(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년09월15일
(11) 등록번호 10-0917064
(24) 등록일자 2009년09월04일

(51) Int. Cl.
H04N 7/173 (2006.01) *H04N 5/44* (2006.01)
(21) 출원번호 10-2007-7012930
(22) 출원일자 2005년11월08일
심사청구일자 2007년09월10일
(85) 번역문제출일자 2007년06월08일
(65) 공개번호 10-2007-0100709
(43) 공개일자 2007년10월11일
(86) 국제출원번호 PCT/US2005/040388
(87) 국제공개번호 WO 2006/052946
국제공개일자 2006년05월18일
(30) 우선권주장
60/626,252 2004년11월08일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
KR1020040049258 A
전체 청구항 수 : 총 25 항

(73) 특허권자
이노패스 소프트웨어, 아이엔시.
미국, 캘리포니아 94089, 서니베일, 캐리비안 드라이브 400
(72) 발명자
구, 진셍
미국, 캘리포니아 94089, 서니베일, 캐리비안 드라이브 400
매너페티, 프렘지스
미국, 캘리포니아 94089, 서니베일, 캐리비안 드라이브 400
(74) 대리인
강명구, 강석용

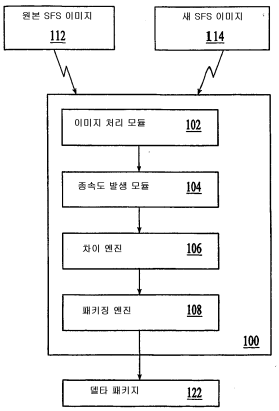
심사관 : 문남두

(54) 정적 파일 시스템 디프런싱 및 업데이트 방법, 장치,시스템

(57) 요 약

정적 파일 시스템의 디프런싱 및 업데이트를 위한 방법 및 시스템이 제공된다. SFS 디프런싱 및 업데이트 방법 및 시스템들은 원본 이미지의 유닛들의 부분-단위 디프런싱(portion level differencing) 및 블록-단위 업데이트(block-level updating)를 포함한다. 한 실시예에서의 디프런싱 및 업데이트는 SFS 이미지들은 블록 정보 및 SFS 이미지 구조에 기초하여 일련의 부분들로 나눈다. 새 SFS 이미지의 각 부분에 대해 델타 파일이 발생되며, 이 델타 파일은 새 SFS 이미지의 부분과, 새 SFS 이미지의 부분에 상응하는 원본 SFS 이미지의 부분 간의 차이 정보를 포함한다. 델타 파일들은 장치로 전달되며, 여기서, 장치의 타겟 SFS 이미지가 델타 파일의 정보를 이용하여 블록 단위로 업데이트된다. 블럭 단위 업데이트는 호스트 장치의 RAM 내 장치 블록에서 새 SFS 이미지의 모든 부분들을 재구성하며, 그후, 재구성된 블록을 호스트 장치의 ROM에 기입(writing)한다.

대 표 도 - 도1



(30) 우선권주장
60/626,292 2004년11월08일 미국(US)
60/626,293 2004년11월08일 미국(US)

특허청구의 범위

청구항 1

정적 파일 시스템(SFS) 이미지들의 디프런싱(differencing)을 위한 장치에 있어서, 상기 장치는,

- 정적 파일 시스템의 이미지들, 즉, 원본 이미지와 새 이미지를 수신하는 수신기,

- 원본 이미지를 다수의 원본 부분들로 분할하고 새 이미지를 다수의 새 부분들로 분할하는 프리프로세서(pre-processor),

- 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬(dependency alignements)을 식별하는 종속도 발생기(dependency generator), 그리고

- 상기 새 부분들 중 한개 이상의 새 부분에 대해 델타 파일을 발생시키는 차이 엔진으로서, 이때, 상기 델타 파일은 상기 한개 이상의 새 부분과, 상기 한개 이상의 새 부분을 좌우하는 원본 부분들 중 한개 이상의 원본 부분 간의 차이를 포함하는 것을 특징으로 하는, 상기 차이 엔진

을 포함하는 것을 특징으로 하는 정적 파일 시스템 이미지들의 디프런싱 장치.

청구항 2

제 1 항에 있어서,

- 업데이트 시퀀스에 따라 새 부분들 중 한개 이상의 새 부분에 대해 델타 파일을 조합하는 패키징 엔진

을 추가로 포함하는 것을 특징으로 하는 정적 파일 시스템 이미지들의 디프런싱 장치.

청구항 3

클라이언트 장치들의 정적 파일 시스템 이미지들을 업데이트하는 방법에 있어서, 상기 방법은,

- 블록 정보 및 이미지 구조 정보를 이용하여 SFS 이미지들의 블록들을 부분들로 나누는 단계로서, 이때, SFS 이미지들은 원본 이미지와 새 이미지를 포함하는 단계,

- 새 이미지의 새 부분에 대해 델타 파일을 발생시킴으로써 부분-단위 디프런싱을 수행하는 단계로서, 이때, 델타 파일은 새 부분과, 원본 이미지의 한개 이상의 대응하는 원본 부분들 간의 차이 정보를 포함하는 단계,

- 델타 파일을 클라이언트 장치에게로 전송하는 단계, 그리고

- 호스트 장치의 RAM의 장치 블록에 새 이미지의 모든 부분들을 재구성함으로써, 그리고 장치 블록을 호스트 장치의 ROM에 기입함으로써, 델타 파일의 정보를 이용하여 클라이언트 장치의 타겟 SFS 이미지를 업데이트하는 단계

를 포함하는 것을 특징으로 하는 클라이언트 장치들의 정적 파일 시스템 이미지들을 업데이트하는 방법.

청구항 4

정적 파일 시스템의 이미지들의 디스프런싱을 위한 방법에 있어서, 상기 방법은,

- 정적 파일 시스템의 이미지들을 수신하는 단계로서, 이때, 상기 이미지들은 원본 이미지와 새 이미지를 포함하는 단계,

- 원본 이미지를 다수의 원본 부분들로 나누고, 새 이미지를 다수의 새 부분들로 나누는 단계,

- 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬을 식별하는 단계, 그리고

- 새 부분들 중 한개 이상의 새 부분에 대해 델타 파일을 발생시키는 단계로서, 이때, 상기 델타 파일은 상기 한개 이상의 새 부분과, 상기 한개 이상의 새 부분을 좌우하는 원본 부분들 중 한개 이상의 원본 부분 간의 차이를 포함하는 단계

를 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디스프런싱을 위한 방법.

청구항 5

정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템에 있어서, 상기 시스템은,

- 정적 파일 시스템의 이미지들, 즉, 원본 이미지와 새 이미지를 수신하는 수신기,

- 상기 수신기에 연결되어, 원본 이미지를 다수의 원본 부분들로 분할하고 새 이미지를 다수의 새 부분들로 분할하는 프리-프로세서,

- 상기 프리-프로세서에 연결되어, 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬을 식별하는 종속도 발생기,

- 상기 종속도 발생기에 연결되어, 한개 이상의 새 부분들을 좌우하는 다수의 원본 부분들 중 한개 이상의 원본 부분들과는 다른 다수의 새 부분들 중 상기 한개 이상의 새 부분에 대해 델타 파일을 발생시키는 차이 엔진으로서, 이때, 상기 델타 파일은 새 부분과, 한개 이상의 원본 부분 간의 인코딩된 차이를 포함하는 것을 특징으로 하는 상기 차이 엔진, 그리고

- 상기 차이 엔진에 연결되어, 델타 파일들을 델타 패키지로 조합하는 패키징 엔진

을 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 6

제 5 항에 있어서, 상기 시스템은 휴대형 장치에 업데이트 엔진을 추가로 포함하고, 상기 휴대형 장치는 한개 이상의 커플링을 통해 델타 패키지를 수신하며, 이때, 상기 업데이트 엔진은,

- 휴대형 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하고,

- 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하며,

- 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는

것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 7

제 6 항에 있어서, 상기 업데이트 엔진은 상기 델타 패키지를 수신하여, 델타 패키지의 한개 이상의 델타 파일의 콘텐트들의 일체성을 확인하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 8

제 6 항에 있어서, 상기 업데이트 엔진은 휴대형 장치의 제 1 메모리 영역에 상기 한개 이상의 새 부분을 재구성하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 9

제 8 항에 있어서, 상기 제 1 메모리 영역이 RAM 내에 있는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 10

제 8 항에 있어서, 상기 업데이트 엔진은, 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 델타 파일들을 계속적으로 식별하고, 식별된델타 파일들에 대응하는 새 부분들을 계속적으로 재구성하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 11

제 10 항에 있어서, 상기 업데이트 엔진은 델타 패키지의 모든 델타 파일들이 휴대형 장치에 호스팅된 원본 부분들에 적용되었음을 결정하고, 이러한 결정에 따라, 재구성된 새 부분들을 휴대형 장치의 제 2 메모리 영역에 기입하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 12

제 11 항에 있어서, 상기 업데이트 엔진은 재구성된 새 부분들의 각 블럭을 제 2 메모리 영역에 기입하는 것을

특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 13

제 11 항에 있어서, 상기 제 2 메모리 영역이 ROM 내에 위치하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 시스템.

