(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년08월20일
(11) 등록번호 10-1432314
(24) 등록일자 2014년08월13일

(51) 국제특허분류(Int. Cl.)
G06F 15/16 (2006.01) *G06F 9/06* (2006.01)
G06F 17/40 (2006.01)
(21) 출원번호 10-2008-7020284
(22) 출원일자(국제) 2007년02월13일
심사청구일자 2012년02월03일
(85) 번역문제출일자 2008년08월19일
(65) 공개번호 10-2008-0106187
(43) 공개일자 2008년12월04일
(86) 국제출원번호 PCT/US2007/004048
(87) 국제공개번호 WO 2007/100509
국제공개일자 2007년09월07일
(30) 우선권주장
11/359,276 2006년02월22일 미국(US)
(56) 선행기술조사문헌
US20040162808 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
리, 진
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(74) 대리인
제일특허법인

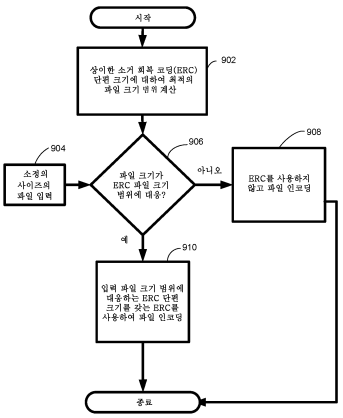
전체 청구항 수 : 총 19 항 심사관 : 권영학

(54) 발명의 명칭 **분산형 네트워크에 저장될 파일들을 인코딩하기 위한 컴퓨터 구현 프로세스 및 이를 실행하기 위한 컴퓨터 판독 가능 매체, 피어 투 피어 네트워크의 저장 신뢰도 및 효율성을 향상시키기 위한 시스템, 및 분산형 네트워크에 저장된 인코딩된 파일을 디코딩하기 위한 컴퓨터 구현 프로세스**

(57) 요 약

분산되는 파일의 크기에 따라 인코딩하는 데 사용되는 단편들의 수를 변화시키는 적응 소거 회복 코딩(ERC)을 사용하는 적응 코딩 저장 시스템이 개시되어 있다. 적응 ERC는 P2P 저장의 효율성 및 신뢰도를 향상시킨다. 또한, P2P 저장 애플리케이션에 대하여 다수의 프로시저가 구현될 수 있다. 일 실시예에서, 소용량의 동적 데이터 파일은 보다 신뢰도 높은 피어 또는 심지어 서버로 향하게 되고, 대용량의 정적 파일은 신뢰도가 낮은 피어의 저장 용량을 사용하여 저장된다. 또한, 균형잡힌 기여도 및 이득을 위하여, 피어는 P2P 네트워크에 저장되어 있는 것과 동일한 양의 콘텐츠를 호스트하여야 한다. 그 결과, 신뢰도가 낮은 피어는 데이터를 덜 분산시킬 수 있고, 신뢰도가 높은 피어는 데이터를 좀더 많이 분산시킬 수 있다. 또한, 소용량의 파일에는 분산 비용이 높게 할당되고, 대용량의 파일에는 분산 비용이 낮게 할당된다.

대 표 도 - 도9



특허청구의 범위

청구항 1

분산형 네트워크에 저장될 파일들을 인코딩하기 위한 컴퓨터 구현 방법으로서, 컴퓨팅 장치를 사용하여 수행되며,

단편(fragment)들의 상이한 소거 회복 코딩(ERC : erasure resilient coding) 수에 대응하는 최적의 파일 크기 범위들을 계산하는 단계 - 상기 각각의 단편들의 ERC 수는 파일 크기들의 대응하는 범위에 대한 최적의 단편들의 수임 - ;

상이한 크기의 파일들의 세트 중 소정의 파일 크기의 파일을 입력하는 단계;

상기 입력된 파일에 대하여 상기 파일 크기가 단편들의 최소 ERC 수인 2에 대한 상기 파일 크기들의 범위보다 작으면, 소거 회복 코딩을 사용하지 않고 상기 입력된 파일을 인코딩하는 단계; 및

상기 입력된 파일에 대하여 상기 입력된 파일의 파일 크기가 상기 파일 크기들의 범위에 대응하면, 상기 입력된 파일의 파일 크기 범위에 대응하는 최적의 단편들의 수 및 소거 회복 코딩을 사용하여 상기 입력된 파일을 인코딩하는 단계 - 상기 소거 회복 코딩은 상기 입력된 파일을 k개의 원본 단편으로 분리(split)하되, 각각의 원본 단편은 갈로아 필드(Galois Field) GF(q)의 벡터이고, q는 상기 필드의 차수(order)이며, ERC 인코딩된 단편들은 상기 k개의 원본 단편들로부터 생성됨 - ; 및

상기 인코딩된 파일을 저장하는 단계

를 포함하는 컴퓨터 구현 방법.

청구항 2

제1항에 있어서,

분산형 네트워크 내의 하나 이상의 피어(peer)들로 상기 인코딩된 파일을 전송하는 단계를 더 포함하는 컴퓨터 구현 방법.

청구항 3

제1항에 있어서,

피어 신뢰도(reliability) 및 파일 콘텐츠의 원하는 신뢰도에 따라, 상기 인코딩된 파일이 저장될 피어들의 수를 계산하는 단계를 더 포함하는 컴퓨터 구현 방법.

청구항 4

제1항에 있어서,

상기 단편들의 상이한 소거 회복 코딩(ERC) 수에 대응하는 최적의 파일 크기 범위들을 계산하는 단계는, 단편들의 특정한 ERC 수에 적합한 최적의 파일 크기로서, 상이한 단편들의 수 간의 경계를 판정하는 단계를 포함하는 컴퓨터 구현 방법.

청구항 5

제1항에 있어서,

상기 파일이 인코딩을 위해 분리되는 단편들의 수 이상의 상기 인코딩된 파일의 파일 단편들의 세트를 획득하는 단계;

상기 인코딩된 파일의 디코딩된 버전을 획득하기 위하여, 상기 파일이 소거 회복 코딩되었다면 상기 인코딩된 파일 단편들을 소거 회복 디코딩을 이용하여 디코딩하는 단계; 및

상기 인코딩된 파일의 디코딩된 버전을 획득하기 위하여, 상기 파일이 소거 회복 코딩되지 않았으면 상기 인코딩된 단편들을 소거 회복 디코딩 없이 디코딩하는 단계

를 더 포함하는 컴퓨터 구현 방법.

청구항 6

제1항에 있어서,

상기 파일의 인코딩에 사용되는 소거 회복 코딩은 리드 솔로몬(Reed Solomon) 코딩인 컴퓨터 구현 방법.

청구항 7

제6항에 있어서,

상기 파일 크기가 10KB보다 작으면, 소거 회복 코딩이 사용되지 않고,

상기 파일 크기가 10KB 내지 33KB이면, 상기 최적의 단편들의 수는 2이고,

상기 파일 크기가 33KB 내지 100KB이면, 상기 최적의 단편들의 수는 4이고,

상기 파일 크기가 100KB 내지 310KB이면, 상기 최적의 단편들의 수는 8이고,

상기 파일 크기가 310KB 내지 950KB이면, 상기 최적의 단편들의 수는 16이고,

상기 파일 크기가 950KB 내지 2.9MB이면, 상기 최적의 단편들의 수는 32이고,

상기 파일 크기가 2.9MB 내지 8.9MB이면, 상기 최적의 단편들의 수는 64이고,

상기 파일 크기가 8.9MB 내지 26MB이면, 상기 최적의 단편들의 수는 128이고,

상기 파일 크기가 26MB보다 크면, 상기 최적의 단편들의 수는 256인 컴퓨터 구현 방법.

