)하고, 그 변경에 관한 정보를 그러한 정보를 저장하는 서버 시스템으로 보내는 것을 수반한다. 나중에, 동일한 사용자에 대해 로그인 된 컴퓨터는 서버로부터 상태 정보를 획득하고 그에 따라 마지막으로 저장된 상태를 재생성할 수 있다.

[0118] 프로세스는 사용자가 일반적인 방식으로 컴퓨팅 디바이스를 작동시키는 박스 1302에서 시작한다. 컴퓨팅 디바이스는, 웹 브라우저 형태의 단일 기본 애플리케이션을 실행하고 다른 애플리케이션들은 웹 브라우저 내에서 웹 앱으로 실행되는 운영체제를 실행할 수 있다. 브라우저 내의 각 판(pane) 및 각 웹 앱은 보안 목적으로 그 자신의 독립된 프로세스에서 샌드박스될 수 있다.

[0119] 박스 1304에서, 상기 프로세스는 주기적으로 또는 디바이스 상의 상태 변경에 응답하여, 디바이스 상의 하나 이상의 활성 DOMs의 이미지를 저장할 수 있다. 그러한 활동(activity)은 관련 상태가 웹 페이지의 현재 상태일 때, 예를 들어 사용자가 웹 페이지 상의 활성 콘텐트(예를 들어, 자바스크립트-생성 콘텐트(JavaScript-created content))와 상호작용 하였을 때 발생할 수 있다. DOM 정보에 대안으로서 또는 그 정보에 부가하여, 다른 상태 정보가 디바이스 및 디바이스 내의 다른 프로세스들의 현재 상태를 완전히 캡쳐하기 위해 저장될 수 있다. 예를 들어, 디바이스 상의 각 활성 프로세스들의 리스트는 유지 및 갱신될 수 있고, 그 프로세스들의 현재 상태를 정의하는 특정 파라미터들도 또한 업데이트 될 수 있다. 그러한 정보는, 동시에 또는 나중에 정보를 컴퓨팅 디바이스로 제공하고 있는 서버로 업로드 될 수 있고, 그 서버 시스템은 최종-저장된 디바이스의 상태가 나중에 재구성될 수 있도록 그 정보를 조직할 수 있다.

[0120] 박스 1308에서, 그와 같은 재구성에 대한 요청이 발생한다. 예를 들어, 서버 시스템에 동일한 사용자로 제1 디바이스 등록된 디바이스에 대한 부팅 프로세스는 그 등록된 디바이스의 사용자 계정에 대한 가장 최근의 상태 정보를 획득하기 위해 자동으로 그 서버 시스템에 접근할 수 있다. 서버 시스템은 특정 디바이스에 대한 저장된 이미지에 접근할 수 있다. (여기서 사용자 또는 사용자 그룹은 디바이스들이 부팅될 때 자동으로 취하게 되는 그들의 디바이스에 대한 이미지를 정의할 수 있다.) 또한, 디바이스는 서버 시스템에 사용자 계정으로 등록된 컴퓨터를 사용자가 마지막으로 사용한 시점에 있었던 상태로 웹 페이지와 다른 객체들을 재구성하기 위한 이미지의 구성요소들과 연관된 데이터를 사용할 수 있고, 이전의 컴퓨터와 나중 컴퓨터는 동일하거나 서로 다르다.

[0121] 박스 1314에서, 상기 프로세스는 상기 페이지들 및 다른 객체들을 이전에 저장된 상태로 온전히 디스플레이한다.

[0122] 도 14는 네트워크를 통해 컴퓨팅 디바이스에 대한 이미지를 제공하는 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는 여기에 도시된 바와 같이, 호스트 서버 시스템과 통신하는 하나 이상의 클라이언트 디바이스들에 대해 일어난다. 이 예시에 도시된 특정 작업 분담은 예시적인 목적으로 제공될 뿐이며, 유사한 구성요소 또는 다른 실시예에서 다른 구성요소에 의해 다른 액션들이 취해질 수 있다. 여기에 도시된 프로세스는 일반적으로 각각이 부팅시에 어떻게 클라이언트 디바이스가 작동할 것인지를 정의하는 이미지들을 구성하기 위한 이미지 데이터를 저장하는 것을 수반한다. 상기 이미지들은 중앙 호스트 서버 시스템에서 저장되고 재구성되며, 이미지 데이터는 부팅 시에 클라이언트 디바이스들로 제공된다.

[0123] 프로세스는 다수의 클라이언트 디바이스들이 디바이스의 사용자들에 의해 일반적인 방식으로 동작하는 박스 1402에서 시작한다. 각각의 사용자들은 박스 1404에서와 같이, 그들의 특정 디바이스 상에서 이미지에 대한 파라미터들을 구축할 수 있고, 박스 1406에서 그 이미지들을 수신할 수 있는 호스트 서버 시스템으로 그 디바이스 이미지들을 제출할 수 있다. 예를 들어, 사용자는 (웹 앱과 같은) 특정 애플리케이션들이 그들의 디바이스를 부팅할 때 로드되길 원할 수 있고, 특정 방식으로 설정된 디바이스 세트(device set) 상의 설정 값(settings)을 원할 수 있다.

[0124] 박스 1408에서, 서버 시스템은 특정 디바이스들에 대한 이미지들을 서버 시스템에 저장된 데이터에 비교한다. 예를 들어, 서버 시스템은 다양한 다른 이미지들을 함께 구성하는 구성요소들을 저장할 수 있고, 단순히 각 구성요소의 하나의 복사본(copy), 및 그 구성요소들을 클라이언트 디바이스들에 대한 각 이미지들에 맵핑하는 기본적인 본문(textual) 또는 유사한 파일을 저장할 수 있다. 예를 들어, 숫자들의 바이너리 리스트가 전체 이미지를 식별하기 위해 이용될 수 있고, 여기에서 상기 리스트에서 각 포지션은 특정 이미지 내에 존재하거나 그렇지 않을 수 있는 특정 구성요소를 식별할 수 있고, 특정 디바이스에 대한 그 포지션에서 하나의 존재는 서버 시스템에게 그 디바이스가 그의 이미지 내에 특정 구성요소를 가지고 있다는 것을 지시할 수 있다. 따라서, 예를 들어 박스 1410에서, 상기 프로세스는 그 디바이스 이미지의 일부를 더 작은 동의어(synonyms)로 교체할 수 있다. 직전에 기술된 예시에서, 상기 동의어는 단순한 비트 값(bit values)일 수 있고, 반면에 다른 실시예에서, 동의어는 시스템에서 특정 구성요소에 대해 유일한 영숫자(alphanumeric) 식별자일 수 있다.

[0125] 박스 1412에서, 서버 시스템은 축소된 사이즈 이미지(reduced size image)를 디바이스가 특정 이미지에 속해있다는 것을 지시하는 디바이스 식별자와 함께 저장할 수 있다. 예를 들어, 시스템은 일 위치에서 특정 구성요소에 대한 코드를 저장할 수 있고, 위에서 논의한 바이너리 스트림(binary stream)과 함께 및 다른 위치에 디바이스에 대해 고유한 디바이스 식별자를 저장할 수 있다.

[0126] 나중에, 특정 디바이스는, 박스 1414에 나타난 바와 같이, 이미지를 요청할 수 있고, 서버 시스템은 박스 1316에서, 그 디바이스에 대한 이미지에 접근하고 그것을 확장할 수 있다. 일 예시로서, 디바이스는 부팅시마다 이미지를 요청할 수 있고, 꺼질 때는 로컬에(locally) 그 이미지를 저장하지 않을 수 있다. 그러한 확장은 위에서 설명한 것과 같이 서버 시스템에 바이너리 리스트를 통해 스테핑(stepping)하는 것과 그 리스트 내의 각 값들이 0 보다는 1인 구성요소들을 모으는(gathering) 것을 수반할 수 있다. 구성요소를 모으거나 디바이스에 대한 이미지를 빌딩하는 다른 기술들도 또한 사용될 수 있다.

[0127] 박스 1418에서, 서버 시스템은 빌딩된 이미지를 클라이언트 디바이스로 다시 전송하고, 박스 1420에서 클라이언트 디바이스는 그 이미지를 로드하고 디바이스가 디바이스의 사용자에 의해 완전하게 동작하는 것을 허용한다. 그리고 여기에서 설명된 기술을 사용하여, 이미지 데이터는, 특히 많은 수의 디바이스들이 그 시스템에 대해서 동작할 때, 시스템에 걸쳐 그 사이즈가 줄어드는 방식으로 호스트 시스템에 저장될 수 있다.

[0128] 도 15는 컴퓨팅 디바이스의 제어 및 원격 모니터링을 제공하기 위한 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는 체크가 일어나는 레벨 이하의 레벨에서 스택으로 진입하는 해커들의 능력을 저하시키기 위하여, 운영체제 스택의 로우-레벨에서 보안 검사(checks)를 수행하여 클라이언트 컴퓨팅 디바이스에 대한 보안을 제공한다.

[0129] 프로세스는 디바이스가 사용자에 의해 켜지는 박스 1502에서 시작한다. 박스 1504에서 디바이스는 그 부트 펌웨어(boot firmware)에 접근하고 부트 프로세스를 친숙한 방식으로 시작한다. 상기 펌웨어는 바이오스 또는 디바이스 상의 다른 구조의 일부일 수 있다. 펌웨어는 또한 디바이스와 호스트 서버 시스템 사이의 무선 통신을 위한 메커니즘을 포함할 수 있고, 박스 1506에서 프로세스는 호스트 서버 시스템으로 메시지를 보내기 위해 그러한 기능을 이용할 수 있다. 상기 메시지는 디바이스가 호스트 서버 시스템에 마지막으로 체크된 이래로 디바이스의 환경에 있어서 변경(change)을 식별하기 위한 요청으로 취급될 수 있다. 박스 1508에서, 서버 시스템은 상기 메시지를 수신하고 디바이스 및 디바이스의 파라미터들을 식별한다. 예를 들어, 호스트 서버 시스템은 디바이스에 관련된, 디바이스의 사용자가 그 디바이스를 도둑맞았고 그것을 잠그거나 지우거나 재포맷 해야 한다는 것을 보고하는 것과 같은, 어떤 이벤트라도 발생하였는지 판단하기 위해 디바이스 식별자를 사용할 수 있다. 부가하여, 디바이스의 아이덴티티(identity)는, 도 14에 대하여 설명된 바와 같이, 디바이스에 대한 이미지를 식별하고 디바이스의 추가적 부팅을 위해 디바이스로 다시 그 이미지를 제공하기 위한 구성요소들을 모으기 시작하기 위해 사용될 수 있다.