청구항 14

정적 파일 시스템의 이미지들의 디프런싱을 위한 방법에 있어서, 상기 방법은,

- 정적 파일 시스템의 이미지들을 수신하는 단계로서, 이때, 상기 이미지들은 원본 이미지와 새 이미지를 포함하고, 상기 이미지들은 다수의 블록들을 포함하는 단계,

- 다수의 블록들의 정보를 이용하여 이미지들을 분할하고, 이에 따라 이미지들을 다수의 부분들로 분할하는 단계,

- 원본 이미지와 새 이미지의 다수의 부분들 간의 차이를 결정함으로서 이미지들의 콘텐트 간의 차이를 결정하는 단계,

- 한개 이상의 부분에 대한 차이를 포함하는 델타 파일을 발생시키는 단계

를 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 15

제 14 항에 있어서,

- 원본 이미지를 호스팅하는 휴대형 무선 장치에 델타 파일을 전송하는 단계

를 추가로 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 16

제 14 항에 있어서,

- 한개 이상의 커플링을 통해 휴대용 장치에서 델타 파일을 수신하는 단계,

- 상기 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계,

- 수신한 델타 파일에 대응하는 의존적 원본 부분들 중 한개 이상의 원본 부분들을 식별하는 단계,

- 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계

를 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 17

제 14 항에 있어서, 다수의 델타 파일들이 한개의 델타 패키지로 조합되는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 18

제 17 항에 있어서,

- 원본 이미지를 호스팅하는 휴대용 무선 장치에 델타 패키지를 전송하는 단계

를 추가로 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 19

제 18 항에 있어서,

- 한개 이상의 커플링을 통해 휴대용 장치에서 델타 패키지를 수신하는 단계,

- 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계,

- 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하는 단계,

- 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계

를 추가로 포함하는 것을 특징으로 하는 정적 파일 시스템의 이미지들의 디프런싱을 위한 방법.

청구항 20

- 정적 파일 시스템의 이미지들을 수신하는 단계로서, 이때, 상기 이미지들은 원본 이미지와 새 이미지를 포함하고, 상기 이미지들은 다수의 블록들을 포함하는 단계,

- 다수의 블록들의 정보를 이용하여 이미지들을 분할하고, 이에 따라 이미지들을 다수의 부분들로 분할하는 단계,

- 원본 이미지와 새 이미지의 다수의 부분들 간의 차이를 결정함으로서 이미지들의 콘텐트 간의 차이를 결정하는 단계,

- 한개 이상의 부분에 대한 차이를 포함하는 델타 파일을 발생시키는 단계

를 포함하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 21

제 20 항에 있어서,

- 원본 이미지를 호스팅하는 휴대형 무선 장치에 델타 파일을 전송하는 단계

를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 22

제 20 항에 있어서,

- 한개 이상의 커플링을 통해 휴대용 장치에서 델타 파일을 수신하는 단계,

- 상기 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계,

- 수신한 델타 파일에 대응하는 의존적 원본 부분들 중 한개 이상의 원본 부분들을 식별하는 단계,

- 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계

를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 23

제 20 항에 있어서, 다수의 델타 파일들이 한개의 델타 패키지로 조합되는 것을 특징으로 하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 24

제 23 항에 있어서,

- 원본 이미지를 호스팅하는 휴대용 무선 장치에 델타 패키지를 전송하는 단계

를 추가로 포함하는 것을 특징으로 하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

청구항 25

제 24 항에 있어서,

- 한개 이상의 커플링을 통해 휴대용 장치에서 델타 패키지를 수신하는 단계,

- 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계,

- 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하는 단계,

- 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계

를 추가로 포함하는 것을 특징으로 하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

명 세 서

기 술 분 야

<1> 본 출원은 2004년 11월 8일에 일제히 출원된 미국특허출원 US 60/626,252 호, US 60/626,292호 그리고 US 60/626,293 호에 기초한 출원이다. 또한 본 출원은 미국특허출원 11월 8일에 일제히 출원된 두건의 미국특허출원 "Updating Compressed Read-Only Memory File System (CRAMFS) Images" 및 "Reorganizing Images in Static File System Differencing and Updating"(출원번호 미정)에 관련된 발명이다.

배 경 기 술

<2> 본 공개 실시예들은 차이 파일을 이용하여 정적 파일 시스템 이미지들을 업데이트하는 데 관련된다.

<3> 소정의 기능을 구현하기 위해 프로세서, 마이크로프로세서, 처리 유닛 등에서 구동되는 소프트웨어는 시간에 따라 변화하거나 복잡도가 증가하는 경향이 있다. 이러한 변화들은 소프트웨어 파일의 그나 오류를 교정해야할 필요성때문에 나타나는 것이다. 이에 따라 새 기술을 추가하거나 기술 진화에 적응할 수 있다. 특히, 무선 이동 장치 등과 같은 이동형 처리 장치에 호스팅된 소프트웨어에는 수많은 소프트웨어 버그들이 포함되어 있는 경우가 많으며, 이들은 물론 교정 대상이다. 소프트웨어는 호스트 장치의 동작에 관련된 한개 이상의 컴퓨터 프로그램, 알고리즘, 파일, 코드 등을 포함한다. 소프트웨어는 모듈이나 컴포넌트라 불리는 소형 유닛들로 분리될 수 있다.

<4> 이동 처리 장치 등과 같은 휴대형 프로세서-기반 장치들은 실시간 동작 시스템(RTOS)을 포함하는 것이 일반적인데, 이 시스템에서는 장치의 모든 소프트웨어 컴포넌트들이 단일 대형 실행 이미지로 링크된다. 더우기, 최근에 파일 시스템 서포트가 대중화되고 있다. 왜냐하면, 이러한 무선 이동 장치에 필요한 기능이 점차 증가하고 있고 컴팩트한 기억 공간이 가능하기 때문이다. 추가적으로, 단일 대형 이미지는 저속 통신 링크(가령, 라디오, 적외선, 또는 시리얼 링크)을 이용하여 장치에 프리로딩(preloading)되거나 임베딩(embedding)될 필요가 있다.

<5> 이동 처리 장치의 소프트웨어를 업데이트함에 있어서의 장애물 중에는 업데이트된 파일을 장치에 전달함에 있어 요구되는 시간, 대역폭, 비용 등이 있다. 업데이트된 파일을 수신하였을 때, 새 파일을 업데이트함에 있어 사용할 수 있는, 장치의 제한된 리소스들도 물론 장애물에 포함된다. 한 예로서, 정적 파일 시스템 이미지들이 호스트 장치의 리소스 제한으로 인해 제자리에서(가령, ROM에서) 업데이트되어야 한다. 가령, RAM, ROM에 관련된 제한사항, 이미지 다운로드 시간, 전력 소모 등등이 그 요인이 될 수 있다. 결과적으로, 리소스-제한 장치들의 소프트웨어를 업데이트함에 있어 호스트 장치에 효율적으로 전달할 수 있으면서 이용이 용이한 차이 파일(difference files)들을 발생시키는 것이 필요하다.

<6> 본 명세서에서 언급되는 특허공보 등의 내용은 본원에서 기재한 것처럼 간주되어야 하며, 본 내용에 완전히 포함되는 것으로 간주되어야 한다.

실 시 예

<14> 정적 파일 시스템(SFS) 디프런싱 및 업데이트를 위한 방법 및 시스템들이 공개된다. SFS 디프런싱 및 업데이트 방법 및 시스템들은 아래 설명되는 바와 같이 원본 이미지의 업데이트 유닛들의 부분-단위 디프런싱(portion level differencing) 및 블록-단위 업데이트(block-level updating)를 포함한다. 한 실시예에서의 디프런싱 및 업데이트는 SFS 이미지들은 블록 정보 및 SFS 이미지 구조에 기초하여 일련의 부분들로 나눈다. 새 SFS 이미지의 각 부분에 대해 델타 파일이 발생되며, 이 델타 파일은 새 SFS 이미지의 부분과, 새 SFS 이미지의 부분에 상응하는 원본 SFS 이미지의 부분 간의 차이 정보를 포함한다(새 부분 한개는 한개보다 많은 원본 부분에 좌우될

수 있고, 추가적으로, 원본 부분이 새 부분과 동일 위치에 있지 않을 수도 있다). 델타 파일들은 장치로 전달되어 장치의 이미지를 새 SFS 이미지로 업데이트하는 데 사용된다. 장치의 타겟 SFS 이미지는 델타 파일들의 정보를 이용하여 블록 단위로 업데이트된다. 한 실시예에 따른 블록 단위 업데이트는 호스트 장치의 RAM 내 장치 블록에서 새 SFS 이미지의 모든 부분들을 재구성하며, 그후, 재구성된 블록을 호스트 장치의 ROM에 기입(writing)한다.

<15> 읽기 전용 파일 시스템이라 불리는 정적 파일 시스템(SFS)은 실시간으로 수정될 수 없는 파일 시스템이다. SFS의 예로는 심비안(Symbian) Z 드라이브(ROM 드라이브라고도 불림), 리눅스 압축 ROM 파일 시스템(CRAMFS), 암호화된 파일 시스템, 그리고, 운영 체제 실행, 내장형 애플리케이션, 멀티미디어 정보, 디폴트 언어 파일, 등등을 저장할 수 있는 파일 시스템 등등이 있다.

<16> 아래 기술되는 실시예에서의 SFS 디프런싱 및 업데이트는 정적 파일 시스템의 이미지들을 수신한다. 원본 이미지와 새 이미지를 포함하는 이미지들은, 각기 다수의 블록들을 포함한다. 가령, 수퍼 블록, 데이터 블록 등을 포함한다. SFS 디프런싱은 블록들의 정보를 이용하여 이미지들을 나누어, 이미지들을 다수개의 부분들로 분할한다. 원본 이미지와 새 이미지의 부분들 간의 차이를 결정함으로서 이미지들의 콘텐트간 차이가 결정된다. 이때, 새 이미지의 각 부분에 대해 차이들이 발생된다. 이 차이들은 부분들 간의 바이트-단위 차이를 포함한다. 그러나 이에 제한되지는 않는다. 새 이미지의 각 부분에 대한 차이들을 포함하는 델타 파일이 발생된다.