청구항 8

삭제

청구항 9

피어 투 피어(peer-to-peer) 네트워크의 저장 신뢰도 및 효율성을 향상시키기 위한 시스템으로서,

범용 컴퓨팅 장치;

상기 범용 컴퓨팅 장치에 의해 실행 가능한 프로그램 모듈들을 포함하는 컴퓨터 프로그램

을 포함하고,

상기 컴퓨터 프로그램의 프로그램 모듈들은 상기 컴퓨팅 장치로 하여금,

소거 회복 코딩을 이용하여 상이한 크기의 파일들의 세트 중 소정의 크기의 파일을 인코딩하기 위하여 최적의 단편들의 수를 판정하고 - 상기 소거 회복 코딩은 상기 파일을 몇 개의 원본 단편으로 분리하고, 각각의 원본 단편은 갈로아 필드(Galois Field) GF(q)의 벡터이고, q는 상기 필드의 차수(order)이며, ERC 인코딩된 단편들은 상기 몇 개의 원본 단편들로부터 생성됨 -,

소거 회복 코딩을 이용하여 상기 파일을 인코딩하기 위한 최적의 단편들의 수가 1이면, 소거 회복 코딩을 이용하여 상기 파일을 인코딩하지 않고,

상기 최적의 단편들의 수가 2 이상이면, 상기 파일을 상기 최적의 단편들의 수로 분리하고, 소거 회복 코딩을 사용하여 상기 파일을 인코딩함으로써 상기 파일을 인코딩하고,

상기 인코딩된 파일을 저장

하도록 하는 시스템.

청구항 10

제9항에 있어서,

피어 신뢰도 및 파일 콘텐츠의 원하는 신뢰도에 따라, 상기 인코딩된 파일 단편들이 저장될 피어들의 수를 계산하기 위한 프로그램 모듈을 더 포함하는 시스템.

청구항 11

제10항에 있어서,

상기 소거 회복 코딩을 이용하여 인코딩된 파일을 네트워크 상의 하나 이상의 피어들로 분산하기 위한 프로그램 모듈을 더 포함하는 시스템.

청구항 12

제11항에 있어서,

상기 파일을 분산하기 위한 프로그램 모듈은,

분산형 네트워크 내의 각각의 피어의 신뢰도를 판정하고,

상기 파일의 크기를 판정하고,

상기 파일을 분산하기 위하여 상기 판정된 파일의 크기에 기초하여 선택된 신뢰도를 갖는 하나 이상의 피어들을 사용하기 위한 서브모듈들을 포함하는 시스템.

청구항 13

제12항에 있어서,

상기 파일을 분산하기 위한 프로그램 모듈은,

상기 파일이 대용량이면, 상기 대용량 파일을 분산하기 위하여 소정의 문턱치보다 낮은 신뢰도를 갖는 피어들을 사용하고,

상기 파일이 대용량이 아니면, 상기 파일을 분산하기 위하여 소정의 문턱치보다 높은 신뢰도를 갖는 피어들을 사용하기 위한 서브모듈들을 포함하는 시스템.

청구항 14

제11항에 있어서,

상기 파일을 분산하기 위한 프로그램 모듈은,

분산형 네트워크 내의 각각의 피어의 신뢰도를 판정하고,

상기 파일이 정적(static)인지 여부를 판정하고,

상기 파일이 정적이면, 상기 파일을 분산하기 위하여 소정의 문턱치보다 낮은 신뢰도를 갖는 피어들을 사용하고,

상기 파일이 정적이지 않으면, 상기 파일을 분산하기 위하여 소정의 문턱치보다 높은 신뢰도를 갖는 피어들을 사용하기 위한 서브모듈들을 포함하는 시스템.

청구항 15

제11항에 있어서,

상기 파일을 분산하기 위한 프로그램 모듈은,

분산형 네트워크 내의 각각의 피어의 신뢰도를 판정하고,

상기 파일의 변화들을 모니터하고,

상기 파일을 신뢰도가 소정의 문턱치보다 높은 피어들에 우선 분산하고,

상기 파일이 변화되지 않은 것으로 관찰되면, 상기 파일을 신뢰도가 소정의 문턱치보다 낮은 피어들에 재분산하기 위한 서브모듈들을 포함하는 시스템.

청구항 16

제9항에 있어서,

분산형 네트워크의 효율성을 향상시키기 위해 서버를 사용하는 프로그램 모듈을 더 포함하고,

상기 서버를 사용하는 프로그램 모듈은,

상기 분산형 네트워크의 모든 동적(dynamic) 파일들을 상기 서버에 백업(back up)하고,

상기 서버에 백업된 동적 파일들이 변화되었는지를 알기 위하여 상기 분산형 네트워크의 서버 및 피어들을 주기적으로 체크시키고,

상기 동적 파일들이 변화되지 않았으면, 이들 파일들을 정적인 것으로 지정하고, 이들을 모두 함께 대용량 파일로 번들링(bundling)하고,

소거 회복 코딩을 이용하여 상기 대용량 파일을 상기 분산형 네트워크로 분산하기 위한 서브모듈들을 더 포함하는 시스템.

청구항 17

제9항에 있어서,

소거 회복 코딩을 이용하여 소정의 크기의 파일을 인코딩하기 위하여 최적의 단편들의 수를 판정하기 위한 프로그램 모듈은,

단편들의 각각의 가능한 소거 회복 코딩 수에 대하여 최적의 파일 크기 범위를 판정하고 - 상기 각각의 단편들의 수는 그 대응하는 범위에 대한 최적의 단편들의 수임 -,

입력되는 파일의 크기가 어느 파일 크기 범위 내에 들어 있는지를 판정하고,

상기 입력되는 파일의 크기가 들어 있는 최적의 파일 크기 범위에 대응하는 최적의 단편들의 수로서 사용하기 위한 서브모듈들을 포함하는 시스템.

청구항 18

분산형 네트워크에 저장된 인코딩된 파일을 디코딩하기 위한 컴퓨터 구현 방법으로서, 상기 방법은 컴퓨팅 장치에 의해 수행되고,

상기 파일을 인코딩하는 데 사용되었던 단편들의 수 이상의 인코딩된 파일의 단편들의 세트를 리트리브(retrieve)하는 단계 - 상기 파일은 소정의 파일 크기에 대하여 최적의 단편들의 수로 소거 회복 인코딩되었으며 피어 신뢰도 및 파일 콘텐츠의 원하는 신뢰도에 따라 결정된 몇 개의 피어들에 저장됨 - ; 및

상기 인코딩된 파일의 디코딩된 버전을 획득하기 위하여 소거 회복 디코딩을 이용하여 상기 인코딩된 단편들을 디코딩하는 단계

를 포함하는 컴퓨터 구현 방법.

청구항 19

제18항에 있어서,

상기 인코딩된 단편들 중 적어도 일부는 상기 분산형 네트워크 내의 하나 이상의 피어들의 저장 매체로부터 리트리브되는 컴퓨터 구현 방법.

청구항 20

제18항에 있어서,

상기 인코딩된 단편들 중 적어도 일부는 상기 분산형 네트워크 상의 서버의 저장 매체로부터 리트리브되는 컴퓨터 구현 방법.

명 세 서

배 경 기 술

[0001] P2P(Peer-to-Peer) 애플리케이션에서, 피어들은 P2P 서비스에 가입(join)할 때, 자신들과 함께 네트워크 밴드폭 및/또는 하드 드라이브 저장 리소스를 가져온다. P2P 시스템에 대한 수요가 증가함에 따라, 시스템의 용량도 증가한다. 이는 클라이언트-서버 시스템과는 현저하게 대조되는 것으로, 클라이언트-서버 시스템에서는 서버의 용량이 고정되어 있고 클라이언트-서버 시스템의 제공업자가 비용을 지불한다. 그 결과, P2P 시스템은 스케일 조절이 가능하기 때문에, 클라이언트-서버 시스템에 비해 실행시킬 때 훨씬 경제적이고 우수하다.