[0130] 박스 1510에서, 서버 시스템은, 디바이스 사용자의 보고 및 디바이스의 분실 여부에 대하여 바로 위에서 논의된 것과 같은, 디바이스에 관해 저장된 정보에 접근한다. 박스 1512에서, 서버 시스템은 디바이스 상에서 수행될 액션들에 대한 데이터를 생성 및 전송한다. 도 14에서 설명한 바와 같이, 그러한 액션은 부팅에서 디바이스에 의해 실행되어야 하는 디바이스에 대한 이미지를 제공하는 것을 포함할 수 있다. 그러한 액션은 또한 디바이스에 대하여 구현되는 보안과 관련될 수 있다. 예를 들어, 만약 부트 펌웨어에 의해 전송된 메시지가 디바이스 상에서 그 부분의 스택이 최근 업데이트 이래로 변경되었다는 것을 나타내면, 디바이스 상에서 수행될 상기 액션들은 디바이스를 지우거나 디바이스의 플래시 메모리 또는 하드 드라이브와 같은 저장 구조를 재포맷하는 것을 포함할 수 있다. 유사한 액션들이 디바이스를 잃어버린 디바이스의 사용자로부터의 외부 지시(indication)에 응답하여 수행될 수 있다. 따라서, 디바이스 상의 액션들은 디바이스로부터 부팅 시에 수신된 정보에 반응하거나, 디바이스로부터 수신된 것은 아니지만 보고에 있어서 사용자 전화(calling)와 같은 외부 소스로부터 수신된 정보에 반응할 수 있다.

[0131] 박스 1514에서, 클라이언트 디바이스는 데이터 및 명령을 수신하고, 박스 1516에서, 클라이언트는 부트 펌웨어를 수신된 데이터를 처리 및/또는 수신된 명령어를 실행하기 위해 사용한다. 수신될 수 있는 다른 타입의 명령들은 디바이스가 회복될 수 있도록 디바이스에 관한 정보를 다시 보고하는 명령을 포함한다. 예를 들어, 상기 디바이스는 전술한 바와 같이, 그 디바이스를 사용중인 도둑의 이미지를 캡쳐하기 위한 시도로 온보드 카메라(onboard camera)를 이용하여 디지털 포토를 찍을 수 있고, 그 포토를 그 디바이스의 위치를 나타내는 GPS 데이터와 함께 전송할 수 있다.

[0132] 도 16은 호스트 컴퓨터 시스템 상에 집중적으로 저장된 컴퓨팅 디바이스의 데이터에 대한 캐싱(caching)을 제공

하기 위한 프로세스의 흐름도이다. 일반적으로, 상기 프로세스는 어떻게 웹 기반 컴퓨팅 디바이스에 대한 특정 데이터가 일반적으로 호스트 서버 시스템에 저장되지만, 또한 호스트 서버 시스템과 통신하고 웹 브라우저 내의 웹 앱들을 포함하는 디바이스 상의 애플리케이션들을 작동시키기 위해 호스트 서버 시스템에 의존하는 클라이언트 디바이스 상으로 캐쉬될 수 있는지를 나타낸다.

[0133] 프로세스는 클라이언트 디바이스 및 호스트 서버 디바이스 모두 디바이스에 대한 계정-기반 정보(account-based information)를 저장하는 1602 및 1604에서 시작한다. 예를 들어, 각 디바이스 또는 시스템은 그 디바이스를 호스트 서버 시스템에 대한 특정 사용자 계정과 연관시키는 디바이스에 대한 사용자 ID를 저장할 수 있다. 박스 1606에서, 클라이언트 디바이스는 서버 측 정보를 요청하고, 박스 1608에서 호스트 서버 시스템은 요청된 정보를 획득하고 그것을 클라이언트 디바이스로 전송한다. 호스트 서버 시스템은 또한 미래에 그 요청 정보에 대해 이루어질 변경들을 식별하기 위한 정보를 저장할 수도 있다. 특정 실시예에서, 호스트 서버 시스템이 클라이언트 디바이스로 그것이 보내졌을 때, 그러한 데이터를 오염될 것으로(as being dirty) 표시(mark)하거나 적어도 예비적으로 표시할 수 있도록, 호스트 서버 시스템은 편집을 위해 클라이언트 디바이스로 제공된 어떤 정보도 편집될 것이라고 가정할 수 있다. 박스 1612에서, 클라이언트 디바이스는 요청된 정보를 수신하고, 클라이언트 디바이스와의 사용자 상호작용에 응답하여, 그 요청된 정보를 수정할 수 있다. 적절한 시점에 상기 수정된 정보는 클라이언트 디바이스에서 호스트 서버 시스템으로 다시 전송된다. 이와 같은 적절한 시간은 서버 시스템으로 정보를 다시 주기적으로 제공하는 시계, 또는 사용자가 애플리케이션에서 정의된 제어를 선택하는 것과 같은 특정 이벤트에 의해 결정될 수 있다.

[0134] 박스 1616에서, 서버 시스템은 수정된 정보를 수신하고, 박스 1618에서 요청된 정보에 대한 변경을 식별하기 위해 저장된 정보를 사용한다. 예를 들어, 시스템은 클라이언트로부터 다시 수신된 정보를, 클라이언트 디바이스로 주어진 정보에 어떤 관련 변경이 이루어졌는지 여부를 판단하기 위해 이전에 오염(dirty)이 표시된 정보에 비교할 수 있다. 박스 1620에서, 서버 시스템은 그 정보의 서브셋(subset)을 업데이트 된 서브셋으로 교체할 수 있다. 따라서, 예를 들어 시스템이 시스템 상의 오염 데이터와 클라이언트 디바이스로부터 다시 받은 데이터 사이의 매치가 없다고 판단하는 경우, 그 서버 시스템은 클라이언트 디바이스로부터 변경된 정보를 삽입할 수 있다.

[0135] 도 17은, 여기에 설명된 기술이 이용될 수 있는 포괄적인 컴퓨터 디바이스(1700) 및 포괄적인 모바일 컴퓨터 디바이스(1750)를 나타낸다. 컴퓨팅 시스템(1700)은 랩탑, 데스크톱, 워크스테이션, PDA(Personal Digital Assistant), 서버, 블레이드(blade) 서버, 메인 프레임, 및 그 밖의 적절한 컴퓨터들과 같은 다양한 형태의 디지털 컴퓨터를 나타내기 위해 사용된다. 컴퓨팅 디바이스(1750)는 PDA, 셀룰러 폰, 스마트폰, 및 그 밖의 유사한 컴퓨팅 디바이스와 같은 다양한 형태의 모바일 디바이스를 나타내기 위해 사용된다. 본 명세서에서 나타낸 구성요소, 그들의 접속 및 관계, 및 그들의 기능들은 단지 예시를 의미하고, 본 명세서에서 설명하거나 또는 청구된 발명의 구현예를 제한하는 것을 의미하지 않는다.

[0136] 컴퓨팅 디바이스(1700)는 프로세서(1702), 메모리(1704), 저장 디바이스(1706), 메모리(1704)와 고속 확장 포트(1710)에 접속하는 고속 인터페이스(1708), 및 저속 버스(1714)와 저장 디바이스(1706)에 접속하는 저속 인터페이스(1712)를 포함한다. 각 구성요소(1702, 1704, 1706, 1708, 1710, 및 1712)는 다양한 버스들을 사용하여 서로 접속되고, 공통 마더보드에 탑재되거나 또는 적절한 경우 다른 방식으로 탑재될 수 있다. 프로세서(1702)는 컴퓨팅 디바이스(1700) 내에서 실행하기 위한 명령어를 처리할 수 있으며, 이러한 명령어에는, 고속 인터페이스(1708)에 연결된 디스플레이(1716)와 같은 외장 입/출력 디바이스 상에서 GUI용 그래픽 정보를 디스플레이 하기 위해, 메모리(1704) 또는 저장 디바이스(1706)에 저장되는 명령어가 포함된다. 다른 구현예에서, 다중 프로세서 및/또는 다중 버스는 적절한 경우, 다중 메모리 및 메모리 타입과 함께 사용될 수 있다. 또한, 다중 컴퓨팅 디바이스(1700)는 각 디바이스가 필요 동작의 부분을 제공하는 형태(예를 들어, 서버 뱅크, 블레이드 서버의 그룹, 또는 다중 프로세서 시스템)로 접속될 수 있다.

[0137] 메모리(1704)는 컴퓨팅 디바이스(1700) 내에 정보를 저장한다. 일 실시예에서, 메모리(1704)는 휘발성 메모리 유닛 또는 유닛들이다. 또 다른 실시예에서, 메모리는 비휘발성 메모리 유닛 또는 유닛들이다. 메모리(1704)는 또한 마그네틱 또는 광학 디스크와 같은, 다른 형태의 컴퓨터-판독 가능한 매체일 수도 있다.