<17> 실시예에 따른 SFS 디프런싱 및 업데이트에서의 업데이트는 휴대형 장치 상에 호스팅된 이미지의 SFS가 휴대형 장치 상에서 제위치에서 업데이트된다. 이러한 업데이트는 한개 이상의 커플링을 통해 휴대형 장치에서 델타 파일을 수신한다. 휴대형 장치 상에 호스팅된 원본 부분들 중 종속적 부분들이 조합되고, 수신한 델타 파일에 대응하는 부분들이 식별된다. 이때, ROM 내 새 부분 위치가 해당 델타 파일에 관련된 SFS 델타 패키지로 인코딩된다. 이러한 업데이트는 식별된 델타 파일에 대응하는 휴대형 장치 상의 한개 이상의 새 부분을 재구성한다. 새 이미지의 재구성된 새 부분들은 휴대형 장치의 ROM에 기입된다.

<18> 다음의 설명에서, SFS 디프런싱 및 업데이트의 이해를 돕기 위해 다양한 세부사례들이 제시된다. 그러나, 이들은 일례에 지날 뿐이며, SFS 디프런싱 및 업데이트가 수많은 특정 세부사항들 중 한가지나 몇가지 없이 구현될 수 있음을 지각하여야 할 것이다.

<19> 도 1은 한 실시예에 따른 SFS 디프런싱 및 업데이트의 블록도표에 해당한다. SFS 디프런싱 시스템은 이미지 처리 모듈(102), 종속도 발생 모듈(104), 차이 엔진/모듈(106), 패키징 엔진/모듈(108) 등을 포함한다. 종속도 발생 모듈(104)은 이미지 처리 모듈(102)에 연결될 수 있다. 한 실시예의 차이 엔진(106)은 종속도 발생 모듈(104)에 연결된다. 한 실시예의 패키징 엔진(108)은 차이 엔진(106)에 연결된다.

<20> 한 실시예의 SFS 디프런싱 시스템(100)은 호스트 컴퓨터 시스템(도시되지 않음)의 컴포넌트들 사이에 연결된다. 이때, 이 컴포넌트들은 프로세서, 컨트롤러, 메모리 소자, 버스 등등 중 한가지 이상을 포함할 수 있다. SFS 디프런싱 시스템(100)의 컴포넌트나 모듈(102-108) 중 한개 이상은 한개 이상의 알고리즘, 프로그램, 또는 루틴의 제어 하에서 동작한다. 호스트 컴퓨터 시스템 프로세서는 프로그램 제어 하에서 호스트 컴퓨터 시스템의 컴포넌트들과, SFS 디프런싱 시스템의 컴포넌트(102-108)들 간을 연결한다. 이미지 처리 모듈(102), 종속도 발생 모듈(104), 차이 엔진(106), 그리고 패키징 엔진(108)이 개별적인 블록으로 도시되지만, 이 블록(102-108)들 중 일부 또는 전부가 단일 칩에 모놀리식 방식으로 집적될 수 있고, 호스트 시스템의 컴포넌트나 칩들 사이에 분포될 수 있으며, 또는, 프로그램이나 알고리즘들의 한가지 또는 여러가지 조합에 의해 제공될 수 있다. 프로그램이나 알고리즘은 소프트웨어 알고리즘, 펌웨어, 또는 하드웨어적으로 구현될 수 있고, 이들의 조합에 의해 구현될 수도 있다.

<21> "모듈(module)"이란 용어는 논리적으로 분리가능한 프로그램의 일부분을 말하는 것으로서, 소프트웨어, 알고리즘, 프로그램, 컴포넌트, 그리고 유닛을 포함하는 한가지 이상의 용어에 혼용하여 사용될 수 있다. "컴포넌트"란 용어는 시스템을 구성하는 일부분들 중 한가지 이상을 의미하는 것으로서, 한개의 컴포넌트는 소프트웨어나 하드웨어일 수 있고, 타컴포넌트들로 분할될 수도 있다. "모듈", "컴포넌트", 그리고 "유닛"은 내용에 따라 여러가지 방식으로 서로의 서브요소로 사용되거나 혼용하여 사용될 수 있다. "프로세서"라는 용어는 한개 이상의 중앙 처리 유닛(CPU), 디지털 신호 프로세서(DSP), 전용 집적 회로(ASIC) 등등과 같은 임의의 로직 처리 유닛을 의미한다.

<22> 동작 시에, SFS 디프런싱 시스템(100)은 한개 이상의 원본 SFS 이미지(112)와 한개 이상의 새 SFS 이미지(114)를 수신하여, 아래 설명되는 바와 같이, 한개 이상의 델타 파일들을 발생시키도록 부분-단위(portion-level) 디

프런싱을 실행한다. 델타 파일들은 한개의 데이터 패키지(122)로 조합되어, (클라이언트 장치라고도 불리는) 휴대형 또는 모바일 기기로 전달될 수 있다. 이 차이들은 비교되는 이미지들의 블록들의 한개 이상의 부분들 사이에서 바이트 단위 차이를 포함한다. 그러나 이에 제한되지는 않는다. SFS 디프런싱 시스템(100)은 프로세서 기반 시스템이나 컴퓨터 시스템에서 델타 파일을 발생시키며, 또는 프로세서 기반 시스템이나 컴퓨터 시스템 하에서 동작하는 델타 파일을 발생시킨다. SFS 디프런싱 시스템이 구동되는 컴퓨터 시스템은 당 분야에 잘 알려진 바와 같이, 함께 동작하는 연산 컴포넌트들 및 장치들을 임의의 갯수로 포함할 수 있다. 컴퓨터 시스템은 더 큰 컴퓨터 시스템이나 네트워크의 한 컴포넌트이거나 한개의 서브시스템일 수 있다.

<23> 델타 파일의 콘텐트들은 원본 이미지(112)와 새 이미지(114) 간의 차이를 효율적으로 표현한다. 델타 파일은 2005년 8월 2일자 미국특허 6,925,467 호에 기술된 바와 같이, 관련 파일의 새 버전(또는 현 버전)과 관련 파일의 과거 버전 간의 차이를 표현하는 대체 또는 삽입 동작들의 실제 데이터와 함께, 메타 데이터를 포함할 수 있다. SFS 디프런싱 시스템(100)은 최소수의 바이트와 지정 포맷이나 프로토콜을 이용하여 델타 패키지(122)의 델타 파일에서 원본 이미지(112)와 새 이미지(114) 간의 차이를 제공한다. 이에 따라 공간적으로 최적화된 델타 파일을 제공한다.

<24> SFS 디프런싱 시스템(100)은 부분-단위 디프런싱을 수행하며, 도 2는 한 실시예에 따른 SFS 디프런싱(200)의 순서도이다. SFS 디프런싱은 앞서 설명한, 또는 그외 타부분에서 설명되는 SFS 디프런싱 시스템(100)을 이용하여 수행될 수 있다. SFS 디프런싱(200)은 정적 파일 시스템의 이미지들을 수신한다(202). 이미지들은 한개의 원본 이미지와 한개의 새 이미지를 포함한다. 그러나 이에 제한되지는 않는다. 각각의 이미지는 다수의 블록들을 포함한다. 가령, 수퍼 블록, 데이터 블록 등을 포함한다. 이 블록들은 지정 크기(가령, 64KB, 128KB, 등)를 가지지만, 이에 제한되지는 않는다. 수신한 이미지들의 블록들은 가령 블록들의 정보를 이용하여 다수의 부분(portions)들로 나누어진다. SFS 디프런싱(200)은 원본 이미지와 새 이미지 각각의 부분들 간의 차이를 결정함으로서 이미지들 콘텐트 간의 차이를 결정한다(206). 새 이미지의 각 부분과, 원본 이미지의 한개 이상의 정렬된 부분들 간의 콘텐트나 데이터의 차이 정보를 포함하는 델타 파일이 발생된다(208).

<25> 또다른 예로서, 도 3은 한 실시예에 따른 SFS 디프런싱(300)의 순서도이다. SFS 디프런싱(300)은 도 2를 참고하여 설명한 바와 같이 정적 파일 시스템(SFS)의 이미지들을 수신한다(302). 수신한 원본 이미지는 한개 이상의 원본 부분들로 나누어지며(304), 수신한 새 이미지는 한개 이상의 새 부분들로 나누어진다(304). SFS 디프런싱은 원본 부분과 새 부분 간의 종속도 정렬(dependency alignment)을 식별한다(306). 새 부분들 중 한개 이상의 부분에 대해 델타 파일이 발생된다(308). 델타 파일은 새 부분과, 한개 이상의 원본 부분들 간의 차이를 포함하지만 이에 제한되지는 않는다. 이때, 새 부분은 원본 부분에 의존한다(종속된다). 대안의 실시예들의 델타 파일들은 한개 이상의 새 부분과, 새 부분을 좌우하는 한개 이상의 원본 부분 간의 차이를 포함할 수 있다.

<26> 도 1을 참고하여 설명한 바와 같이 이미지 처리 모듈(102)은 원본 SFS 이미지(112)와 새 SFS 이미지(114) 각각을 수신한다. 한 실시예의 이미지 처리 모듈(102)은 특정 이미지 영역을 분석하고 이미지들의 정보를 추출함으로서, 수신한 이미지들의 부분-단위 디프런싱을 시작한다. 이미지들에 관해 추출되는 정보는 디프런싱 및 업데이트 동작에 사용된다. 이러한 분석 과정에서, 정적 파일 시스템 구조 및 내부 포맷의 정보를 디코딩하여, SFS 이미지 디프런싱 및 업데이트를 수행함에 있어 사용되는 관련 정보를 획득한다. 분석으로부터의 결과는 (가령, 수퍼 블록, 데이터 블록 등등의) 이미지 블록들의 위치 및 크기, 사용되는 압축 라이브러리(압축될 경우), 이미지 해역에 사용되는 암호화 종류(암호화될 경우), 등등을 포함할 수 있다.