[0002] P2P 시스템에서, 피어는 밴드폭뿐만 아니라 다른 피어들을 서브하는 저장 공간에 기여한다. 피어에 의해 기여되는 총체적인 저장 공간은 분산형 저장 클라우드(distributed storage cloud)를 형성한다. 데이터는 클라우드에 저장되고, 클라우드로부터 리트리브될 수 있다. P2P 저장은 다수의 애플리케이션에 사용될 수 있다. 하나의 애플리케이션은 분산형 백업이다. 피어는 자신의 데이터를 P2P 클라우드에 백업(back up)할 수 있다. 피어에게 고장이 발생하면, 데이터는 클라우드로부터 복구될 수 있다. 다른 P2P 애플리케이션은 분산형 데이터 액세스이다. 클라이언트는 복수의 데이터 유지 피어로부터 데이터를 동시에 리트리브할 수 있기 때문에, P2P 리트리브는 단일의 소스로부터 데이터를 리트리브하는 것에 비해 보다 높은 쓰루풋을 가질 수 있다. 또 다른 애플리케이션은 주문형 영화 보기(on-demand movie viewing)이다. 미디어 서버는 P2P 클라우드에 영화 파일을 우선적으로 시드(seed)할 수 있다. 클라이언트가 영화를 볼 때, P2P 클라우드 및 서버 모두로부터 영화를 스트리밍할 수 있기 때문에, 서버 로드를 감소시키고, 네트워크 백본(netowrk backbone) 상의 트래픽을 감소시키고, 스트리밍 영화 품질을 향상시킬 수 있다.

[0003] P2P 네트워크의 피어가 서버처럼 동작할 수 있지만, 이들은 하나의 중요한 측면, 즉 신뢰도(reliability) 측면에서 상업용 웹/데이터베이스 서버와는 다르다. 피어는 대게 그 여분의 하드 드라이브 공간 및 유휴 중인 밴드폭을 이용하여 P2P 애플리케이션을 지원하는 통상의 컴퓨터이기 때문에, 통상적인 서버보다는 신뢰도가 다소 떨어진다. 사용자는 때때로 피어 컴퓨터 또는 P2P 애플리케이션의 전원을 끄는 선택을 할 수 있다. 강제적인 수요(compulsory need), 예를 들어 대용량 파일의 업로드/다운로드는 피어에게 P2P 액티비티에 필요한 밴드폭을 부족하게 할 수 있다. 피어 컴퓨터는 소프트웨어/하드웨어의 업그레이드 또는 패치(patch)의 필요성 또는 바이러스 침투 때문에 오프라인일 수 있다. 또한, 피어의 컴퓨터 하드웨어 및 네트워크 링크는 본래 통상적인 서버 컴퓨터 및 그 상업용 네트워크 링크(이들은 신뢰도가 높게 설계되어 있음)에 비해 신뢰도가 훨씬 떨어진다. 상업용 서버/서버 클러스터는 "six nine" 신뢰도(실패율이 10^{-6}임, 이러한 실패율에서는 해마다 약 30초의 다운 시간이 허용됨)로 설계되지만, 좋은 소비자 피어는 단지 "two nine" 신뢰도(실패율이 10^{-2}이거나, 날마다 약 15분의 다운 시간)를 가질 수 있고, 피어가 단지 50%의 신뢰도(50%의 시간에 다운됨), 심지어는 10%의 신뢰도(90%의 시간에 다운됨)를 갖는 것도 통상적이다.

[0004] 대부분의 P2P 애플리케이션, 예를 들어 P2P 백업 및 데이터 리트리브는 P2P 저장에 대한 신뢰도를 서버의 신뢰도("six nine" 신뢰도)와 동일한 레벨로 유지하기를 원한다. 피어의 최소 밴드폭 및 저장 리소스를 사용하여 신뢰도 높고 효율적인 P2P 저장을 구축하기 위한 방법의 도전에 직면해 있다.

[0005] <발명의 개요>

[0006] 피어 투 피어(P2P) 네트워크에서 데이터를 효율적이고 신뢰도 높게 저장하기 위한 적응(adaptive) 코딩 저장 시스템 및 방법이 개시되어 있다. 적응 코딩 저장 시스템 및 방법은, 저장되고 분산되는 파일 크기에 기초하여, 소거 회복 코딩(ERC : erasure resilient coding)을 위한 단편들의 수, 즉 단편들의 ERC 수를 조절한다.

[0007] 적응 코딩 저장 시스템의 여러 실시예는 P2P 네트워크의 효율성 및 신뢰도를 향상시키기 위한 프로시저들을 사용한다. 예를 들어, 일 실시예에서는, 작은 동적 데이터가 좀더 신뢰도 높은 피어, 또는 심지어 서버(만약 서버 컴포넌트 지원이 이용 가능한 경우)로 향하게 된다. 또한, 다른 실시예에서는, 균형잡힌 P2P 네트워크를 위하여, 신뢰도가 낮고 소용량 파일들을 분산시키는 피어들은 적은 데이터를 분산시키도록 허용된다.

[0008] 단, 배경 기술 섹션에서 설명된 기존의 피어 투 피어 저장 및 분산형 시스템의 상술한 한계는 본 발명에 따른 적응 코딩 저장 시스템의 특정 구현에 의해 해결될 수 있고, 본 시스템 및 프로세스는 위에서 언급된 단점들 중 일부 또는 전부를 해결하려는 구현에 어떤 방식으로든 제한되지 않는다. 오히려, 이하의 설명에서 명백해지는 바와 같이, 본 시스템 및 프로세스는 보다 더 폭 넓은 응용을 포함함다.

[0009] 또한, 본 발명의 개요는 이하 실시예에서 상술되는 개념의 선택을 더욱 간략화된 형태로 도입하기 위하여 제공되는 것임을 알아야 한다. 본 발명의 개요는 청구 대상의 핵심적인 특징 또는 본질적인 특징을 식별하기 위한 것이 아니며, 또한 청구 대상의 범위를 판정하는 것을 돕기 위한 것도 아니다.

실 시 예

[0024] 이하의 본 적응 코딩 저장 시스템의 바람직한 실시예의 설명에서는, 본 명세서의 일부를 형성하는 첨부 도면을 참조하였고, 이 첨부 도면에는 적응 코딩 저장 시스템이 실행될 수 있는 구체적인 실시예를 예로서 도시하였다. 본 적응 코딩 저장 시스템의 범위를 벗어나지 않고도, 다른 구현이 사용될 수 있고, 또한 구조적인 변화가 행해질 수 있음이 이해될 것이다.

[0025] **1.0 예시적인 운영 환경**

[0026] 도 1은 본 발명이 구현되기에 적합한 컴퓨팅 시스템 환경(100)의 일례를 도시하고 있다. 컴퓨팅 시스템 환경(100)은 적합한 컴퓨팅 환경의 일례에 불과하며, 본 발명의 용도 또는 기능성의 범위에 관해 어떤 제한을 암시하고자 하는 것이 아니다. 컴퓨팅 환경(100)이 예시적인 운영 환경(100)에 도시된 컴포넌트들 중 임의의 하나 또는 그 컴포넌트들의 임의의 조합과 관련하여 어떤 의존성 또는 요구사항을 갖는 것으로 해석되어서는 안된다.

[0027] 본 발명은 많은 기타 범용 또는 특수 목적의 컴퓨팅 시스템 환경 또는 구성에서 동작할 수 있다. 본 발명에서 사용하는 데 적합할 수 있는 잘 알려진 컴퓨팅 시스템, 환경 및/또는 구성의 예로는 퍼스널 컴퓨터, 서버 컴퓨터, 핸드-헬드 또는 랩톱 장치, 이동식 컴퓨터, 또는 이동 전화 및 PDA와 같은 통신 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 시스템, 셋톱 박스, 프로그램가능한 가전제품, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 상기 시스템들이나 장치들 중 임의의 것을 포함하는 분산 컴퓨팅 환경, 기타 등등이 있지만 이에 제한되는 것은 아니다.