[0138] 저장 디바이스(1706)는 컴퓨팅 디바이스(1700)를 위한 대용량 저장소(mass storage)를 제공할 수 있다. 일 실시예에서, 저장 디바이스(1706)는 플로피 디스크 디바이스, 하드 디스크 디바이스, 광 디스크 디바이스, 또는 테이프 디바이스, 플래시 메모리 또는 다른 유사한 고체 상태 메모리 디바이스, 또는 저장 영역 네트워크 또는 다른 구성에 존재하는 디바이스를 포함하는 디바이스 어레이와 같은 컴퓨터-판독가능 매체이거나 그 매체를 담고

있을 수 있다. 컴퓨터 프로그램 제품은 정보 캐리어(information carrier) 내에 유형적으로 구체화된다. 또한, 컴퓨터 프로그램 제품은 실행될 때, 상술한 것과 같은 하나 이상의 방법을 수행하는 명령어를 포함한다. 정보 캐리어는 메모리(1704), 저장 디바이스(1706), 프로세서(1702) 상의 메모리, 또는 전파 신호와 같은 컴퓨터- 또는 기계-판독가능 매체이다.

[0139] 저속 제어부(1712)가 저대역-집약적 동작(lower bandwidth-intensive operations)을 관리하는 반면, 고속 제어부(1708)는 컴퓨팅 디바이스(900)에 대한 대역-집약적 동작을 관리한다. 이러한 기능(duties)들의 배치는 단지 예시적인 것이다. 일 구현예에서, 고속 제어부(1708)는 메모리(1704), 디스플레이(1716)(예를 들어, 그래픽 프로세서 또는 가속기를 통함)에 연결되고, 다양한 확장 카드(도시되지 않음)을 수용할 수 있는 고속 확장 포트(1710)에 연결된다. 일부 구현예에서는, 저속 제어부(1712)는 저장 디바이스(1706) 및 저속 확장 포트(1714)에 연결된다. 다양한 통신 포트(예를 들어, USB, 블루투스, 이더넷, 무선 이더넷)를 포함할 수 있는 저속 확장 포트는 키보드, 포인팅 디바이스, 스캐너와 같은 하나 이상의 입/출력 디바이스들에 연결되거나, 또는 예컨대 네트워크 어댑터를 통하여, 스위치나 라우터와 같은 네트워킹 디바이스에 연결될 수 있다.

[0140] 컴퓨팅 디바이스(1700)는 도면에 도시된 바와 같이, 복수의 다른 형태로 구현될 수 있다. 예를 들어, 컴퓨팅 디바이스(1700)는 표준 서버(1720)로 구현되거나 이러한 서버들의 그룹에서 여러 번(multiple time) 구현될 수 있다. 또한, 컴퓨팅 디바이스(1700)는 랙 서버 시스템(1724)의 부분으로서 구현될 수 있다. 추가적으로, 컴퓨팅 디바이스(1700)는 랩탑 컴퓨터(1722)와 같은 개인용 컴퓨터 내에 구현될 수 있다. 대안적으로, 컴퓨팅 디바이스(1700)로부터의 구성요소는 디바이스(1750)와 같은 모바일 디바이스(도시되지 않음) 내 다른 구성요소와 조합될 수 있다. 이러한 디바이스 각각은 하나 이상의 컴퓨팅 디바이스(1700, 1750)를 포함하고, 전체 시스템은 서로 통신하는 다중 컴퓨팅 디바이스(1700, 1750)로 구성될 수 있다.

[0141] 컴퓨팅 디바이스(1750)는 여러 구성요소 중에서 프로세서(1752), 메모리(1764), 디스플레이(1754)와 같은 입/출력 디바이스, 통신 인터페이스(1766), 및 트랜스시버(1768) 등을 포함한다. 또한, 디바이스(1750)에는 추가적인 저장소를 제공하기 위하여, 마이크로 드라이브 또는 다른 디바이스와 같은 저장 디바이스가 제공될 수 있다. 구성요소(1750, 1752, 1764, 1754, 1766, 및 1768) 각각은 다양한 버스를 이용하여 서로 접속되고, 구성요소의 몇몇은 공통의 마더보드에 탑재되거나 적절한 다른 방법으로 탑재될 수 있다.

[0142] 프로세서(1752)는 컴퓨팅 디바이스(1750) 내에서의 실행을 위해 명령어를 실행하며, 이 명령어에는 메모리(1764)에 저장된 명령어가 포함된다. 프로세서는 별개이며 다수의 아날로그 및 디지털 프로세서들을 포함하는 칩들의 칩셋으로서 구현될 수 있다. 프로세서는, 예를 들어, 사용자 인터페이스의 컨트롤, 디바이스(1750)에 의해 실행되는 애플리케이션, 및 컴퓨팅 디바이스(1750)에 의한 무선 통신과 같은 디바이스(1750)의 다른 구성요소들 사이에 조정을 제공할 수 있다.

[0143] 프로세서(1752)는 제어 인터페이스(1758) 및 디스플레이(1754)에 연결된 디스플레이 인터페이스(1756)를 통해 사용자와 통신할 수 있다. 디스플레이(1754)는, 예를 들어, TFT LCD(Thin-Film-Transistor Liquid Crystal Display) 디스플레이 또는 OLED(Organic Light Emitting Diode) 디스플레이, 또는 다른 적절한 디스플레이 기술일 수 있다. 디스플레이 인터페이스(1756)는 그래픽 및 다른 정보를 사용자에게 나타내기 위해 디스플레이(1754)를 구동하는 적절한 회로를 포함할 수 있다. 제어 인터페이스(1758)는 사용자로부터 명령들을 수신하고, 프로세서(1752)에 제출하기 위해 그 명령들을 변환한다. 더욱이, 확장 인터페이스(1762)는 디바이스(1750)와 다른 디바이스들 간에 근거리 통신이 가능하도록 하기 위해, 프로세서(1752)와의 통신에 제공될 수 있다. 확장 인터페이스(1762)는, 예를 들어, 일부 구현예에서는 유선 통신을 제공하고 다른 구현예에서 무선 통신을 제공하며, 또한 다중 인터페이스가 사용될 수 있다.

[0144] 메모리(1764)는 컴퓨팅 디바이스(1750) 내에 정보를 저장한다. 메모리(1764)는 컴퓨터 판독가능 매체 또는 미디어, 휘발성 메모리 유닛 또는 유닛들, 또는 비휘발성 메모리 유닛 또는 유닛들 중 하나 이상으로서 구현될 수 있다. 또한, 확장 메모리(1774)가 제공되어, 예를 들어 SIMM(Single In Line Memory Module) 카드 인터페이스를 포함하는 확장 인터페이스(1774)를 통해 디바이스(1750)에 접속될 수 있다. 이러한 확장 메모리(1774)는 디바이스(1750)를 위한 여분의 저장 공간을 제공할 수 있고, 또한 애플리케이션 또는 디바이스(1750)를 위한 다른 정보를 저장할 수 있다. 특히, 확장 메모리(1774)는 상술된 프로세스를 실행하거나 보조하기 위한 명령어를 포함하고, 또한 보안 정보를 포함할 수 있다. 따라서, 예를 들어, 확장 메모리(1774)는 디바이스(1750)용 보안 모듈(security module)로서 제공될 수 있고, 디바이스(1750)의 안전한 사용을 가능하게 하는 명령어로 프로그램될 수 있다. 더욱이, 보안 애플리케이션은, 해킹할 수 없는 방식(non-hackable manner)으로 SIMM 카드 상에 식별 정보를 위치시킨 것과 같은 추가적 정보와 함께 SIMM 카드를 통해 제공될 수 있다.

[0145] 메모리는 아래에서 논의되는 것과 같이 예를 들어, 플래시 메모리 및/또는 NVRAM 메모리를 포함할 수 있다. 일 구현예에서, 컴퓨터 프로그램 제품은 정보 캐리어에 유형적으로 구체화된다. 컴퓨터 프로그램 제품은 실행될 때, 상술된 것과 같은 하나 이상의 방법을 수행하는 명령어를 포함한다. 정보 캐리어는 메모리(1764), 확장 메모리(1774), 프로세서(1752) 또는 예를 들어 트랜스시버(1768) 또는 확장 인터페이스(1762)를 통해 수신될 수 있는 전파된 신호(propagated signal)와 같은 컴퓨터- 또는 기계-판독가능 매체이다.

[0146] 디바이스(1750)는 디지털 신호 처리 회로를 필요에 따라 포함하는 통신 인터페이스(1766)를 통해 무선으로 통신할 수 있다. 통신 인터페이스(1766)는 GSM 음성 호, SMS, EMS, 또는 MMS 메시징, CDMA, TDMA, PDC, WCDMA, CDMA2000, 또는 GPRS 등과 같은 다양한 모드 또는 프로토콜 하에서의 통신을 제공할 수 있다. 이러한 통신은 예를 들어, 무선-주파수 트랜스시버(1768)를 통해 수행될 수 있다. 또한, 단거리(short range) 통신은 예를 들어, 블루투스, Wi-Fi, 또는 다른 이러한 트랜스시버(도시되지 않음)를 사용하여 수행될 수 있다. 이에 더하여, GPS(Global Position System) 수신기 모듈(1770)은 추가적인 네비게이션- 및 위치- 관련 무선 데이터를 디바이스(1750)에 제공할 수 있으며, 이 무선 데이터는 디바이스(1750)에서 실행중인 애플리케이션에 의해 적절하게 사용될 수 있다.

[0147] 또한, 디바이스(1750)는 사용자로부터의 발화 정보(spoken information)를 수신하고, 그 발화 정보를 사용 가능한 디지털 정보로 변환하는 오디오 코덱(1760)을 이용하여, 청취 가능하게(audibly) 통신할 수 있다. 또한, 오디오 코덱(1760)은 예를 들어, 디바이스(1750)의 핸드셋 내의 스피커를 통하는 것과 같이 해서, 사용자가 들을 수 있는 음성을 생성한다. 이러한 음성은 음성 전화 호로부터의 음성을 포함할 수 있고, 녹음된 음성(예를 들어, 음성 메시지, 뮤직 파일 등)은 포함할 수 있고, 또한 디바이스(1750) 상에서 동작하는 애플리케이션에 의해 생성된 음성을 포함할 수 있다.