<27> 타겟 장치 블록 정보에 기초하여, 이미지 처리 모듈(102)은 SFS 이미지를 다수의 부분들로 나눈다. 도 4는 한 실시예에 따라 이미지 처리 모듈(102)에 의해 부분(402)들로 분할한 후 SFS 이미지(400)를 도시한다. 이 부분들(402)은 이미지의 블록(가령, "Block 4")의 부분들을 포함한다. 가령, 한 실시예의 분리에 따르면, Block 2는 부분(402-2a), 부분(402-2b), 부분(402-2c)을 포함하고, Block 4는 부분(402-4a)과 부분(402-4b)을 포함한다. 분리 동작에 따르면, 한개의 블록은 SFS 이미지의 특정 구조나 블록의 데이터 콘텐트에 적절하게 임의의 숫자의 부분들을 포함할 수 있다. 이 부분들은 수퍼 블록/헤더 부분, 제어 메타-데이터 부분, 파일 부분, 등등을 포함할 수 있다. SFS 이미지 처리 모듈(102)은 (이미지가 압축되었을 경우) 압축해제, (이미지가 암호화되었을 경우) 해역, 그리고, 수신한 이미지로부터 이미지 처리 모듈(102)이 데이터를 추출하기 위해 필요한 그외 다른 처리를 수행할 수 있다.

<28> 한 실시예의 이미지 처리 모듈(102)은 SFS 이미지/파일 네임과, SFS 이미지/파일 네임의 한개 이상의 위치 등과 같은 매핑 정보를 포함하는 파일을 또한 출력한다. 이 파일은 힌트 파일로도 불린다.

<29> 종속도 발생 모듈(104)(종속도 분석 모듈이나 종속도 발생기라고도 불림)은 원본 이미지와 새 이미지들 간의 매

핑이나 정렬도(alignment)를 결정하거나 발생시킨다. 이러한 정렬도는 원본 이미지의 부분들에 따라 좌우되는 새 이미지의 부분들에 관한 정보를 포함한다. 한 실시예의 정렬도는 프로세서 기반 포터블 장치에 호스팅된 원본 이미지의 제위치 업데이트 중 업데이트되어야할 새 이미지의 부분들의 순서에 관한 정보를 포함할 수 있다. 종속도 발생 모듈(104)은 정렬을 수행함에 있어 도메인 지식으로 SFS의 특정 구조의 정보를 이용하며, 이러한 정렬은 이미지의 블록 경계부에 대해 상대적으로 결정된다.

<30> 차이 엔진(106)은 새 이미지의 각각의 새 부분의 데이터와, 원본 이미지의 원본 부분의 데이터 간의 부분-단위 차이를 결정하거나 연산한다. 특히, 차이 엔진(106)은 새 이미지의 각 부분을 좌우하는 원본 이미지의 한개 이상의 부분들을 독출하거나 수집하기 위해 종속도 발생 모듈(104)로부터의 정보를 이용한다. 원본 부분과 새 부분 간의 데이터의 식별된 차이들은 델타 파일이라 불리는 파일에 인코딩된다. 새 부분과 원본 부분 간 데이터의 인코딩된 차이에 추가하여, 델타 파일은 새 부분과 원본 부분 간 종속도들의 종속도 정보같은 제어 정보를 포함할 수 있다. 더우기, 델타 파일은 델타 바디의 확인 정보(가령, 체크섬 값, CRC, 등등), 업데이트될 이미지의 원본 부분, 그리고 이에 대응하는 종속도 정보를 포함할 수 있다.

<31> 차이 엔진(106)이 원본 이미지와 새 이미지 간의 차이를 부분 단위로 결정하지만, 대안의 실시예에서의 차이 엔진은 서로 다른 배율(granularity: 가령, 여러개의 부분들, 등)을 이용하여 차이를 결정할 수 있다. 더우기, 상술한 차이 엔진(106)은 새 이미지를 좌우하는 원본 이미지의 부분과는 다른 새 이미지의 각 부분에 대한 델타 파일을 발생시키지만, 대안의 실시예에서는 각부분에 대해 두개 이상의 델타 파일을 포함하거나, 한개의 델타 파일에 다수개의 부분들의 차이 정보를 포함할 수 있다.

<32> 패키징 엔진(108)은 차이 엔진(106)에 의해 발생된 델타 파일들을 수신하여, 이 델타 파일을 델타 패키지나 차이 패키지로 조합한다. 한 실시예의 패키징 엔진(108)은 업데이트 지시나 순서에 따라 이 델타 파일들을 델타 패키지로 조합한다. 그러나 이에 제한되지는 않는다. 패키징 엔진(108)은 델타 패키지를 암호화하는 등의, 델타 파일이나 델타 패키지에 대한 추가적인 동작들을 수행할 수도 있다.

<33> 상술한 SFS 디프런싱 시스템(100)이 이미지 처리 모듈(102), 종속도 발생 모듈(104), 차이 엔진(106), 그리고 패키징 엔진(108)을 개별적인 모듈이나 컴포넌트로 기재하였으나, 본 실시예는 이에 제한되지 않는다. 대안의 실시예들은 한개 이상의 모듈에 모듈(102-108)의 기능들을 포함할 수 있고, 또는, 임의 갯수의 모듈이나 시스템 컴포넌트 중에 모듈(102-108)의 기능을 분포시킬 수 있다.

<34> SFS 디프런싱 및 업데이트는 휴대형 전자 기기같은 장치에 호스팅된 SFS 이미지들이 상술한 바와 같이 장차 상에서 제위치에서 업데이트되는 방식의 업데이트를 포함한다. 이러한 업데이트는 한개 이상의 커플링을 통해 휴대형 장치에서 델타 파일을 수신한다. 휴대형 장치에 호스팅된 원분 부분들의 종속적 부분들이 조합되고, 종속적 원본 부분들 중 한개 이상의 부분이, 수신한 델타 파일에 대응할 경우, 식별된다. 업데이트는 식별된 델타 파일에 대응하는 휴대형 장치 상의 한개 이상의 새 부분을 재구성한다. 재구성되는 새 이미지의 새 부분들은 휴대형 장치의 ROM에 기입된다.

<35> 도 5는 한 실시예에 따른 SFS 디프런싱 및 업데이트 시스템(500)의 블록도표이다. 시스템(500)은 SFS 디프런싱 시스템(100)과 SFS 업데이트 시스템(550)을 포함한다. SFS 디프런싱 시스템(100)은 이미지 처리 모듈(102), 종속도 발생 모듈(104), 차이 엔진(106), 패키지 엔진(108),을 포함한다. 그러나 이에 제한되지는 않는다. 동작시에, SFS 디프런싱 시스템(100)은 한개 이상의 원본 SFS 이미지(112)와 한개 이상의 새 SFS 이미지(114)를 수신하고 부분-단위(portion-level) 디프런싱을 수행하여, 한개 이상의 델타 파일들을 포함하는 델타 패키지(122)를 발생시킨다.

<36> SFS 디프런싱 시스템(100)은 프로세서 기반 시스템 또는 컴퓨터 시스템에서 델타 파일을 발생시키며, 또는 프로세서 기반 시스템이나 컴퓨터 시스템 하에서 동작하는 델타 파일들을 발생시킨다. SFS 디프런싱 시스템(100)이 동작하는 컴퓨터 시스템은 함께 동작하는 연산 컴포넌트들 및 장치들을 임의의 갯수로 포함할 수 있다. 컴퓨터 시스템은 이보다 큰 대형 컴퓨터 시스템이나 네트워크 내의 한개의 서브시스템이나 컴포넌트일 수도 있다.

<37> SFS 업데이트 시스템(550)은 프로세서 기반 장치 상에 호스팅되어 델타 파일을 수신하고, 휴대형 장치에 호스팅된 원본 이미지들에 대한 업데이트를 수행한다. SFS 업데이트 시스템(550)이 구동되는 프로세서 기반 장치나 시스템은 함께 동작하는 연산 컴포넌트들 및 장치들을 임의의 갯수로 포함할 수 있다. 이 컴퓨터 시스템이 대형 컴퓨터 시스템이나 네트워크의 일개 컴포넌트나 서브시스템일 수도 있다. 프로세서 기반 장치나 시스템은 셀룰러폰, PC, 휴대형 연산 장치, 휴대 전화, 휴대용 통신 기기, 가입자 장치나 유닛, PDA 등등과 같은 이동 장치를 포함할 수 있다. 이동 장치는 "이동 통신 장치", "휴대형 통신 장치", "통신 장치" 등으로도 불리는데, 이러한

모든 장치들 및 이에 상응하는 기기들을 포함하며, 무선 방식의 통신 장치에 국한되는 것도 아니다.

<38> SFS 디프런싱 시스템(100)과 SFS 업데이트 시스템(550)은 통신 경로(506)을 통해 통신한다. 통신 경로(506)는 프로세서 기반 장치나 시스템들 간에 파일들을 전달할 수 있는 임의의 매체를 포함한다. 따라서, 통신 경로(506)는 무선 연결, 유선 연결, 하이브리드 무선/유선 연결을 포함한다. 통신 경로(506)는 LAN, MAN, WAN, 전용 네트워크, 인터오피스나 백엔드 네트워크, 인터넷 등등과 같은 네트워크에 대한(를 통한) 연결을 또한 포함한다. 이 통신 경로(506)는 통신 서비스 제공자나 캐리어의 다양한 네트워크 컴포넌트들(도시되지 않음)을 포함할 수 있다. 그러나 이에 제한되지는 않는다. 더우기, 통신 경로(506)는 탈착가능한 고정식 매체, 가령, 플라피 디스크, 하드 디스크 드라이브, CD-ROM 디스크, 전화 선, 버스, 전자 메일 메시지 등등을 포함할 수 있다.