[0028] 본 발명은 일반적으로 마이크 어레이(198)의 컴포넌트를 포함한 하드웨어 모듈과 조합하여 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터 실행가능 명령어와 관련하여 기술될 것이다. 일반적으로, 프로그램 모듈은 특정 태스크를 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 개체, 컴포넌트, 데이터 구조 등을 포함한다. 본 발명은 또한 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 태스크가 수행되는 분산 컴퓨팅 환경에서 실시되도록 설계된다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 메모리 저장 장치를 비롯한 로컬 및 원격 컴퓨터 저장 매체 둘다에 위치할 수 있다. 도 1과 관련하여, 본 발명을 구현하는 예시적인 시스템은 컴퓨터(110) 형태의 범용 컴퓨팅 장치를 포함한다.

[0029] 컴퓨터(110)의 컴포넌트들은 처리 장치(120), 시스템 메모리(130), 및 시스템 메모리를 비롯한 각종 시스템 컴포넌트들을 처리 장치(120)에 연결시키는 시스템 버스(121)를 포함하지만 이에 제한되는 것은 아니다. 시스템 버스(121)는 메모리 버스 또는 메모리 컨트롤러, 주변 장치 버스 및 각종 버스 아키텍처 중 임의의 것을 이용하는 로컬 버스를 비롯한 몇몇 유형의 버스 구조 중 어느 것이라도 될 수 있다. 예로서, 이러한 아키텍처는 ISA(Industry Standard Architecture) 버스, MCA(Micro Channel Architecture) 버스, EISA(Enhanced ISA) 버스, VESA(Video Electronics Standard Association) 로컬 버스, 그리고 메자닌 버스(Mezzanine Bus)로도 알려진 PCI(Peripheral Component Interconnect) 버스 등을 포함하지만 이에 제한되는 것은 아니다.

[0030] 컴퓨터(110)는 통상적으로 각종 컴퓨터 판독가능 매체를 포함한다. 컴퓨터(110)에 의해 액세스 가능한 매체는 그 어떤 것이든지 컴퓨터 판독가능 매체가 될 수 있고, 이러한 컴퓨터 판독가능 매체는 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체를 포함한다. 예로서, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함하지만 이에 제한되는 것은 아니다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함한다.

[0031] 컴퓨터 저장 매체는 RAM, ROM, PROM, EPROM, EEPROM, 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 컴퓨터(110)에 의해 액세스되고 원하는 정보를 저장할 수 있는 임의의 기타 매체를 포함하지만 이에 제한되는 것은 아니다. 통신 매체는 통상적으로 반송파(carrier wave) 또는 기타 전송 메커니즘(transport mechanism)과 같은 피변조 데이터 신호(modulated data signal)에 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터 등을 구현하고 모든 정보 전달 매체를 포함한다. "피변조 데이터 신호"라는 용어는, 신호 내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 예로서, 통신 매체는 유선 네트워크 또는 직접 배선 접속(direct-wired connection)과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함한다. 상술된 매체들의 모

든 조합이 또한 컴퓨터 판독가능 매체의 영역 안에 포함되는 것으로 한다.

[0032] 시스템 메모리(130)는 판독 전용 메모리(ROM)(131) 및 랜덤 액세스 메모리(RAM)(132)와 같은 휘발성 및/또는 비휘발성 메모리 형태의 컴퓨터 저장 매체를 포함한다. 시동 중과 같은 때에, 컴퓨터(110) 내의 구성요소들 사이의 정보 전송을 돕는 기본 루틴을 포함하는 기본 입/출력 시스템(BIOS)(133)은 통상적으로 ROM(131)에 저장되어 있다. RAM(132)은 통상적으로 처리 장치(120)가 즉시 액세스 할 수 있고 및/또는 현재 동작시키고 있는 데이터 및/또는 프로그램 모듈을 포함한다. 예로서, 도 1은 운영 체제(134), 애플리케이션 프로그램(135), 기타 프로그램 모듈(136) 및 프로그램 데이터(137)를 도시하고 있지만 이에 제한되는 것은 아니다.

[0033] 컴퓨터(110)는 또한 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 저장매체를 포함한다. 단지 예로서, 도 1은 비이동식·비휘발성 자기 매체에 기록을 하거나 그로부터 판독을 하는 하드 디스크 드라이브(141), 이동식·비휘발성 자기 디스크(152)에 기록을 하거나 그로부터 판독을 하는 자기 디스크 드라이브(151), CD-ROM 또는 기타 광 매체 등의 이동식·비휘발성 광 디스크(156)에 기록을 하거나 그로부터 판독을 하는 광 디스크 드라이브(155)를 포함한다. 예시적인 운영 환경에서 사용될 수 있는 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 기억 매체로는 자기 테이프 카세트, 플래시 메모리 카드, DVD, 디지털 비디오 테이프, 고상(solid state) RAM, 고상 ROM 등이 있지만 이에 제한되는 것은 아니다. 하드 디스크 드라이브(141)는 통상적으로 인터페이스(140)와 같은 비이동식 메모리 인터페이스를 통해 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151) 및 광 디스크 드라이브(155)는 통상적으로 인터페이스(150)와 같은 이동식 메모리 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0034] 위에서 설명되고 도 1에 도시된 드라이브들 및 이들과 관련된 컴퓨터 저장 매체는, 컴퓨터(110)를 위해, 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 및 기타 데이터를 저장한다. 도 1에서, 예를 들어, 하드 디스크 드라이브(141)는 운영 체제(144), 애플리케이션 프로그램(145), 기타 프로그램 모듈(146), 및 프로그램 데이터(147)를 저장하는 것으로 도시되어 있다. 여기서 주의할 점은 이들 컴포넌트가 운영 체제(134), 애플리케이션 프로그램(135), 기타 프로그램 모듈(136), 및 프로그램 데이터(137)와 동일하거나 그와 다를 수 있다는 것이다. 이에 관해, 운영 체제(144), 애플리케이션 프로그램(145), 기타 프로그램 모듈(146) 및 프로그램 데이터(147)에 다른 번호가 부여되어 있다는 것은 적어도 이들이 다른 사본(copy)이라는 것을 나타내기 위한 것이다. 사용자는 키보드(162), 및 통상 마우스, 트랙볼(trackball) 또는 터치 패드라고 부르는 포인팅 장치(161) 등의 입력 장치를 통해 명령 및 정보를 컴퓨터(110)에 입력할 수 있다.

[0035] 다른 입력 장치(도시 생략)로는 조이스틱, 게임 패드, 위성 안테나, 스캐너, 라디오 수신기, 및 텔레비전 또는 브로드캐스트 비디오 수신기 등을 포함할 수 있다. 이들 및 기타 입력 장치는 종종 시스템 버스(121)에 연결되는 유선 또는 무선 사용자 입력 인터페이스(160)를 통해 처리 장치(120)에 접속되곤 하지만, 예를 들어 병렬 포트, 게임 포트, USB(universal serial bus), IEEE 1394 인터페이스, 블루투스™ 무선 인터페이스, IEEE 802.11 무선 인터페이스 등과 같은 기타 통상의 인터페이스 및 버스 구조에 의해 접속될 수도 있다. 또한, 컴퓨터(110)는 마이크 또는 마이크 어레이(198)와 같은 음성 또는 오디오 입력 장치뿐만 아니라, 예를 들어 병렬, 직렬, USB, IEEE 1394, 블루투스™ 등과 같은 통상의 유선 또는 무선 인터페이스를 또 포함하는 오디오 인터페이스(199)를 통해 접속되는 스피커(197) 또는 기타 사운드 출력 장치를 포함할 수 있다.