[0148] 컴퓨팅 디바이스(1750)는 도면에 도시된 것처럼, 다수의 다양한 형태로 구현될 수 있다. 예를 들어, 컴퓨팅 디바이스(1750)는 셀룰러 전화(1780)로서 구현될 수 있다. 또한, 컴퓨팅 디바이스(1750)는 스마트폰(1782), PDA, 또는 다른 유사한 모바일 디바이스의 일부로서 구현될 수 있다.

[0149] 본 명세서에 기재된 시스템의 다양한 구현예와 기술은 디지털 전자 회로, 집적 회로, 특별하게 설계된 ASICs(Application Specific Integrated Circuit), 컴퓨터 하드웨어, 펌웨어, 소프트웨어, 및/또는 그것의 조합물로 실현될 수 있다. 이러한 다양한 구현예는 하나 이상의 컴퓨터 프로그램으로 된 구현예를 포함하며, 이 컴퓨터 프로그램은 적어도 하나의 프로그램 가능한 프로세서를 포함하는 프로그램 가능한 시스템에서 실행가능하고 및/또는 해석 가능하다. 또한, 전용 또는 범용 프로세서일 수 있는 이 프로그램 가능한 프로세서는 데이터와 명령어를 송수신하기 위해, 저장 시스템, 적어도 하나의 입력 디바이스 및 적어도 하나의 수신 디바이스에 연결된다.

[0150] 컴퓨터 프로그램(또한 프로그램, 소프트웨어, 소프트웨어 애플리케이션, 또는 코드로 알려짐)은 프로그램 가능한 프로세서를 위한 기계 명령어를 포함하고, 고레벨 절차 및/또는 객체 지향 프로그램 언어(object-oriented programming language) 및/또는 어셈블리/기계 언어로 구현될 수 있다. 본 명세서에서 사용되는 바와 같이, 용어 "기계 판독가능 매체(machine-readable medium)"와 "컴퓨터 판독가능 매체(computer-readable medium)"는 기계 명령어 및/또는 데이터를 프로그램 가능한 프로세서에 제공하기 위해 이용되는 임의의 컴퓨터 프로그램 제품, 장치, 및/또는 디바이스(예를 들어, 마그네틱 디스크, 광학 디스크, 메모리, PLDs(Programmable Logic Devices))를 가리키며, 기계 판독가능 신호와 같은 기계 명령어를 수신하는 기계 판독가능 매체를 포함한다. 용어 "기계 판독가능 신호(machine-readable signal)"는 기계 명령어 및/또는 데이터를 프로그램 가능한 프로세서에 제공하기 위해 사용되는 임의의 신호를 가리킨다.

[0151] 사용자와의 상호 작용을 제공하기 위하여, 본 명세서에 기술된 시스템과 기술은, 정보를 사용자에게 디스플레이하기 위한 디스플레이 디바이스(예를 들어, CRT(cathode ray tube) 또는 LCD 모니터)와 사용자가 컴퓨터에 입력을 제공할 수 있는 키보드 및 포인팅 디바이스(예를 들어, 마우스 또는 트랙볼)를 구비한 컴퓨터 상에서 구현될 수 있다. 사용자와의 상호 작용을 제공하기 위하여 다른 종류의 디바이스가 또한 사용될 수 있다; 예를 들어, 사용자에게 제공되는 피드백(feedback)은 임의의 형태의 감각 피드백(예를 들어, 시각 피드백, 청각 피드백 또는 촉각 피드백)일 수 있고, 사용자로부터의 입력은 음향(acoustic), 음성(speech) 또는 촉각(tactile) 입력을 포함하는 임의의 형태로 수신될 수 있다.

[0152] 본 명세서에서 설명한 시스템과 기술은, 백 엔드(back end) 구성요소(예를 들어, 데이터 서버와 같은), 또는 미들웨어 구성요소(예를 들어, 애플리케이션 서버), 또는 프론트 엔드(front end) 구성요소(예를 들어, 본 명세서에서 설명된 시스템 및 기술의 구현예와 사용자가 상호 작용할 수 있는 그래픽 사용자 인터페이스 또는 웹 브라

우저를 구비한 클라이언트 컴퓨터), 또는 이러한 백 엔드, 미들웨어, 또는 프론트 엔드 구성요소들의 임의의 조합을 포함하는 컴퓨팅 시스템으로 구현될 수 있다. 시스템의 구성요소는 디지털 데이터 통신의 임의의 형태 또는 매체(예를 들어, 통신 네트워크)에 의해 상호 접속될 수 있다. 통신 네트워크의 예로서, 근거리 네트워크("LAN"), 광역 네트워크("WAN"), 및 인터넷이 있다.

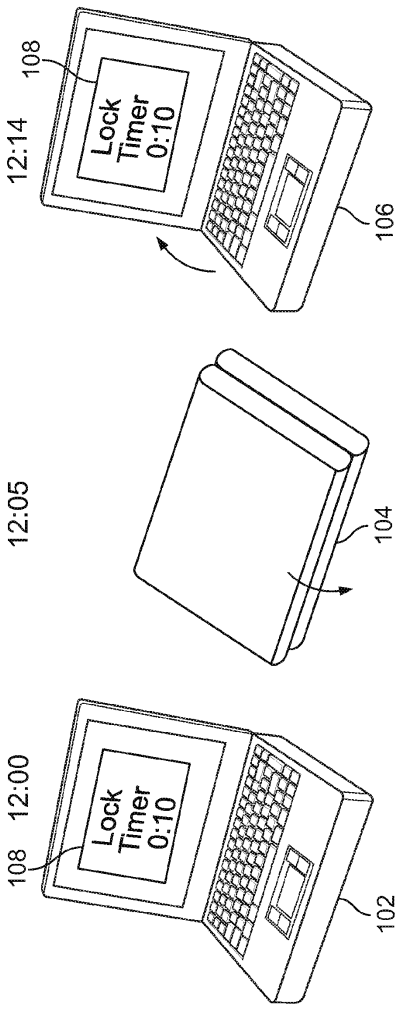
[0153] 컴퓨팅 시스템은 클라이언트와 서버를 포함할 수 있다. 클라이언트와 서버는 보통 서로 떨어져 있으며, 일반적으로는 통신 네트워크를 통하여 상호 작용한다. 클라이언트와 서버의 관계는 각각의 컴퓨터 상에서 실행되고 상호 클라이언트-서버 관계를 갖는 컴퓨터 프로그램에 의하여 발생한다.

[0154] 다수의 실시예들이 기술되었다. 그럼에도 불구하고, 본 발명의 범위 및 사상으로부터 벗어나지 않고 다양한 변형들이 가해질 수 있음이 이해될 것이다. 예를 들어, 본 명세서의 다수의 설명들이 텔레비전 광고들을 참조하여 이루어졌으나, 미래의 다른 형태들, 라디오 광고들 및 온-라인 비디오 광고들과 같은 시청자(viewership)-기반의 광고들 또한 고려될 수 있다.

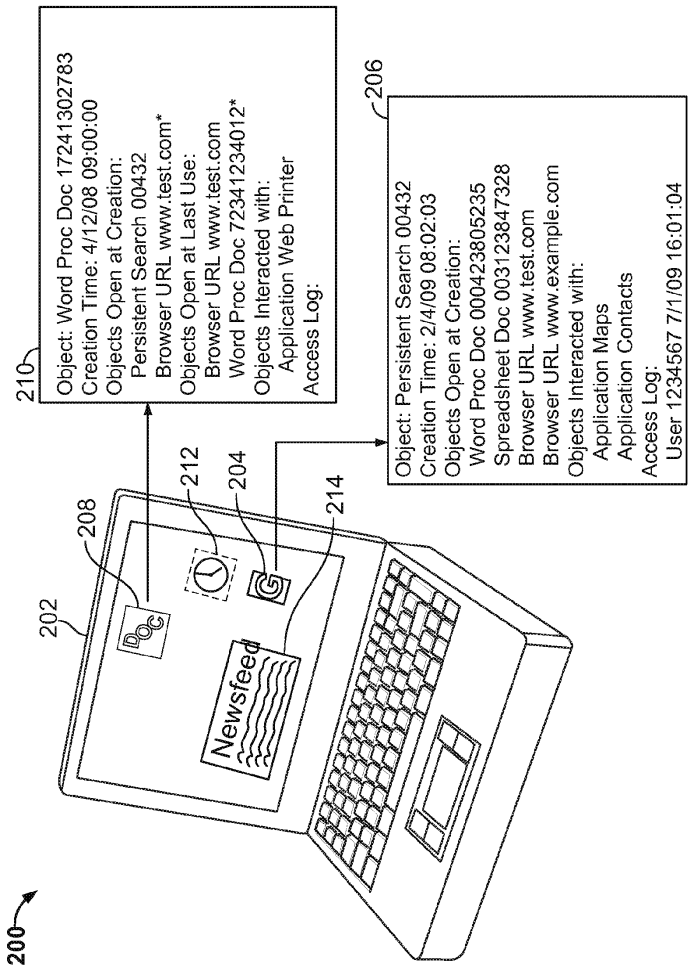
[0155] 더욱이, 도면에서 묘사된 논리 흐름은 희망하는 결과를 달성하기 위해, 도시된 특정 순서 또는 시계열적 순서일 필요는 없다. 덧붙여, 다른 단계들이 제공되거나, 그로부터 단계들이 제거될 수 있으며, 다른 구성요소들이 설명된 시스템에 추가되거나 그로부터 제거될 수 있다. 따라서 다른 실시예들은 후술하는 청구범위의 범위 내에 속한다.