<39> 델타 패키지(122)는 통신 경로(506)를 통해 SFS 업데이트 시스템(550)에 전송된다. SFS 업데이트 시스템(550)은 델타 패키지(122)의 정보를 이용하는 업데이트 모듈(552)을 포함하여, 휴대형 장치 상에 호스팅된 SFS 이미지(112P)의 제위치 업데이트(554)를 수행한다. 업데이트 모듈(552)은 델타 패키지(122)의 콘텐트들을, 휴대형 장치에 호스팅된 원본 SFS 이미지(112P)의 부분들에 적용함으로서 휴대형 장치에 새 이미지(114P)를재구성한다. 재구성된 새 이미지(114P)의 부분들은 휴대형 장치의 ROM에 기입된다. 가령, 호스팅된 원본 이미지(112P) 상에 새 이미지(114P)를 기입하는 방식으로 이루어진다. 이 SFS 업데이트 프로세스를 완료하면, 휴대형 장치 상에 이제 호스팅된 SFS 이미지는 첫번째 SFS 디프런싱 시스템(100)에서 수신한 새 SFS 이미지(114)와 실질적으로 동일하다.

<40> 도 6은 한 실시예에 따른 SFS 업데이트(600)의 순서도이다. SFS 업데이트(600)는 리소스-제한 연산 장치에 사용된다. SFS 업데이트는 한개 이상의 커플링을 통해 휴대형 장치에서 델타 파일을 수신한다(602). 휴대형 장치 상에 호스팅된 SFS 이미지의 의존적 원본 부분들이 조합된다(604). 업데이트(600)는 수신한 델타 파일에 대응하는 의존적 원본 부분들 중 한개 이상의 부분을 식별한다(606). 업데이트(600)는 식별된 델타 파일에 대응하는 휴대형 장치 상의 한개 이상의 새 부분을 재구성한다(608). 재구성된 새 이미지의 새 부분들은 그후 휴대형 장치의 메모리(가령, ROM)에 기입된다. 이는 델타 패키지에서 수신한 모든 델타 파일들의 처리와 동시에 이루어질 수도 있고, 델타 파일들의 처리 이후에 이루어질 수도 있다.

<41> 도 7은 한 실시예에 따라 장치에서 SFS 이미를 제위치에서 업데이트(700)하는 과정의 순서도이다. 업데이트(700)는 도 5를 참고하여 설명한 바와 같이 업데이트 모듈(552)에 의해 수행될 수 있다. 업데이트(700)는 수신한 델타 패키지에 인코딩된 제어 정보를 이용하여, 델타 패키지의 데이터가 훼손되지 않았는 지를 결정하여(702), 델타 패키지 콘텐트의 일체성을 확인한다. 업데이트될 각 SFS 이미지 부분의 일체성이 또한 확인될 수 있다. 델타 패키지의 어떤 데이터가 훼손된 경우, 에러가 보고된다(704).

<42> 델타 페키지의 데이터가 훼손되지 않았다고 결정될 경우(702), 업데이트는 업데이트 시작을 위해 델타 패키지의 시작에 대한 포인터를 설정한다(706). 한 실시예의 델타 패키지에는 부분 업데이트 오더와 장치 블록 업데이트 오더가 인코딩된다. 업데이트의 시작 시에 델타 파일이 존재함이 결정되고(708) 업데이트가 진행된다. 델타 파일이 보안 용도를 위해 암호화될 수 있으며, 이러한 경우에, 델타 파일은 업데이트 프로세스의 시작 이전에 또는 시작과 동시에 해역된다(적절한 키를 이용). 업데이트(700)는 델타 파일 콘텐트의 정보(가령, 제어 정보)를 이용하여 종속도 형성 정보를 디코딩하고(712), (필요시) 원본 이미지의 관련 부분들을 독출하며(712), 원본 이미지의 의존적 부분들을 해역(712)하고, 휴대형 장치에 호스팅된 원본 이미지의 원본 부분들의 의존적 콘텐트를 조합한다(712).

<43> 업데이트(700)는 원본 이미지의 원본 부분의 의존적 콘텐트에 델타 파일의 콘텐트를 적용함으로서(714) 업데이트되는 이미지의 새 부분을 발생시키거나 재구성한다. 재구성된 새 부분은 호스트 장치의 RAM 영역에 기입되고, SFS 이미지에 적절한 대로 암호화된다(714). 새 부분은 원본 부분을 대응하는 부분으로 대체하도록 원본 이미지의 블록 내 특정 위치에 배치된다(714). 새 이미지의 각 블록이 생성되거나 재구성된 후, 상기 특정 블록이 ROM에 기입된다. 대안의 실시예들은 업데이트 프로세스의 다른 지점에서, 재구성된 블록들을 ROM에 기입할 수 있다. 업데이트 프로세스는 그후, 다음 블록에 대한 델타 파일 처리를 진행할 수 있다.

<44> 상술한 업데이트(712, 714)는 델타 패키지의 종점에 도달할 때까지, 그리고 델타 패키지의 모든 델타 파일들이 처리되었을 때가지 계속된다. 이렇게 함에 있어서, 델타 패키지가 원본 이미지의 추가적 부분들을 업데이트하는 데 사용되는 델타 파일들을 추가로 포함하는 지를 결정한다(716). 업데이트될 원본 이미지의 추가적 부분들에 대응하는 처리되지 않은 델타 파일들이 델타 패키지에 포함될 경우, 이러한 처리되지 않은 델타 파일들을 독출하고 처리하기 위해 동작이 재개된다. 델타 패키지의 모든 델타 파일들이 처리되어 원본 이미지에 적용되었다고 결정될 경우(716), 업데이트(700)는 새 SFS 이미지를 휴대형 장치의 ROM에 기입한다. 한 실시예의 업데이트는

원본 SFS 이미지를 새 SFS 이미지로 덮어쓴다. 그러나, 대안의 실시예에서는 ROM의 한개 이상의 다른 영역에, 또는 그외 다른 장치 메모리 영역에 새 SFS 이미지를 기입할 수 있다.

<45> 도 1, 2, 3, 5, 6, 7에서는 프로세스의 동작들이 한개 이상의 프로세서의 제어 하에 있었으나, 여기에 제한되는 것은 아니다. 당 분야의 통상의 지식을 가진 자라면, 본원의 순서도 및 상세한 설명들에 기초하여 한 실시예의 SFS 디프런싱 및 SFS 업데이트를 구현하거나, 소스 코드, 마이크로코드, 프로그램 로직 어레이 등등을 생성할 수 있을 것이다. 이러한 순서도들에 따라 동작하는 알고리즘이나 루틴은 컴퓨터 시스템의 기계 판독형 또는 컴퓨터 판독형 메모리 영역이나 장치에 프로그램 코드로 저장된다. 이러한 메모리 영역이나 장치는 칩(가령, EEPROM 반도체 칩)에 하드와이어 방식이나 프리프로그래밍 방식으로 구성된, 또는, 디스크같은 탈착가능한 매체의 관련 메모리 영역에 관련 프로세서들의 일부분을 형성한다. 그러나 이에 제한되지는 않는다.

<46> 한 실시예의 SFS 디프런싱 및 업데이트는 SFS 이미지의 디프런싱을 위한 장치를 포함한다. 한 실시예의 장치는 SFS 이미지를 수신하는 수신기를 포함하며, 이때, 이미지들은 원본 이미지와 새 이미지를 포함한다. 한 실시예의 장치는 원본 이미지를 수많은 원본 부분들로 분할하여 새 이미지를 수많은 새 부분들로 나누는 프리-프로세서(pre-processor)를 또한 포함한다. 한 실시예의 장치는 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬(depedency alignments)을 식별하는 종속도 발생기를 포함한다. 한 실시예의 장치는 새 부분들 중 한개 이상의 부분에 대해 델타 파일을 발생시키는 차이 엔진을 포함하며, 이때, 델타 파일은 한개 이상의 새 부분과, 상기 한개 이상의 새 부분을 좌우하는 원본 부분들 중 한개 이상의 부분들 간의 차이를 포함한다.

<47> 한 실시예의 장치는 업데이트 시퀀스에 따라 새 부분들 중 한개 이상의 부분에 대해 델타 파일을 조합하는 패키징 엔진을 추가로 포함한다.

<48> 한 실시예의 SFS 디프런싱 및 업데이트는 블록 정보와 이미지 구조 정보에 기초하여 SFS 이미지의 블록들을 부분들로 나누는 단계로 구성되는 방법을 포함하며, 이때, SFS 이미지들은 원본 이미지들과 새 이미지들을 포함한다. 한 실시예의 방법은 새 이미지의 새 부분에 대해 델타 파일을 발생시킴으로서 부분-단위 디프런싱을 수행하며, 이때, 델타 파일은 새 부분과, 원본 이미지의 한개 이상의 대응하는 원본 부분들 간의 차이 정보를 포함한다. 한 실시예의 방법은 호스트 장치의 RAM에 장치 블록의 새 이미지의 모든 부분들을 재구성함으로서, 그리고 장치 블록을 호스트 장치의 ROM에 기입함으로서, 델타 파일의 정보를 이용하여 클라이언트 장치의 타겟 SFS 이미지를 업데이트하는 과정을 포함한다.