[0036] 모니터(191) 또는 다른 유형의 디스플레이 장치도 비디오 인터페이스(190) 등의 인터페이스를 통해 시스템 버스(121)에 접속될 수 있다. 모니터 외에, 컴퓨터는 프린터(196) 등의 기타 주변 출력 장치를 포함할 수 있고, 이들은 출력 주변장치 인터페이스(195)를 통해 접속될 수 있다.

[0037] 컴퓨터(110)는 원격 컴퓨터(180)와 같은 하나 이상의 원격 컴퓨터로의 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(180)는 또 하나의 퍼스널 컴퓨터, 서버, 라우터, 네트워크 PC, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있고, 통상적으로 컴퓨터(110)와 관련하여 상술된 구성요소들의 대부분 또는 그 전부를 포함하지만, 메모리 저장 장치(181)만이 도 1에 도시되어 있다. 도 1에 도시된 논리적 접속으로는 LAN(171) 및 WAN(173)이 있지만, 기타 네트워크를 포함할 수도 있다. 이러한 네트워킹 환경은 사무실, 전사적 컴퓨터 네트워크(enterprise-wide computer network), 인트라넷, 및 인터넷에서 일반적인 것이다.

[0038] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 네트워크 인터페이스 또는 어댑터(170)를 통해 LAN(171)에 접속된다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 통상적으로 인터넷과 같은 WAN(173)을 통해 통신을 설정하기 위한 모뎀(172) 또는 기타 수단을 포함한다. 내장형 또는 외장형일 수 있는 모뎀(172)은 사용자 입력 인터페이스(160) 또는 기타 적절한 메커니즘을 통해 시스템 버스(121)에 접속된다. 네트워크화된 환경에서, 컴

퓨터(110) 또는 그의 일부와 관련하여 기술된 프로그램 모듈은 원격 메모리 저장 장치에 저장될 수 있다. 예로서, 도 1은 원격 애플리케이션 프로그램(185)이 메모리 장치(181)에 있는 것으로 도시하고 있지만 이에 제한되는 것은 아니다. 도시된 네트워크 접속은 예시적인 것이며 이 컴퓨터들 사이에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 이해할 것이다.

[0039] 일반적으로, 적응 코딩 저장 시스템은 도 2에 도시된 네트워크와 같은 P2P 네트워크에서 동작한다. 특정한 데이터 스트리밍 세션에 있어서, "서버"(200)는 데이터 또는 스트리밍 미디어가 처음에 발원한 P2P 네트워크의 노드로서 규정되고, "클라이언트"(또는 수신기)(210)는 현재 데이터를 요청하는 노드로서 규정되고, "서빙 피어(serving peer)"(220)는 데이터의 완전한 사본 또는 부분적인 사본을 이용하여 클라이언트를 서브하는 노드로서 규정된다.

[0040] 일반적으로, 서버(200), 클라이언트(210) 및 서빙 피어(220)는 모두 인터넷과 같은 네트워크에 접속된 최종 사용자 노드(end-user node)이다. 서버(200)는 항상 데이터를 서브할 수 있기 때문에, 서버 노드는 서빙 피어(220)로도 동작한다. 또한, 서버 노드(200)는 서빙 피어(220)에 의해 행해질 수 없는 관리 기능, 예를 들어 이용 가능한 서빙 피어의 리스트를 유지하고, DRM(digital rights management) 기능을 행하는 등의 관리 기능을 행할 수 있다. 또한, 통상적인 P2P 스킴에서와 마찬가지로, 여기에 기재되는 적응 코딩 저장 시스템은 피어 노드(220)가 더 많이 배치됨에 따라 효율이 증가된다는 이점이 있다. 특히, 피어 노드(220)의 수가 증가함에 따라, 데이터 서버(200)의 부하가 감소되므로, 이에 의해 실행 비용은 더 적어지고, 특정 데이터 전송 세션 동안 각각의 클라이언트 노드(210)가 더욱 좋은 데이터 품질을 수신할 수 있다.

[0041] 또한, 특정 노드의 역할이 변할 수 있음을 명백히 하여야 한다. 예를 들면, 특정 노드는 어느 특정 데이터 전송에서는 클라이언트(210)로서 동작할 수 있고, 다른 세션에서는 서빙 피어(220)로서 동작할 수 있다. 또한, 특정 노드는 하나 이상의 데이터 파일들 또는 이들 파일들의 일부를 전송하면서, 동시에 하나 이상의 다른 서빙 피어로부터 다른 데이터를 수신하기 위하여 클라이언트 노드(210) 및 서버(200) 또는 서빙 피어(220)로서 동시에 동작할 수 있다.

[0042] 데이터 전송 동안, 클라이언트(200)는 우선 원하는 데이터의 전부 또는 일부를 유지하는 다수의 인접(close-by) 피어(200)들의 위치를 찾고, 복수의 피어(서버(200)를 포함할 수 있음)로부터 데이터를 수신한다. 결과적으로, 각각의 서빙 피어(220)는 클라이언트(210)의 다운로드 요청의 일부를 서비스함으로써 전체적인 업로드 부담을 감소시키는 것에 의해 서버(200)를 돕는 역할을 한다. 그 결과, 특히 여러 클라이언트가 존재하는 경우에는, 서버(200)를 도울 서빙 피어(220)가 많으면 이용 가능한 서빙 밴드폭이 현저하게 높기 때문에, 클라이언트(210)가 보다 높은 데이터 품질을 종종 수신할 수 있다.

[0043] 지금까지는 예시적인 동작 환경에 대해 설명하였고, 본 실시예 섹션의 나머지 부분은 적응 코딩 저장 시스템 및 프로세스를 사용하는 프로그램 모듈에 대하여 설명할 것이다.

[0044] ## 2.0 신뢰도 높고 효율적인 피어 투 피어 저장

[0045] 적응 코딩 저장 시스템은 ERC 코딩을 사용할지 여부를 적응적으로 판정하고, 최적의 신뢰도 및 효율성을 위해 소정의 파일 크기에 대한 ERC 코딩에 최적의 단편들의 수를 사용하는 적응 소거 회복 코딩(ERC) 스킴을 제공한다. 파일의 ERC 코딩에 사용되는 단편들의 수를 본 논의의 목적상 "단편들의 ERC 수"라고 칭하도록 한다. 이하의 단락들은 피어 투 피어(P2P) 저장 효율성 및 신뢰도, 및 P2P 네트워크에서의 ERC 사용에 대한 논의뿐만 아니라 사용되는 단편들의 ERC 수에 관한 논의를 제공한다. 그 후, 적응 코딩 저장 시스템 및 프로세스의 다양한 실시예가 논의된다.

[0046] ### 2.1 P2P 저장에서의 신뢰도 : 데이터 중복성(redundancy)

[0047] 신뢰도가 낮은 부분을 갖는 시스템에 신뢰도를 부여할 수 있는 특별한 해결책은 중복성을 사용하는 것이다. 만약 네트워크의 각각의 개별적인 피어가 p의 신뢰도를 가지면, p_0의 원하는 신뢰도를 실현하기 위하여, 단순히 n개의 피어에 정보를 복사(replicate)할 수 있다.

수학식 1

[0048] $$n = \log(1-p_0)/\log(1-p),$$

[0049] 여기서, n은 정보를 유지하는 피어의 수이다. 리트리브의 시점에, 클라이언트는 정보 저장 피어를 하나씩 접촉할 수 있다. 정보 저장 피어들 중 하나가 온라인인 한, 정보는 높은 신뢰도로 리트리브될 수 있다.

[0050] 단순한 복사 전략은 신뢰도는 높게 실현하지만, 효율적이지 않다. 도 3은 "six nine" 신뢰도를 실현하는 데 필요한 정보 저장 피어의 수를 도시한다. 피어 신뢰도가 50%이면, 정보를 복사하여 20개의 피어에 저장할 필요가 있다. 이는 정보를 분산하고 저장하기 위한 밴드폭 및 저장 공간을 20배 이상으로 되게 한다. 명백하게는, 정보 신뢰도 대신에 효율성이 희생되었다.