도면

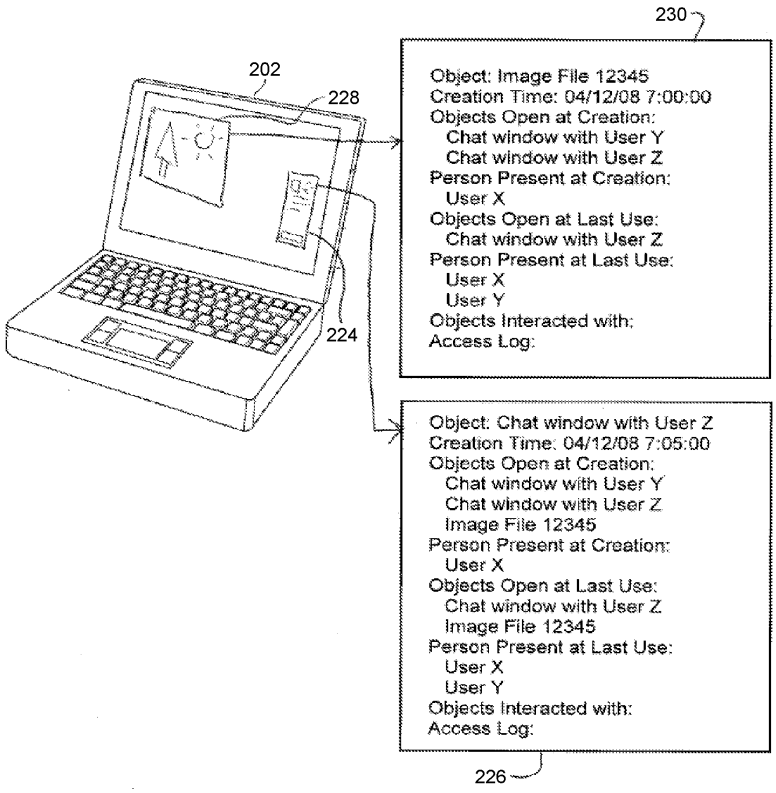
도면1



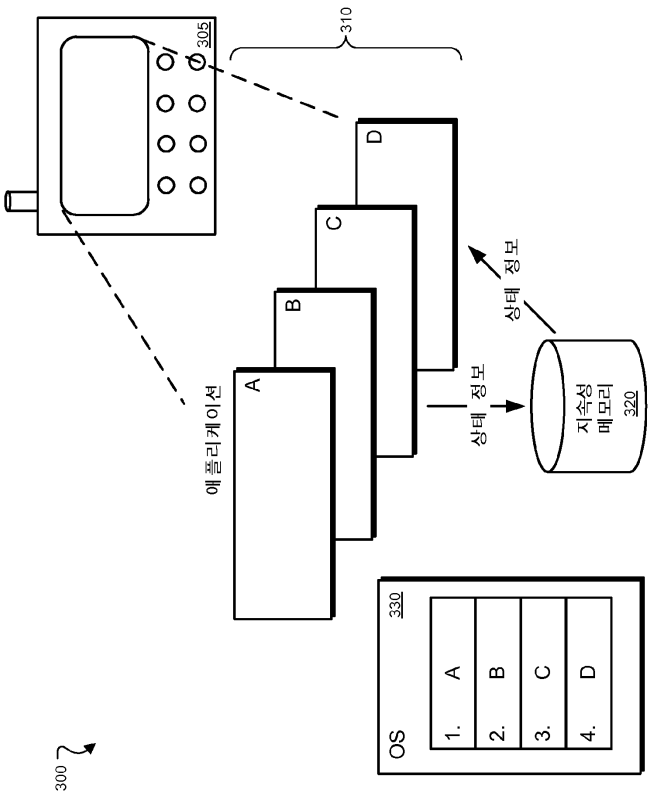
도면2a


200
202
208
212
204
214
Newsfeed
210
Object: Word Proc Doc 17241302783
Creation Time: 4/12/08 09:00:00
Objects Open at Creation:
Persistent Search 00432
Browser URL www.test.com*
Objects Open at Last Use:
Browser URL www.test.com
Word Proc Doc 72341234012*
Objects Interacted with:
Application Web Printer
Access Log:
206
Object: Persistent Search 00432
Creation Time: 2/4/09 08:02:03
Objects Open at Creation:
Word Proc Doc 000423805235
Spreadsheet Doc 003123847328
Browser URL www.test.com
Browser URL www.example.com
Objects Interacted with:
Application Maps
Application Contacts
Access Log:
User 1234567 7/1/09 16:01:04

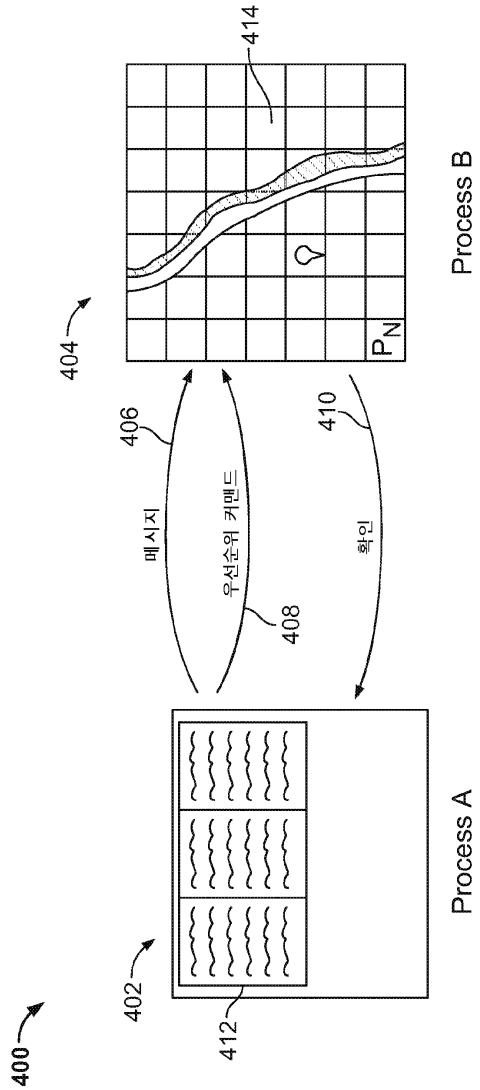
도면2b


230
202
228
224
Object: Image File 12345
Creation Time: 04/12/08 7:00:00
Objects Open at Creation:
Chat window with User Y
Chat window with User Z
Person Present at Creation:
User X
Objects Open at Last Use:
Chat window with User Z
Person Present at Last Use:
User X
User Y
Objects Interacted with:
Access Log:
Object: Chat window with User Z
Creation Time: 04/12/08 7:05:00
Objects Open at Creation:
Chat window with User Y
Chat window with User Z
Image File 12345
Person Present at Creation:
User X
Objects Open at Last Use:
Chat window with User Z
Image File 12345
Person Present at Last Use:
User X
User Y
Objects Interacted with:
Access Log:
226