<49> 한 실시예의 SFS 디프런싱 및 업데이트는 SFS 이미지들을 수신하는 단계로 구성되는 방법을 포함한다. 이때, 이미지들은 원본 이미지와 새 이미지를 포함한다. 한 실시예의 방법은 원본 이미지를 다수의 원본 부분들로 나누며, 새 이미지를 다수의 새 부분들로 나눈다. 한 실시예의 방법은 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬을 식별하는 과정을 포함한다. 한 실시예의 방법은 새 부분들 중 한개 이상의 부분에 대해 델타 파일을 발생시키는 과정을 포함하며, 이때, 델타 파일은 한개 이상의 세 부분들과, 상기 한개 이상의 새 부분들을 좌우하는 원본 부분들 중 한개 이상의 원본 부분 간의 차이를 포함한다.

<50> 한 실시예의 SFS 디프런싱 및 업데이트는, 정적 파일 시스템의 이미지들, 즉, 원본 이미지와 새 이미지를 수신하는 수신기를 포함하는 시스템을 포함한다. 상기 시스템은 상기 수신기에 연결되어, 원본 이미지를 다수의 원본 부분들로 분할하고 새 이미지를 다수의 새 부분들로 분할하는 프리-프로세서를 포함한다. 상기 시스템은 상기 프리-프로세서에 연결되어, 다수의 원본 부분들과 다수의 새 부분들 간의 종속도 정렬을 식별하는 종속도 발생기를 포함한다. 상기 시스템은 상기 종속도 발생기에 연결되어, 한개 이상의 새 부분들을 좌우하는 다수의 원본 부분들 중 한개 이상의 원본 부분들과는 다른 다수의 새 부분들 중 상기 한개 이상의 새 부분에 대해 델타 파일을 발생시키는 차이 엔진으로서, 이때, 상기 델타 파일은 새 부분과, 한개 이상의 원본 부분 간의 코딩된 차이를 포함하는 것을 특징으로 하는 상기 차이 엔진을 포함한다. 상기 시스템은, 상기 차이 엔진에 연결되어, 델타 파일들을 델타 패키지로 조합하는 패키징 엔진을 포함한다.

<51> 상기 시스템은 휴대형 장치에 업데이트 엔진을 추가로 포함하고, 상기 휴대형 장치는 한개 이상의 커플링을 통해 델타 패키지를 수신하며, 이때, 상기 업데이트 엔진은, 휴대형 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하고, 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하며, 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성한다.

<52> 상기 업데이트 엔진은 상기 델타 패키지를 수신하여, 델타 패키지의 한개 이상의 델타 파일의 콘텐트들의 일체성을 확인한다.

<53> 상기 업데이트 엔진은 휴대형 장치의 제 1 메모리 영역에 상기 한개 이상의 새 부분을 재구성한다.

<54> 상기 제 1 메모리 영역은 RAM 내에 있다.

<55> 상기 업데이트 엔진은, 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 델타 파일들을 계속적으로 식별하고, 식별된 델타 파일들에 대응하는 새 부분들을 계속적으로 재구성한다.

<56> 상기 업데이트 엔진은 델타 패키지의 모든 델타 파일들이 휴대형 장치에 호스팅된 원본 부분들에 적용되었음을 결정하고, 이러한 결정에 따라, 재구성된 새 부분들을 휴대형 장치의 제 2 메모리 영역에 기입한다.

<57> 상기 업데이트 엔진은 재구성된 새 부분들의 각 블럭을 제 2 메모리 영역에 기입한다.

<58> 상기 제 2 메모리 영역이 ROM 내에 위치한다.

<59> 한 실시예의 SFS 디프런싱 및 업데이트는, 정적 파일 시스템의 이미지들을 수신하는 단계로서, 이때, 상기 이미지들은 원본 이미지와 새 이미지를 포함하고, 상기 이미지들은 다수의 블록들을 포함하는 단계를 포함하는 방법을 제시한다. 이 방법은 다수의 블록들의 정보를 이용하여 이미지들을 분할하고, 이에 따라 이미지들을 다수의 부분들로 분할하는 단계를 포함한다. 이 방법은 원본 이미지와 새 이미지의 다수의 부분들 간의 차이를 결정함으로서 이미지들의 콘텐트 간의 차이를 결정하는 단계를 포함한다. 이 방법은 한개 이상의 부분에 대한 차이를 포함하는 델타 파일을 발생시키는 단계를 포함한다.

<60> 한 실시예의 방법은 원본 이미지를 호스팅하는 휴대형 무선 장치에 델타 파일을 전송하는 단계를 포함한다.

<61> 한 실시예의 방법은, 한개 이상의 커플링을 통해 휴대용 장치에서 델타 파일을 수신하는 단계를 포함한다. 한 실시예의 방법은, 상기 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계를 포함한다. 한 실시예의 방법은, 수신한 델타 파일에 대응하는 의존적 원본 부분들 중 한개 이상의 원본 부분들을 식별하는 단계를 포함한다. 한 실시예의 방법은, 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계를 포함한다.

<62> 한 실시예의 방법은 다수의 델타 파일들이 한개의 델타 패키지로 조합되는 것을 특징으로 한다.

<63> 한 실시예의 방법은 원본 이미지를 호스팅하는 휴대용 무선 장치에 델타 패키지를 전송하는 단계를 포함한다.

<64> 한 실시예의 방법은, 한개 이상의 커플링을 통해 휴대용 장치에서 델타 패키지를 수신하는 단계를 포함한다. 한 실시예의 방법은, 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계를 포함한다. 한 실시예의 방법은 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하는 단계를 포함한다. 한 실시예의 방법은 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계를 포함한다.

<65> 한 실시예의 SFS 디프런싱 및 업데이트는, 정적 파일 시스템의 이미지들을 수신하는 단계로서, 이때, 상기 이미지들은 원본 이미지와 새 이미지를 포함하고, 상기 이미지들은 다수의 블록들을 포함하는 단계를 포함하는 이미지 간의 차이를 결정하는 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 매체를 포함한다. 상기 매체는 다수의 블록들의 정보를 이용하여 이미지들을 분할하고, 이에 따라 이미지들을 다수의 부분들로 분할하는 단계를 포함하는 방법을 실행시킬 수 있다. 상기 매체는 원본 이미지와 새 이미지의 다수의 부분들 간의 차이를 결정함으로서 이미지들의 콘텐트 간의 차이를 결정하는 단계를 포함하는 방법을 실행시킬 수 있다. 상기 매체는 한개 이상의 부분에 대한 차이를 포함하는 델타 파일을 발생시키는 단계를 포함하는 방법을 실행시킬 수 있다.

<66> 상기 매체는 원본 이미지를 호스팅하는 휴대형 무선 장치에 델타 파일을 전송하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 한개 이상의 커플링을 통해 휴대용 장치에서 델타 파일을 수신하는 단계를 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 상기 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 수신한 델타 파일에 대응하는 의존적 원본 부분들 중 한개 이상의 원본 부분들을 식별하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다.

<67> 상기 매체는 다수의 델타 파일들이 한개의 델타 패키지로 조합되는 것을 특징으로 한다. 상기 매체는 원본 이미지를 호스팅하는 휴대용 무선 장치에 델타 패키지를 전송하는 단계를 추가로 포함하는 이미지 간의 차이를 결정

하는 방법을 실행시킬 수 있다. 상기 매체는 한개 이상의 커플링을 통해 휴대용 장치에서 델타 패키지를 수신하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 휴대용 장치에 호스팅된 다수의 원본 부분들 중 의존적 원본 부분들을 조합하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 상기 의존적 원본 부분들 중 한개 이상의 원본 부분에 대응하는 델타 패키지의 한개 이상의 델타 파일을 식별하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다. 상기 매체는 식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성하는 단계를 추가로 포함하는 이미지 간의 차이를 결정하는 방법을 실행시킬 수 있다.

<68> SFS 디프런싱 및 업데이트의 태양들은 다양한 회로에 프로그래밍된 기능으로 구현될 수 있다. 이때, 다양한 회로란, 필드 프로그래머블 게이트 어레이(FPGA)같은 프로그래머블 로직 소자(PLD), 프로그래머블 어레이 로직(PAL) 소자, 전기적으로 프로그래밍가능한 로직 및 메모리 소자, 표준형 셀-기반 장치, 그리고 ASIC 등등을 예로 들 수 있다. SFS 디프런싱 및 업데이트의 태양들을 구현하기 위한 그외 다른 가능성으로는, 소프트웨어, 펌웨어, 임베디드 마이크로프로세서, 메모리를 구비한 마이크로컨트롤러 등을 포함한다. 더우기, SFS 디프런싱 및 업데이트의 태양들은 소프트웨어 기반 회로 에뮬레이션을 구비한 마이크로프로세서, 디스크리트 로직(시퀀스형 및 컴비네이션형), 전용 소자(custom devices), 퍼지 (신경) 로직, 퀀텀 소자, 그리고 이러한 소자 종류들의 조합에서 구현될 수 있다. 물론, 아래의 소자 기술들도 다양한 컴포넌트 종류로 제공될 수 있다. 즉, 금속-산화물 반도체 전계 효과 트랜지스터(MOSFET) 기술(가령, CMOS), 쌍극성 기술(가령, 에미터-커플드 로직(ECL)), 폴리머 기술(가령, 실리콘-컨져게이트 폴리머 및 메탈-컨져게이트 폴리머-메탈 구조), 아날로그 및 디지털 혼합형 기술, 등등이 다양한 컴포넌트 종류로 제공될 수 있다.

<69> 본원의 SFS 디프런싱 및 업데이트는 본원에서 언급한 SFS 디프런싱 및 업데이트 시스템에 대해서만이 아니라, 그외 다른 처리 시스템 및 통신 시스템에도 적용될 수 있다.

도면의 간단한 설명

<7> 도 1은 일 실시예에 따른 정적 파일 시스템(SFS) 디프런싱 시스템의 블록도표이다.

<8> 도 2는 일 실시예에 따른 SFS 디프런싱의 순서도이다.