[0051] 2.2 P2P에서의 소거 회복 코딩

[0052] 여전히 동일한 신뢰도를 유지하면서 효율성을 향상시키는 데에, ERC는 유용한 툴일 수 있다. ERC는 원본 파일을 k개의 원본 단편들 $\{x_i\}$, $i=0,\cdots,k-1$로 분리하는 데, 이 단편 각각은 갈로아 필드(Galois Field) GF(q)(여기서, q는 필드의 차수(order)임) 상의 벡터이다. 만약 64KB의 길이의 파일을 인코딩한다고 가정할 때, $q=2^{16}$ 및 k=16을 사용하면, 각각의 단편은 4KB로 되고, 2K 워드(word)로 이루어지게 되며, 각각의 워드는 GF(2^{16})의 구성 요소로 이루어진다. 그리고, ERC는 원본 단편으로부터 코딩된 단편을 생성한다. ERC 코딩된 단편은 이하의 연산에 의해 형성된다.

수학식 2

[0053] $$c_j = \mathbf{G}_i \cdot [x_0 \quad x_1 \quad \cdots \quad x_{k-1}]^t,$$

[0054] 여기서 c_j는 코딩된 단편이고, $\mathbf{G}_i$는 k 차원 생성기 벡터이고, 수학식 2는 행렬 곱이며, 이들 모두 GF(q) 상에 있다. 디코딩 시에, 피어는 m개(여기서 m은 k와 동일하거나 k보다 약간 큰 수)의 코딩된 단편들을 모으고, k개의 원본 단편들을 디코딩하려고 한다. 이는 이하의 수학식의 해를 구하는 것과 동일하다.

수학식 3

[0055] $$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_{k-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix},$$

[0056] 만약 생성기 벡터에 의해 형성된 행렬이 풀 랭크(full rank) k를 가지면, 원본 메시지가 복구될 수 있다.

[0057] 이용 가능한 ERC에는 여러 가지가 있다. 특히 관심을 끄는 것은 RS(Reed-Solomon) 코드이다. RS 코드는 구조화된 생성기 벡터를 사용하고, MDS(maximum distance separable)이다. 그 결과, 임의의 k개의 특이적으로(distinctive) 코딩된 단편이 원본 단편을 디코딩할 수 있을 것이다. RS 코드의 다른 이점은, 코딩된 단편이 위의 생성기 벡터의 인덱스 i에 의해 용이하게 식별되고 관리될 수 있기 때문에, 중복된 RS 코드의 검출이 용이하다는 것이다. 이하의 ERC 논의에서는, RS 코드가 사용된다고 가정한다. 그러나, 적응 코딩 저장 시스템은 임의의 수의 통상적인 ERC들로 구현될 수 있다.

[0058] 2.3 ERC : 단편들의 수

[0059] P2P 저장에서 ERC를 사용하는 것에 의해, 데이터 파일이 보다 많은 피어에 분산되지만, 각각의 피어는 단지 원본 파일의 1/k 크기인 하나의 코딩된 단편만을 저장할 필요가 있으며, 이는 동일한 레벨의 신뢰도를 실현하는

데 필요한 밴드폭 및 저장 공간을 전반적으로 감소시키기 때문에, 이에 의해 효율성이 향상된다. 소정의 원하는 신뢰도 레벨을 실현하기 위하여 코딩된 단편들이 분산될 필요가 있는 피어들의 수를 n_1으로 한다. RS 코드는 MDS 코드이기 때문에, k개의 특이적으로 코딩된 단편들을 유지하는 k개의 피어들이면 원본 파일을 복구하는 데 충분할 것이다. 정확하게 m개의 이용 가능한 피어가 있을 확률은 이항 분포를 통해 계산될 수 있다.

수학식 4

$$p(m, n_1) = \binom{n_1}{m} p^m (1-p)^{n_1-m}.$$

[0060]

[0061] 따라서, p, p_0 및 k로부터 n_1을 계산할 수 있다.

수학식 5

$$n_1 = \arg\min_{o} \left\{ \sum_{m<k} \binom{o}{m} p^m (1-p)^{o-m} < 1-p_0 \right\}.$$

[0062]

[0063] 복사율(replication ratio) r은 이하와 같이 규정된다.

수학식 6

[0064] $$r = n_1 / k.$$

[0065] 파일의 r개의 사본이 P2P 클라우드에 분산되고 저장될 필요가 있으므로, 복사율 r은 효율성의 좋은 표시자이다.

[0066] 단편들의 상이한 ERC 수 k에 대해 "six nine" 신뢰도를 실현하기 위하여 원하는 복사율이 도 4에 도시되어 있다. ERC를 사용하면 필요한 복사율을 크게 감소시킴을 알 수 있다. 단편들의 ERC 수 k=256과 비(non) ERC(k=1)를 비교하면, 원하는 복사율이 10%의 피어 신뢰도에 대해서는 r=132에서 r=13.1로 감소되고, 50%의 피어 신뢰도에 대해서는 r=20에서 r=2.5로 감소되고, 99%의 피어 신뢰도에 대해서는 r=3에서 r=1.05로 감소된다. ERC는 신뢰도를 희생시키지 않고도 효율성을 향상시킬 수 있다.

[0067] 또한, 단편들의 ERC 수가 클수록 복사율을 더욱 감소시킴을 알 수 있다. 50%의 피어 신뢰도에 의하면, k=8에서 16, 32, 64, 128 및 256으로 가면 복사율은 r=5.75에서 4.375, 3.53, 3.02, 2.68 및 2.48로 감소되게 된다. 대응하는 효율성의 향상률은 각각 24%, 19%, 15%, 11% 및 8%이다. 이는 보다 큰 효율성을 위해서는 단편들의 ERC 수가 큰 것을 사용해야 함을 제시하는 것으로 보인다.

[0068] 그러나, 단편들의 ERC 수가 더 크다는 것은, 코딩된 단편을 저장하고 리트리브하기 위해 더욱 많은 피어가 필요함을 의미한다. 도 5에는, "six nine" 신뢰도를 실현하기 위하여 코딩된 단편을 유지하는 데 필요한 피어들의 수가 도시되어 있다. 다시 50%의 피어 신뢰도에 의하면, k=8에서 16, 32, 64, 128 및 256으로 가면 정보 저장 피어의 수는 n_1=46에서 70, 113, 193, 343 및 630으로 증가된다. k가 두 배 증가할 때마다 정보를 저장하는 데 필요한 피어의 수가 52%, 61%, 71%, 78%, 84% 늘어나게 된다. 또한, k를 두 배로 하면, 정보 리트리브 동안 접촉되는 피어의 수를 적어도 두 배 필요로 한다.

[0069] 대부분의 실제 P2P 네트워크에서, 피어들 간에 접속을 구축하는 것은 무시못할(non-trivial) 양의 오버헤드를 필요로 한다. 오버헤드의 한 부분은 적절한 피어 아이덴티티의 리트리브 및 (예를 들어, DHT(Distributed Hash Table)를 통한) 적절한 라우팅 경로의 탐색 때문일 수 있다. 오버헤드의 다른 부분은, 만약 하나의 피어 또는 두 피어 모두가 소정의 NAT(network address translation) 알고리즘, 예를 들어 STUN(simple traversal of UDP through NAT) 뒤에 있는 경우에, 그 NAT 알고리즘을 호출할 필요가 있기 때문이다. 두 피어 간의 접속을 구축하기 위한 평균 오버헤드를 *overhead*(이 예에서는 16KB로 설정됨)라고 하면, 크기 s의 파일을 저장하는 데 필요한 전반적인 네트워크 밴드폭을 이하와 같이 계산할 수 있다.