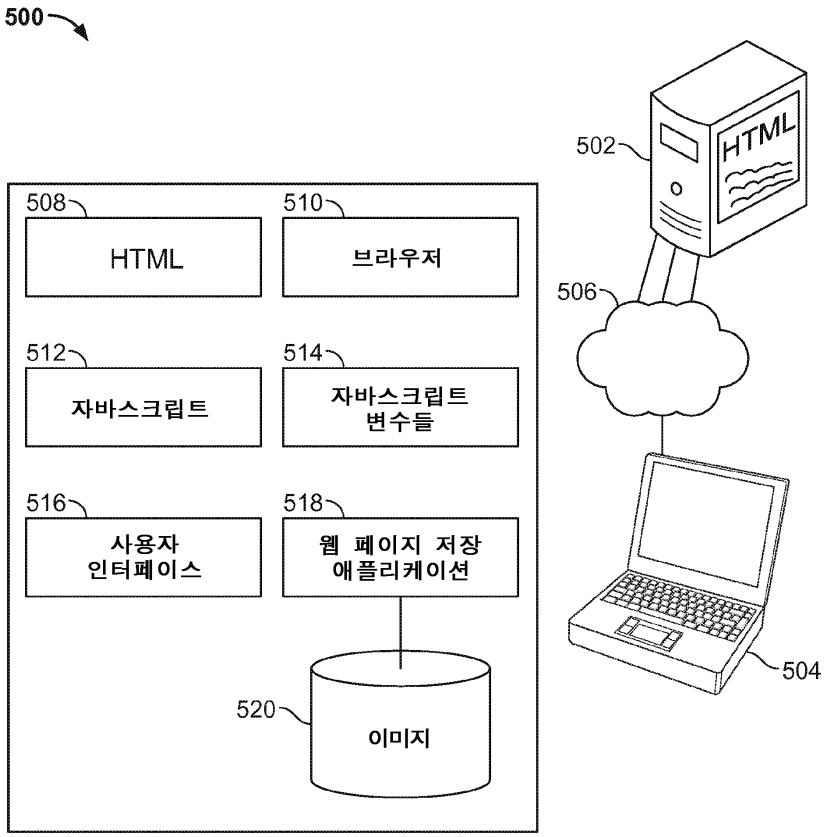
도면3


305
310
D
C
B
A
에플리케이션
상태 정보
상태 정보
지속성 메모리 320
OS
330
1. A
2. B
3. C
4. D
300

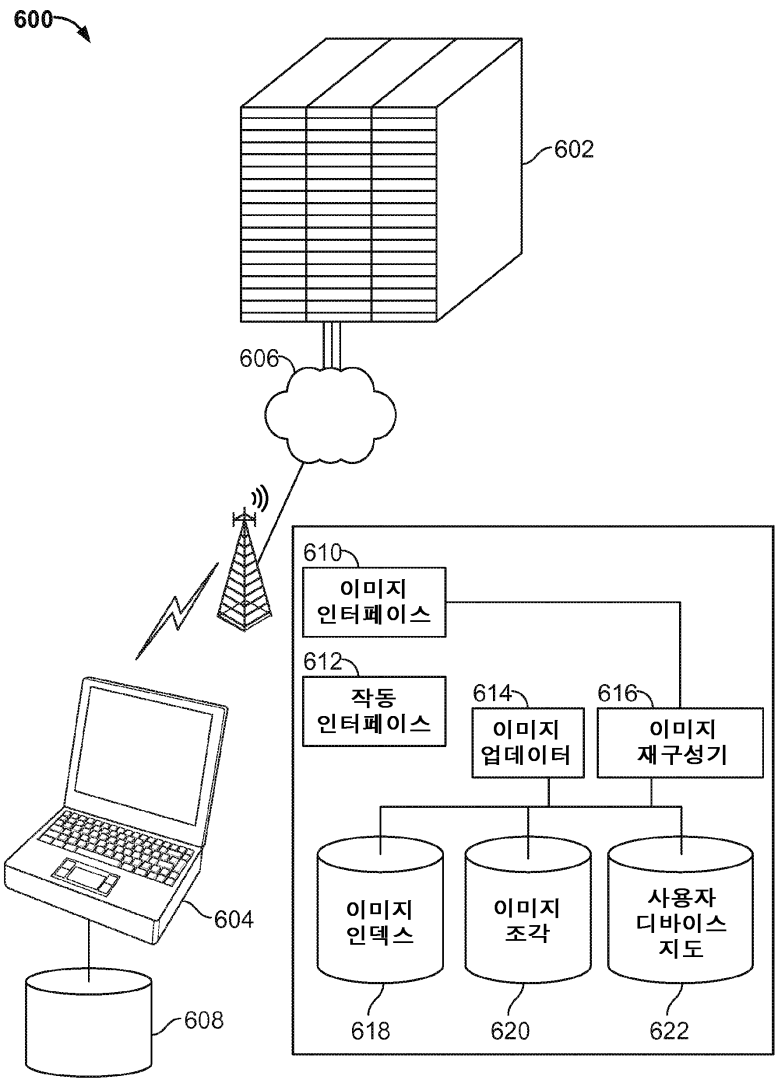
도면4


400
402
404
406
408
410
412
414
Process A
Process B
메시지
우선순위 커맨드
확인
P_N

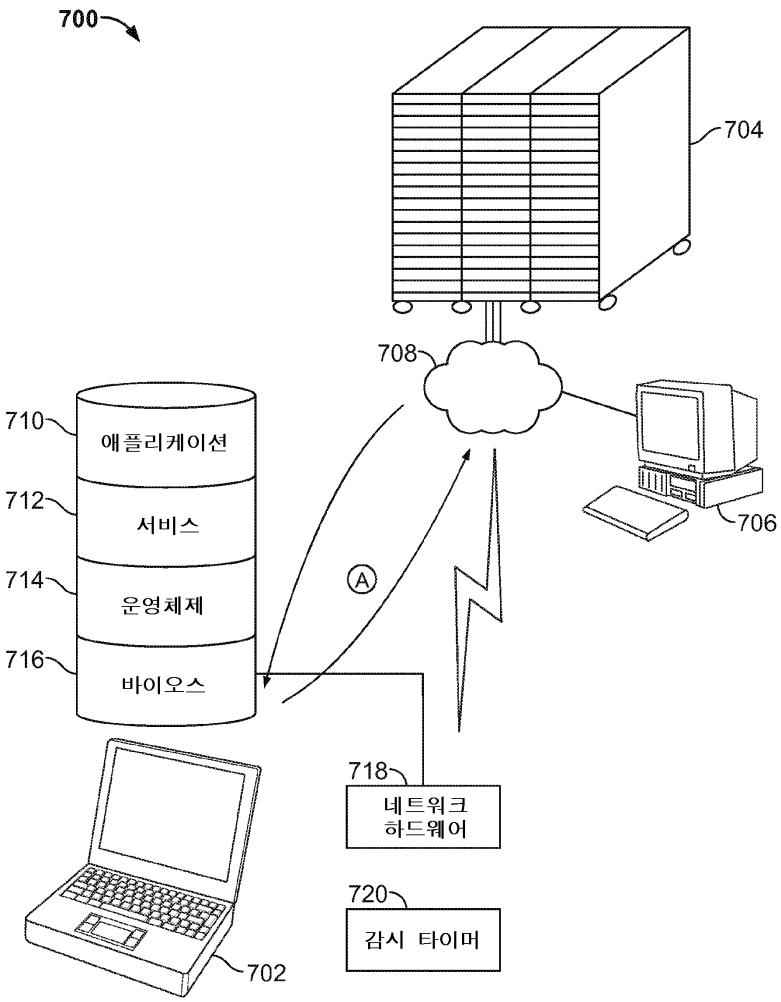
도면5


500
502
HTML
506
504
508
HTML
510
브라우저
512
자바스크립트
514
자바스크립트
변수들
516
사용자
인터페이스
518
웹 페이지 저장
애플리케이션
520
이미지

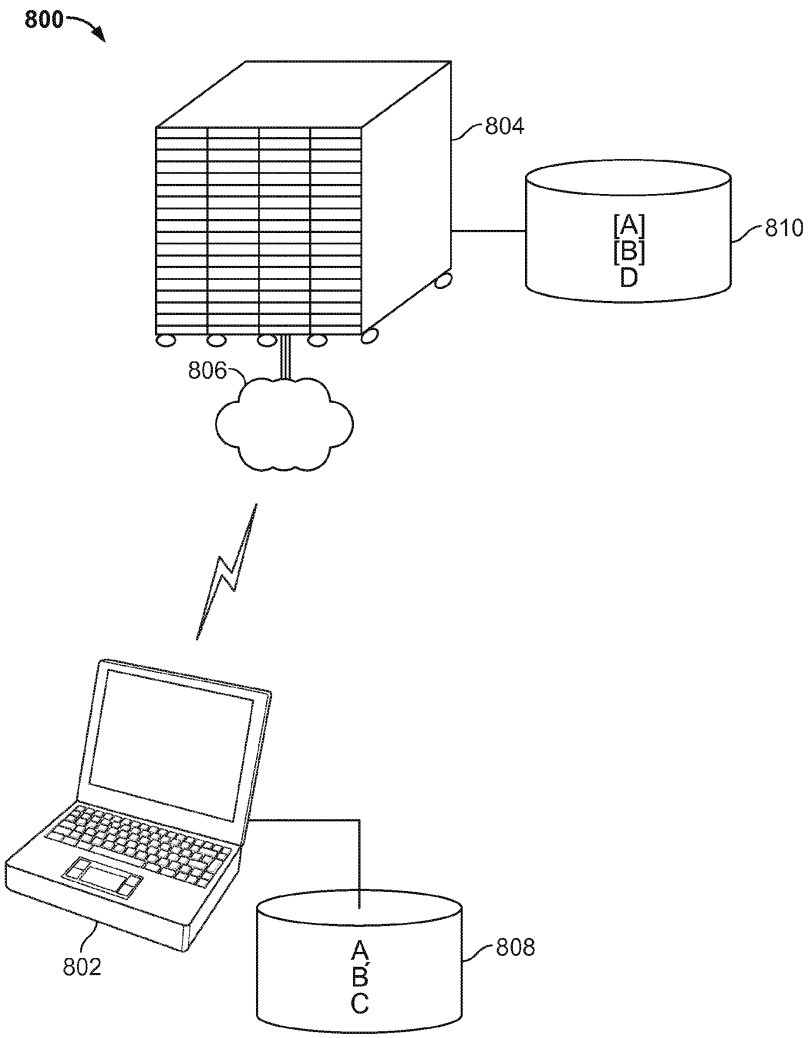
도면6


600
602
606
610
이미지
인터페이스
612
작동
인터페이스
614
이미지
업데이터
616
이미지
재구성기
이미지
인덱스
이미지
조각
사용자
디바이스
지도
618
620
622
604
608

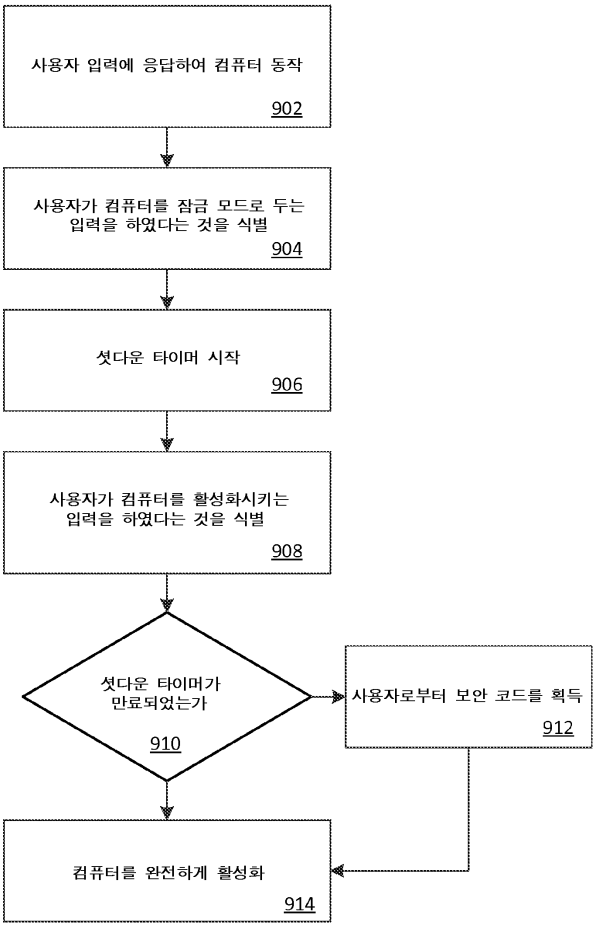
도면7


700
704
708
710
애플리케이션
712
서비스
714
운영체제
716
바이오스
706
Ⓐ
718
네트워크
하드웨어
720
감시 타이머
702