<9> 도 3은 일 실시예에 따른 SFS 디프런싱의 순서도이다.

<10> 도 4는 일 실시예에 따라, 이미지 블록들을 부분부분으로 나눈 후 SFS 이미지의 예를 도시한다.

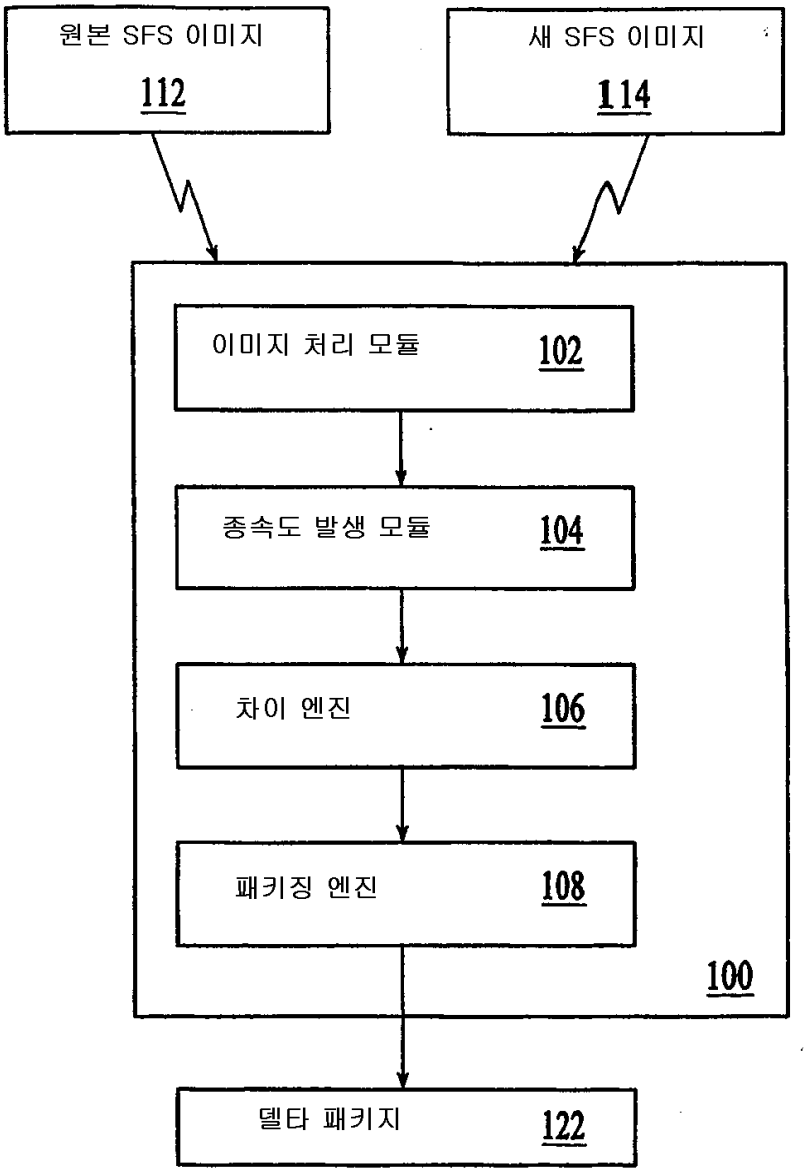
<11> 도 5는 일 실시예에 따른 SFS 디프런싱 및 업데이트 시스템의 블록도표이다.

<12> 도 6은 일 실시예에 따른 SFS 업데이트의 순서도이다.

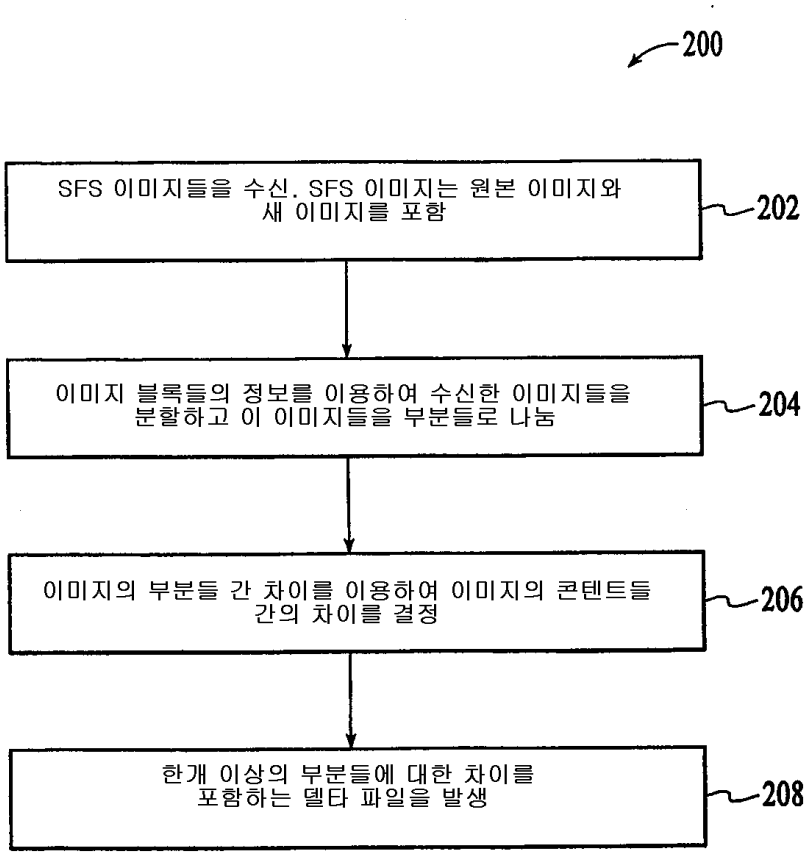
<13> 도 7은 일 실시예에서 장치 내의 SFS 이미지들을 제위치에서 업데이트하는 과정의 순서도이다.

도면

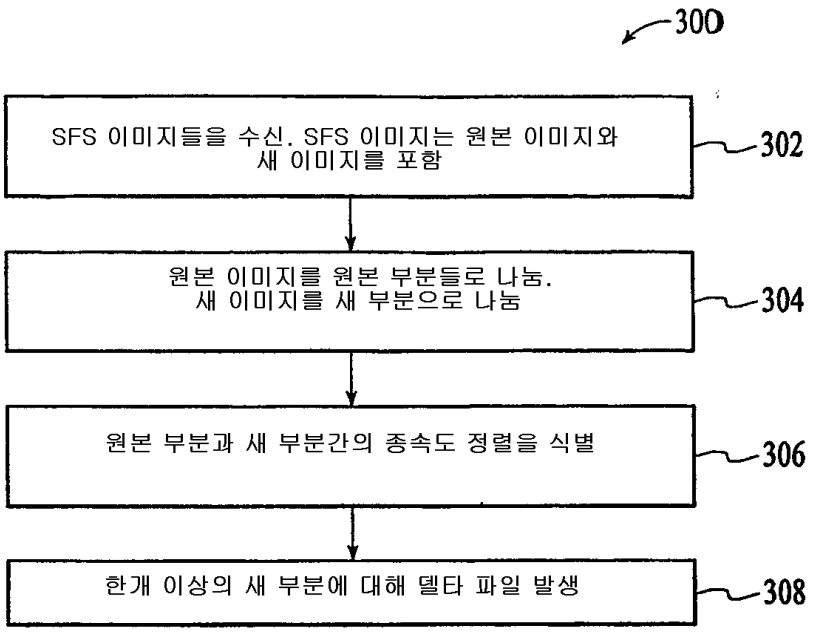
도면1


원본 SFS 이미지
112
새 SFS 이미지
114
이미지 처리 모듈
102
종속도 발생 모듈
104
차이 엔진
106
패키징 엔진
108
100
델타 패키지
122

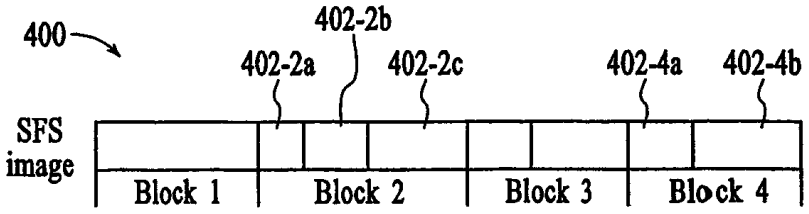
도면2


200
SFS 이미지들을 수신. SFS 이미지는 원본 이미지와 새 이미지를 포함
202
이미지 블록들의 정보를 이용하여 수신한 이미지들을 분할하고 이 이미지들을 부분들로 나눔
204
이미지의 부분들 간 차이를 이용하여 이미지의 콘텐트들 간의 차이를 결정
206
한개 이상의 부분들에 대한 차이를 포함하는 델타 파일을 발생
208