수학식 7

[0070] $$store\_bandwidth = s * r + n_1 * overhead.$$

[0071] 수학식 7에 의하면, 단편들의 ERC 수가 더욱 크다고 해서 항상 최상의 효율성으로 되는 것은 아님을 인식할 수 있다. 대신에, 작은 파일에 대해서는, 단편들의 ERC 수가 작거나 또는 심지어 비 ERC가 사용되어야 한다. 상이한 파일 크기들 및 단편들의 ERC 수에 대하여 수학식 (7)에서 요구되는 전반적인 밴드폭을 계산하고, 도 6에 그 곡선을 도시하였다. 단편들의 상이한 ERC 수 간의 경계는 단편들의 특정한 ERC 수에 적절한 최적의 파일 크기 범위이다. 예를 들면, 도 6의 아래에 있는 곡선은 파일 크기 경계를 도시하며, 그 파일 크기 경계 아래에서는 비 ERC가 사용되어야 하고, 그 파일 크기 경계 위에서는 k=2의 단편의 수를 갖는 ERC가 사용되어야 한다. 흥미로운 점은, 파일 크기 경계는 피어 이용 가능성과 상대적으로 무관하며, 이는 최적의 ERC 단편 파라미터의 선택을 매우 간단하게 한다는 것이다. 일반적으로, 대략 10KB보다 작은 파일에 있어서는, ERC가 사용되지 않아야 한다. k=2, 4, 8, 16, 32, 128 및 256의 단편들의 수를 갖는 ERC에 있어서, 가장 적절한 파일 크기 범위는 각각 대략 10-33KB, 33-100KB, 100-310KB, 310-950KB, 950KB-2.9MB, 2.9MB-8.9MB, 8.9-26MB, >26MB 이다.

[0072] ## 2.4 적응 ERC 스킴

[0073] 적응 코딩 저장 시스템 및 방법은 P2P 네트워크의 콘텐츠를 신뢰도 높게 효율적으로 저장하기 위하여 단편들의 적절한 ERC 수를 적응적으로 선택한다. 도 6에 구축된 파일 경계 곡선을 사용하면, 시스템의 일 실시예는 상이한 파일 크기에 대하여, 비 ERC, 및 k=2, 4, 8, 16, 32, 64, 128, 256의 단편의 수를 갖는 ERC를 사용하는 것을 적응적으로 선택한다. 적응 ERC 접근 방식이 고정된 파라미터 ERC와 비교되어, 그 네트워크 밴드폭 사용에서의 차이가 도 7 및 도 8에 도시되어 있으며, 여기서 피어 신뢰도는 각각 50% 및 99%이다. k=1(비 ERC), 8, 32 및 256과 같은 단편의 고정된 ERC 수를 사용하는 것과 비교하면, 적응 ERC 방법은 50%의 피어 신뢰도에 대해서는 61%, 26%, 25%, 및 50%의 평균만큼, 99%의 피어 신뢰도에 대해서는 평균 50%, 18%, 29% 및 57%의 평균만큼 효율성을 향상시킬 수 있다. 효율성의 향상이 현저하다.

[0074] 가장 일반적인 관점에서, 적응 코딩 저장 프로세스의 일 실시예가 도 9에 도시되어 있다. 프로세스 단계(902)에 도시된 바와 같이, 적응 코딩 저장 시스템은 상이한 수의 단편에 대하여 최적의 파일 크기 경계를 계산한다. 인코딩될 소정의 파일 크기의 파일이 입력된다(프로세스 단계(904)). 프로세스 단계(906)에 도시된 바와 같이, 입력된 파일 크기가 비 소거 코딩(k=1)에 대응하는지 여부에 관한 체크가 행해진다. 만약 입력된 파일 크기가 ERC에 대응하지 않으면, 파일은 ERC를 사용하지 않고 인코딩된다(프로세스 단계(908)). 만약 파일 크기가 ERC 파일 크기 범위에 대응하면, 파일은 ERC 코딩 및 입력된 파일의 파일 크기에 대응하는 단편의 수를 사용하여 인코딩되고, 여기서 단편의 수는 그 크기 파일에 대한 최적의 단편들의 수이다(프로세스 단계(910)).

[0075] 여기에 기재되는 적응 ERC 프로세스의 대부분의 애플리케이션은 P2P 백업 또는 복구에 있다. 피어는 네트워크 내의 다른 피어들에 파일을 백업하고, 그 후 파일이 손실된 경우(예를 들면, 컴퓨터 고장(computer crash)으로 인해 파일이 손실된 경우)에 네트워크 내의 피어들로부터 파일을 리트리브함으로써 이 파일을 복구할 수 있다. 일반적으로, 도 10은 적응 코딩 저장 기술이 P2P 시스템에서 사용될 수 있는 방식을 도시하는 예시적인 동작 흐름도를 도시한다. 단, 도 10에서 파선 또는 점선으로 표현되는 임의의 박스 및 박스 간의 상호 접속은 여기서 기재되는 적응 코딩 저장 시스템의 다른 실시예를 나타내고, 이하 기술되는 바와 같이 이들 다른 실시예의 일부 또는 전부는 이 문서 전체를 통해 설명되는 그 밖의 다른 실시예들과 조합되어 사용될 수 있음을 이해하여야 한다.

[0076] 특히, 도 10에 도시된 바와 같이, 네트워크의 피어에 데이터를 백업하고 싶은 경우와 같은 데이터 전송 동작에 앞서, 서버(200) 또는 피어(220)는 저장을 위하여 다른 피어로 전송될 데이터를 인코딩한다(단계(1000)). 적응 코딩 저장 시스템은 임의의 수의 종래의 코덱, 예를 들면, MPEG 1/2/4, WMA, WMV 등과 함께 동작할 수 있다. 또한, 인코딩 프로세스(1000) 동안, 서버(200) 또는 피어(220)는 데이터 헤더, 및 데이터 구조를 포함하는 컴패니언 파일(companion) 파일 모두를 생성한다.

[0077] 위에서 논의된 바와 같이, 일 실시예에서는, 데이터가 인코딩되고 나면(단계(1000)), 인코딩된 데이터 패킷이 다수의 고정된 크기의 데이터 유닛으로 분리된다(단계(1005)). 또한, 인코딩된 데이터와 마찬가지로, 데이터 헤더 및 데이터 구조도 인코딩된 데이터 패킷을 분리하는 데 사용되는 것과 동일한 고정된 크기의 다수의 데이터 유닛으로 분리된다(단계(1005)). 이 정보를 고정된 길이의 데이터 유닛으로 분리하는 단계(1005)는 피어로

하여금 데이터 전송 동작에 앞서 메모리 블록을 미리 할당할 수 있게 하고, 이에 의해 데이터 전송 프로세스 동안 계산적으로 고가인 메모리 할당 동작을 피할 수 있다. 또한, 더욱 작은 데이터 유닛을 사용하면, 각각의 피어에 의해 소비되는 정확한 양의 밴드폭을 통해 데이터를 저장하는 피어 또는 클라이언트에 의한 미세한 제어가 데이터 전송 동작 기간 동안 클라이언트 데이터 유닛 요청을 만족하게 할 수 있다.

[0078] 인코딩된 데이터, 데이터 헤더 및 데이터 구조를 더 작은 데이터 유닛으로 분리하는 단계(1005) 외에, 만약 소거 회복 코딩이 사용되면, 서빙 피어의 신뢰도가 본래 낮은 통상적인 P2P 환경에서 중복성을 증가시키기 위해 추가적인 계층의 코딩이 사용된다. 특히, 상술한 바와 같이, 일 실시예에서는, 만약 소거 회복 코딩이 데이터 파일에 대하여 적절한 것으로 판정되면, 데이터 유닛은 다수의 데이터 블록으로 더욱 분할되고, 소거 회복 코딩 프로세스 단계(1010)가 파일을 인코딩하는 데 사용된다.