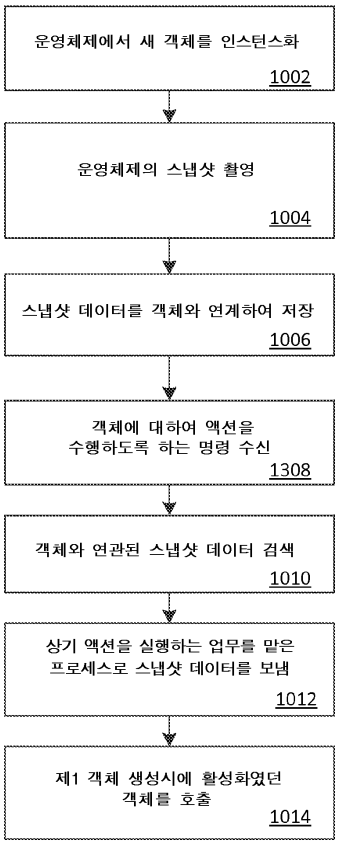
도면8



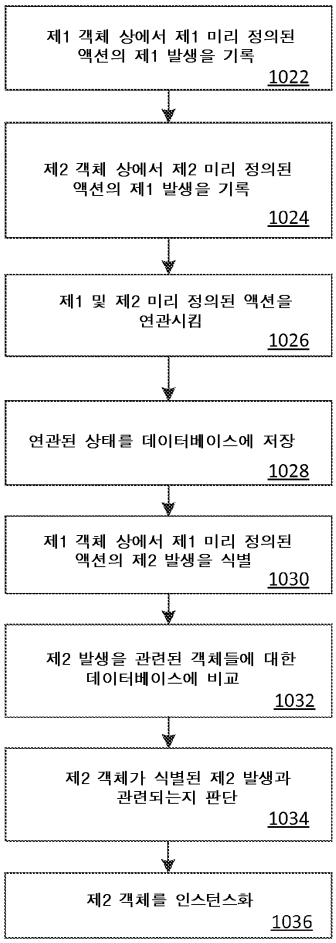
도면9


사용자 입력에 응답하여 컴퓨터 동작
902
사용자가 컴퓨터를 잠금 모드로 두는 입력을 하였다는 것을 식별
904
셧다운 타이머 시작
906
사용자가 컴퓨터를 활성화시키는 입력을 하였다는 것을 식별
908
셧다운 타이머가 만료되었는가
910
사용자로부터 보안 코드를 획득
912
컴퓨터를 완전하게 활성화
914

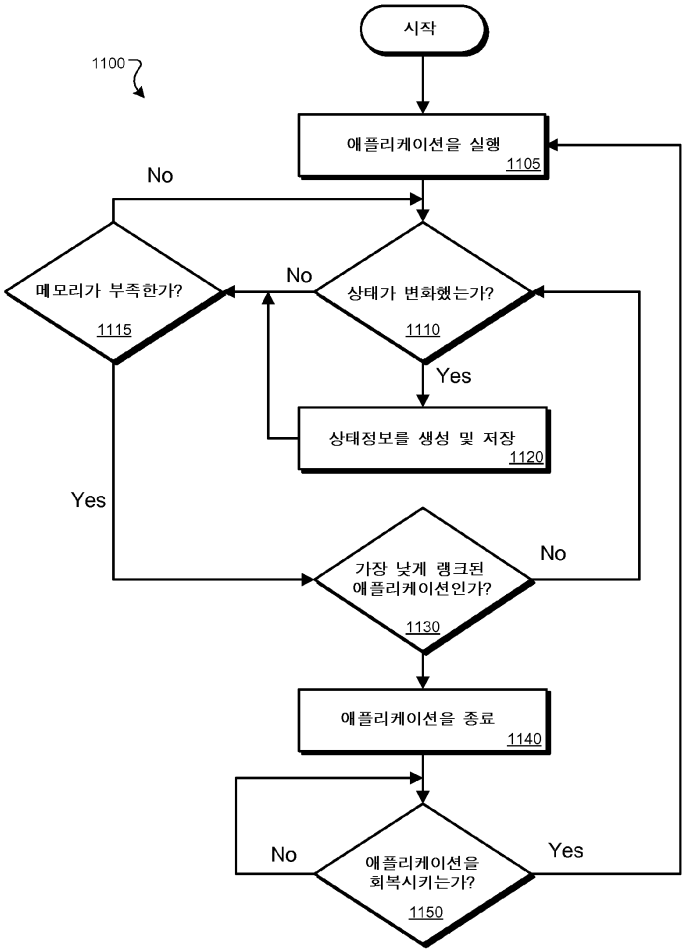
도면10a


운영체제에서 새 객체를 인스턴스화
1002
운영체제의 스냅샷 촬영
1004
스냅샷 데이터를 객체와 연계하여 저장
1006
객체에 대하여 액션을
수행하도록 하는 명령 수신
1308
객체와 연관된 스냅샷 데이터 검색
1010
상기 액션을 실행하는 업무를 맡은
프로세스로 스냅샷 데이터를 보냄
1012
제1 객체 생성시에 활성화였던
객체를 호출
1014

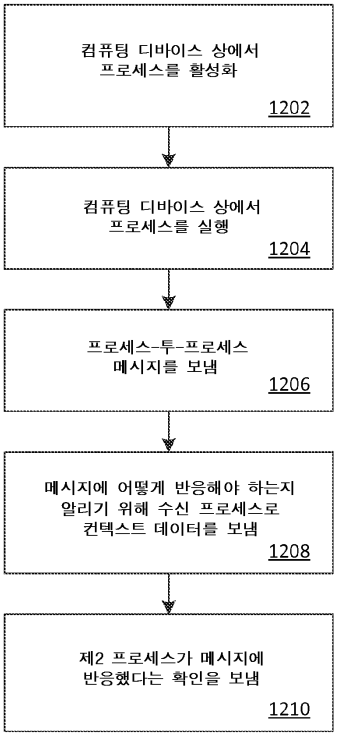
도면10b


제1 객체 상에서 제1 미리 정의된 액션의 제1 발생을 기록
1022
제2 객체 상에서 제2 미리 정의된 액션의 제1 발생을 기록
1024
제1 및 제2 미리 정의된 액션을 연관시킴
1026
연관된 상태를 데이터베이스에 저장
1028
제1 객체 상에서 제1 미리 정의된 액션의 제2 발생을 식별
1030
제2 발생을 관련된 객체들에 대한 데이터베이스에 비교
1032
제2 객체가 식별된 제2 발생과 관련되는지 판단
1034
제2 객체를 인스턴스화
1036

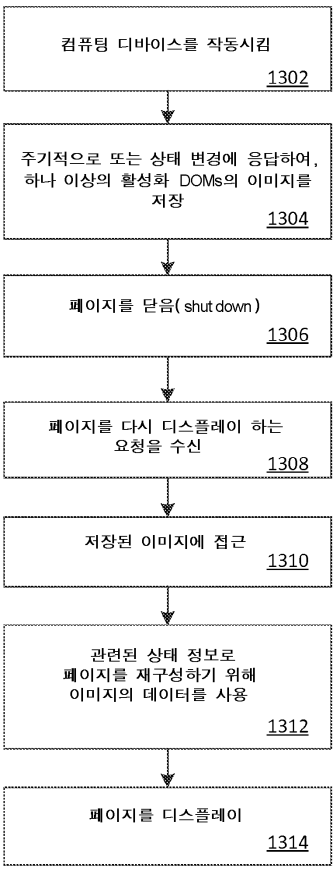
도면11


시작
1100
애플리케이션을 실행
1105
No
메모리가 부족한가?
1115
No
상태가 변화했는가?
1110
Yes
상태정보를 생성 및 저장
1120
Yes
가장 낮게 랭크된
애플리케이션인가?
1130
No
애플리케이션을 종료
1140
No
애플리케이션을
회복시키는가?
1150
Yes

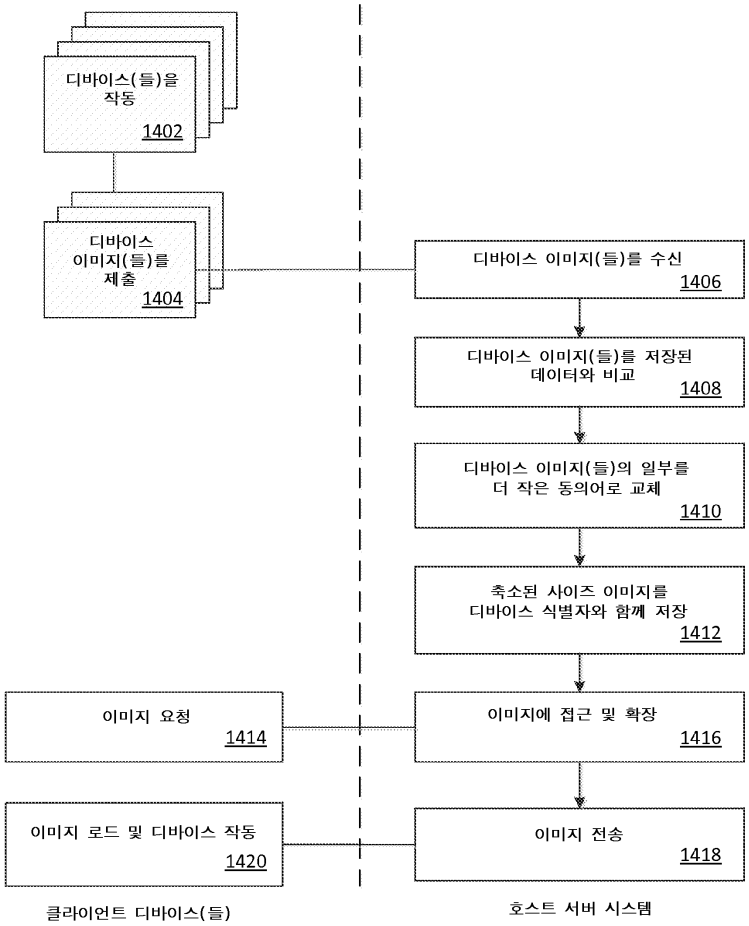
도면12


컴퓨팅 디바이스 상에서
프로세스를 활성화
1202
컴퓨팅 디바이스 상에서
프로세스를 실행
1204
프로세스-투-프로세스
메시지를 보냄
1206
메시지에 어떻게 반응해야 하는지
알리기 위해 수신 프로세스로
컨텍스트 데이터를 보냄
1208
제2 프로세스가 메시지에
반응했다는 확인을 보냄
1210