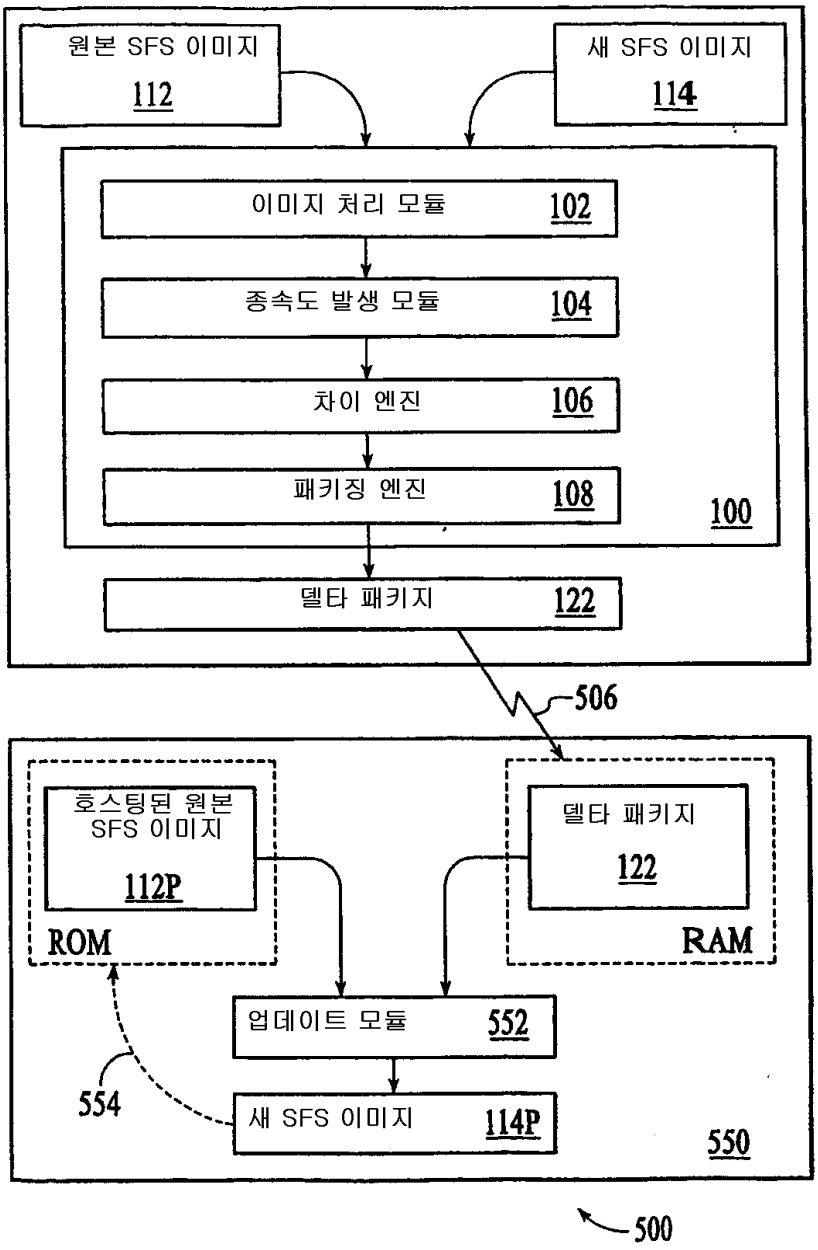
도면3


300
SFS 이미지들을 수신. SFS 이미지는 원본 이미지와 새 이미지를 포함
302
원본 이미지를 원본 부분들로 나눔. 새 이미지를 새 부분으로 나눔
304
원본 부분과 새 부분간의 종속도 정렬을 식별
306
한개 이상의 새 부분에 대해 델타 파일 발생
308

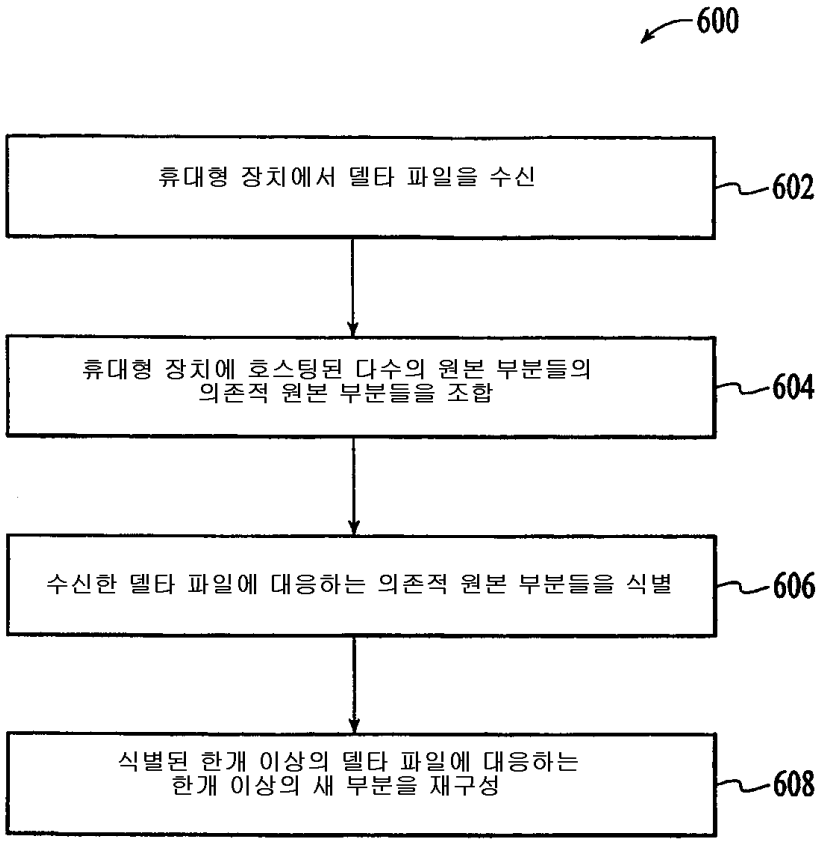
도면4


400
402-2b
402-2a
402-2c
402-4a
402-4b
SFS image
Block 1
Block 2
Block 3
Block 4

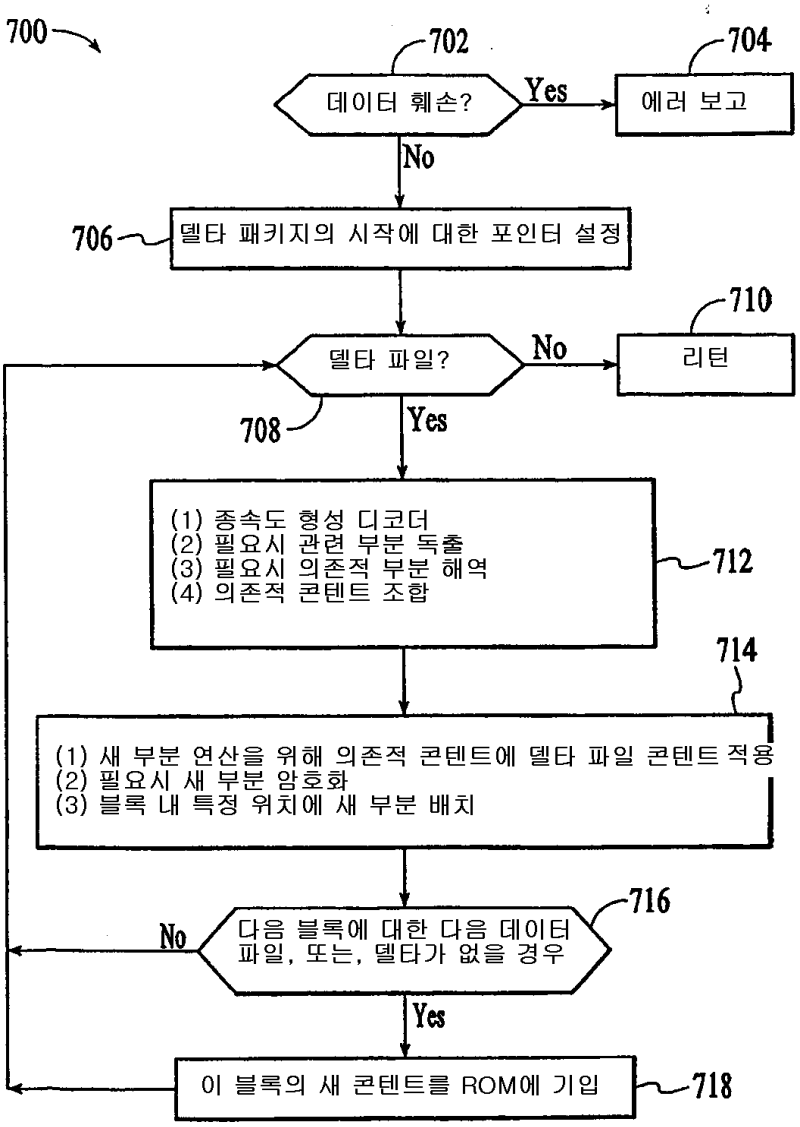
도면5


원본 SFS 이미지
112
새 SFS 이미지
114
이미지 처리 모듈
102
종속도 발생 모듈
104
차이 엔진
106
패키징 엔진
108
100
델타 패키지
122
506
호스팅된 원본 SFS 이미지
112P
ROM
델타 패키지
122
RAM
업데이트 모듈
552
554
새 SFS 이미지
114P
550
500

도면6


600
휴대형 장치에서 델타 파일을 수신
602
휴대형 장치에 호스팅된 다수의 원본 부분들의 의존적 원본 부분들을 조합
604
수신한 델타 파일에 대응하는 의존적 원본 부분들을 식별
606
식별된 한개 이상의 델타 파일에 대응하는 한개 이상의 새 부분을 재구성
608

도면7


700
702
데이터 훼손?
Yes
704
에러 보고
No
706
델타 패키지의 시작에 대한 포인터 설정
710
델타 파일?
No
리턴
708
Yes
(1) 종속도 형성 디코더
(2) 필요시 관련 부분 독출
(3) 필요시 의존적 부분 해역
(4) 의존적 콘텐트 조합
712
714
(1) 새 부분 연산을 위해 의존적 콘텐트에 델타 파일 콘텐트 적용
(2) 필요시 새 부분 암호화
(3) 블록 내 특정 위치에 새 부분 배치
716
No
다음 블록에 대한 다음 데이터 파일, 또는, 델타가 없을 경우
Yes
이 블록의 새 콘텐트를 ROM에 기입
718