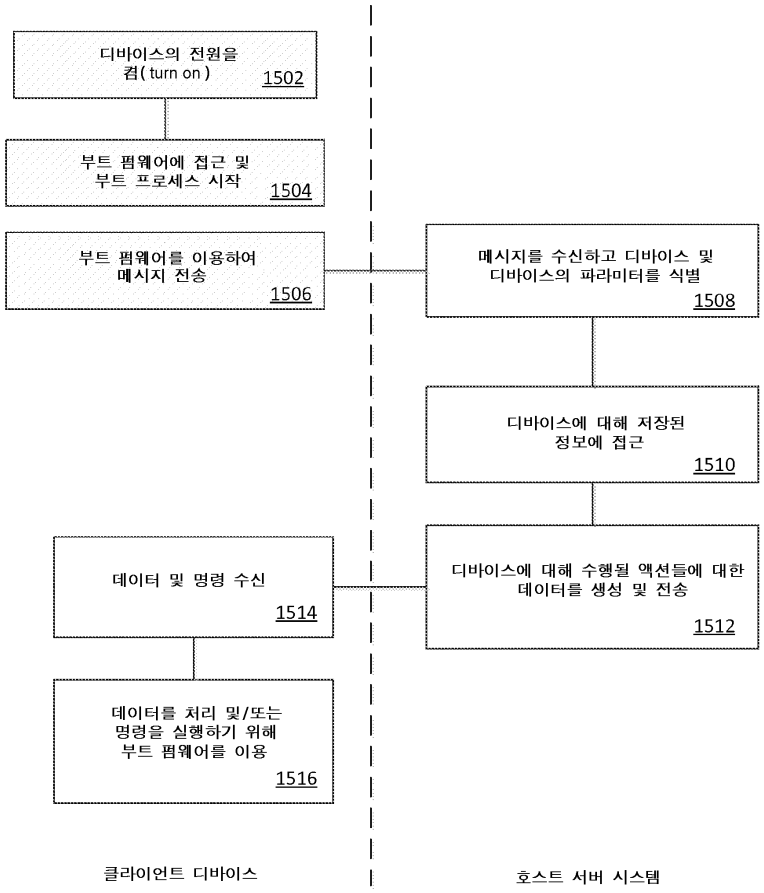
도면13


컴퓨팅 디바이스를 작동시킴
1302
주기적으로 또는 상태 변경에 응답하여,
하나 이상의 활성화 DOMs의 이미지를
저장
1304
페이지를 닫음(shut down)
1306
페이지를 다시 디스플레이 하는
요청을 수신
1308
저장된 이미지에 접근
1310
관련된 상태 정보로
페이지를 재구성하기 위해
이미지의 데이터를 사용
1312
페이지를 디스플레이
1314

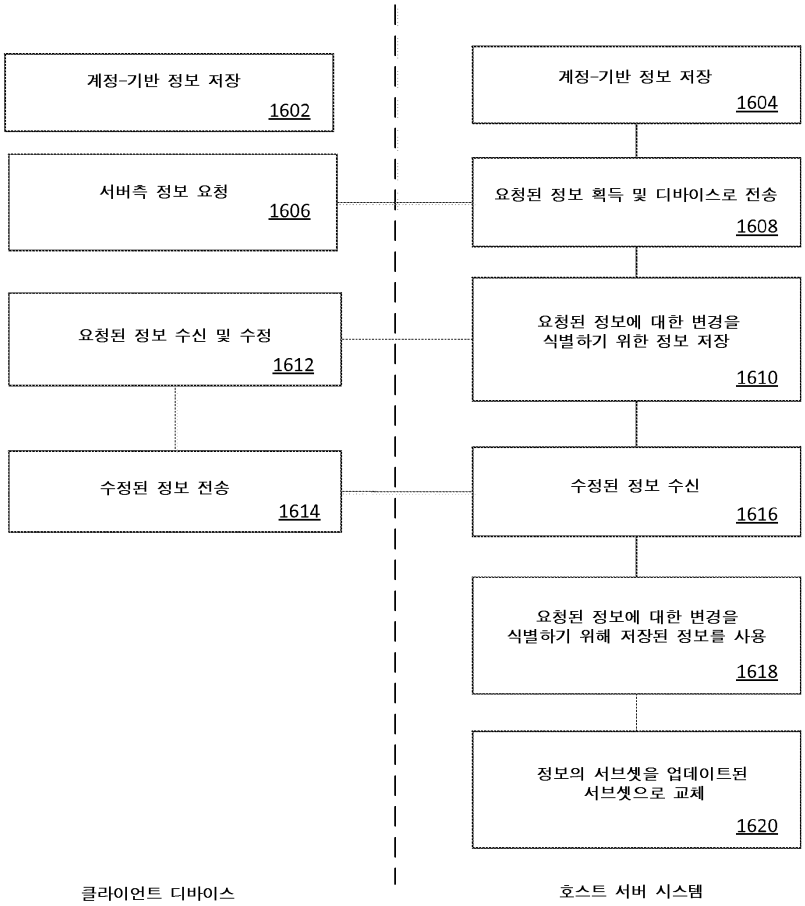
도면14


디바이스(들)을 작동
1402
디바이스 이미지(들)를 제출
1404
디바이스 이미지(들)를 수신
1406
디바이스 이미지(들)를 저장된 데이터와 비교
1408
디바이스 이미지(들)의 일부를 더 작은 동의어로 교체
1410
축소된 사이즈 이미지를 디바이스 식별자와 함께 저장
1412
이미지 요청
1414
이미지에 접근 및 확장
1416
이미지 로드 및 디바이스 작동
1420
이미지 전송
1418
클라이언트 디바이스(들)
호스트 서버 시스템

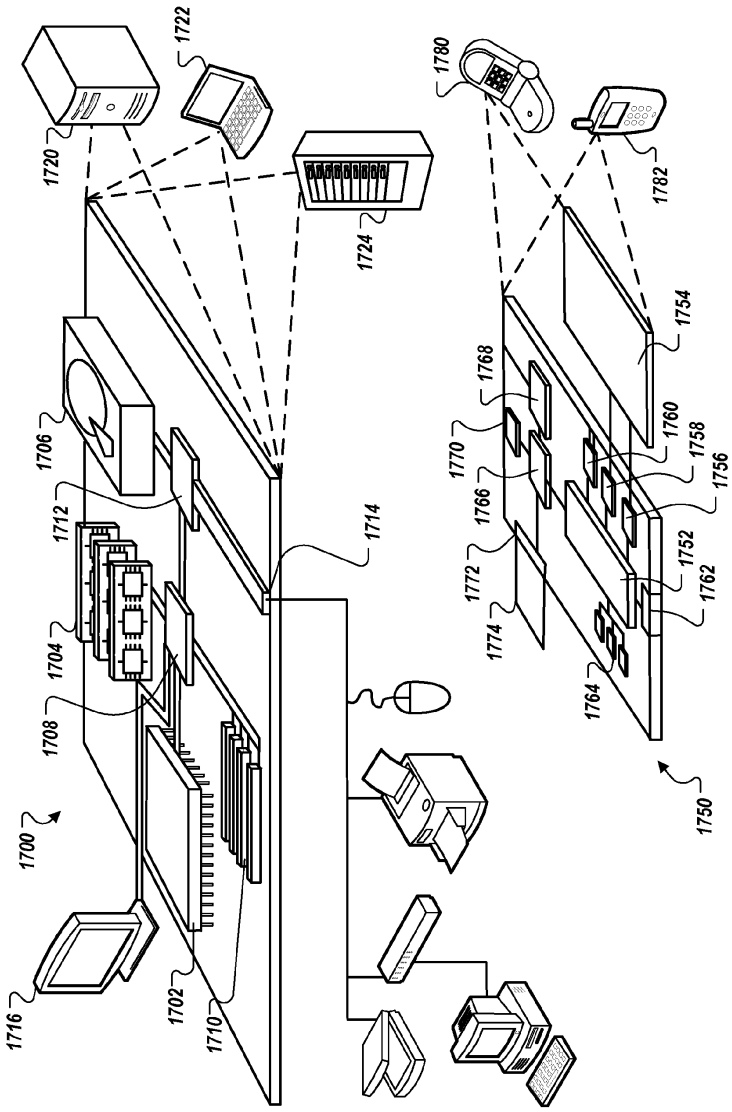
도면15


디바이스의 전원을 켬(turn on) 1502
부트 펌웨어에 접근 및 부트 프로세스 시작 1504
부트 펌웨어를 이용하여 메시지 전송 1506
메시지를 수신하고 디바이스 및 디바이스의 파라미터를 식별 1508
디바이스에 대해 저장된 정보에 접근 1510
디바이스에 대해 수행될 액션들에 대한 데이터를 생성 및 전송 1512
데이터 및 명령 수신 1514
데이터를 처리 및/또는 명령을 실행하기 위해 부트 펌웨어를 이용 1516
클라이언트 디바이스
호스트 서버 시스템

도면16


계정-기반 정보 저장
1602
서버측 정보 요청
1606
요청된 정보 수신 및 수정
1612
수정된 정보 전송
1614
클라이언트 디바이스
계정-기반 정보 저장
1604
요청된 정보 획득 및 디바이스로 전송
1608
요청된 정보에 대한 변경을 식별하기 위한 정보 저장
1610
수정된 정보 수신
1616
요청된 정보에 대한 변경을 식별하기 위해 저장된 정보를 사용
1618
정보의 서브셋을 업데이트된 서브셋으로 교체
1620
호스트 서버 시스템

도면17


1722
1720
1724
1780
1782
1754
1768
1760
1758
1756
1770
1766
1752
1762
1772
1774
1764
1750
1706
1712
1714
1704
1708
1700
1716
1702
1710