[0079] 이러한 코딩 단계(1010)를 사용하면, 하나 이상의 피어가 특정한 데이터 유닛을 재구성하는 데 필요한 데이터 블록을 가지면서, 피어들 중 어느 피어가 필요한 데이터를 포함하는지 여부를 식별하기 위한 클라이언트에 대한 요구를 간단하게 하는 것을 보장한다. 또한, 일 실시예에서, 각각의 서빙 피어(220)에 의해 사용되는 소거 회복 코딩 키는 서버(200)에 의해 각각의 피어에 자동으로 할당된다. 그러나, 다른 실시예에서는, 각각의 서빙 피어(220)가 소거 회복 코딩 키를 간단히 랜덤하게 선택한다. 그리고, 이 키는, 클라이언트가 각각의 피어(220)에 처음으로 접촉할 때, 클라이언트(210)에 의해 리트리브된다.

[0080] 데이터 파일이 초기에 인코딩되고(단계(1000)), 데이터 유닛으로 분리되고(단계(1005)), 가능하다면 소거 코딩되었으면(단계(1010)), 결과적인 데이터 유닛 또는 데이터 블록이 여러 피어(220)에 분산된다(단계(1015)). 이러한 분산(단계(1015))은, 인코딩된 데이터의 블록 또는 패킷이 단순히 전체적 또는 부분적으로 다수의 피어들에 제공되고, 데이터를 리트리브하고 싶어하는 클라이언트에 의해 호출될 때의 장래의 데이터 전송을 위해 그 다수의 피어들에 캐싱되거나(cached) 또는 저장된다는 점에서, 계획적일 수 있다.

[0081] 데이터가 서빙 피어(220)로 분산되었으면(단계(1015)), 클라이언트(210)가 저장으로부터 이 데이터를 리트리브하고 싶어하는 경우, 클라이언트(210)는 이 피어에 데이터 요청을 시작하려고 한다. 또한, 위에서 언급된 바와 같이, 서버(200)는 클라이언트(210)에 데이터를 전송할 목적으로 피어(220)로서 동작할 수도 있다.

[0082] 이때, 클라이언트(210)는 이용 가능한 서빙 피어(220) 리스트를 우선 리트리브함으로써 데이터 전송 세션을 시작한다. 이 리스트는 서버(200), 또는 피어(220)들 중 하나의 피어로부터 직접, 또는 가능한 서빙 피어를 식별하기 위한 통상적인 DHT(distributed hash table) 방법을 사용함으로써 리트리브된다. 클라이언트(1010)가 피어 리스트를 검색하였으면, 클라이언트는 각각의 서빙 피어(220)에 접속하고, 각각의 피어로부터 이용 가능한 파일 리스트를 리트리브한다(단계(1025)). 클라이언트(210)가 각각의 피어(220)의 이용 가능한 파일 리스트를 리트리브하였으면, 클라이언트는 클라이언트와 하나 이상의 피어 간의 네트워크 접속을 통해 그 하나 이상의 피어로부터 그 정보에 대응하는 데이터 유닛을 요청함으로써 그 하나 이상의 피어로부터 전송될 데이터의 데이터 구조 및 데이터 헤더를 리트리브한다(단계(1035)).

[0083] 데이터 헤더는 일반적으로 데이터를 설명하는 포괄적인 정보, 예를 들어 데이터 내의 채널의 수, 각각의 채널의 속성 및 특성(오디오 샘플링 속도, 비디오 해상도/프레임 속도), 사용된 코덱, 미디어의 저작자/저작권 보유자 등을 포함한다. 결과적으로, 데이터 전송 세션의 시작 시에 데이터 헤더를 리트리브하면, 클라이언트(220)는 이어서 수신되는 패킷을 디코딩(단계(1065))하기 위하여 필요한 툴을, 데이터 전송 세션 동안 이 패킷을 수신하는 것에 앞서 셋업하거나 초기화할 수 있게 된다(단계(1040)).

[0084] 또한, 특정한 데이터의 데이터 구조를 리트리브(단계(1035))한 후, 클라이언트는 그 데이터 구조를 분석하고, 데이터 전송 프로세스 동안 요청될 필요가 있는 전송된 데이터의 데이터 유닛의 데이터 유닛 ID를 계산한다(단계(1045)). 그 후, 클라이언트(210)는 하나 이상의 피어(220)로부터 그 데이터 유닛을 하나씩 요청한다(단계(1050)).

[0085] 마지막으로, 특정한 데이터 패킷을 구성하는 모든 데이터 유닛이 클라이언트(210) 요청에 따라 리트리브되었으면(단계(1050)), 그 데이터 패킷들은 원본 데이터 패킷으로 재조립된다(단계(1055)). 그 후, 재조립된 데이터 패킷들은 디코딩되고(단계(1060)), 그 후 클라이언트(210)에서 복구될 수 있다(단계(1065)).

[0086] **3.0 P2P 저장 : 정책 및 설계 전략**

[0087] P2P 네트워크에 저장될 파일 크기에 기초하여 단편들의 ERC 수를 조정하는 것 외에, 효율성 또한 향상될 수 있

다. 여기에 기재되는 적응 코딩 저장 시스템의 여러 가지 실시예는 이하에 기재되는 바와 같은 소정의 전략을 사용함으로써 저장 효율성을 향상시키도록 설계된다. 이 전략은 적응 코딩 저장 시스템과 함께 사용되거나 아니면 임의의 P2P 네트워크에서 사용될 수 있다.

[0088] 3.1 P2P 저장 비용

[0089] 이 섹션에서는, P2P 네트워크에 파일을 저장하는 것이 "six nine" 신뢰도의 서버에 파일을 직접 저장하는 것과 비교된다. P2P 솔루션은 서버 밴드폭과 비용은 감소시키지만, 피어가 파일을 P2P 저장으로 분산하기 위하여 보다 많은 밴드폭을 소비할 것을 요구함을 알 수 있다. P2P 솔루션에서는 네트워크 밴드폭의 사용이 전반적으로 증가된다. 클라이언트의 업로드 밴드폭의 증가는 P2P 저장 시스템의 비용으로 고려될 수 있다. 상이한 피어 신뢰도 및 파일 크기에 대한 이러한 비용이 표 1에 기재되어 있다.

표 1

P2P에서 증가되는 밴드폭 사용 비용

[0090]

	파일 크기				
신뢰도	10KB	100KB	1MB	10MB	100MB
10%	332.9	79.1	29.5	16.5	12.5
50%	51.0	12.11	4.34	2.23	1.56
99%	9.4	1.87	0.65	0.22	0.09

[0091] 피어 신뢰도가 높고 파일 크기가 크면, P2P 저장을 사용하는 비용이 작음을 알 수 있다. 예를 들면, 99%의 신뢰도를 갖는 피어에 100MB의 파일을 저장하면 9%의 비용만이 발생한다. 그러나, 피어 신뢰도가 낮고 파일 크기가 작으면, 비용이 상당할 수 있다.

[0092] 3.2 P2P 저장 정책

[0093] 표 1로부터, P2P 저장 클라우드를 사용하는 것에 관한 이하의 정책을 도출할 수 있다.

[0094] a) 대용량 파일을 저장하기 위해서는 신뢰도 낮은 피어를 사용하고, 소용량 파일을 저장하기 위해서는 신뢰도 높은 피어를 사용해야 한다. 만약 신뢰도 낮은 피어에는 대용량 파일을 할당하고, 신뢰도 높은 피어에는 소용량 파일을 할당하면, P2P 시스템에 대한 비용은 더욱 감소될 것이다.

[0095] b) 정적 파일을 저장하기 위해서는 신뢰도 낮은 피어를 사용하고, 동적 파일을 저장하기 위해서는 신뢰도 높은 피어를 사용해야 한다. 변화되지 않는 파일을 정적 파일이라 하고, 끊임없이 변화되는 파일을 동적 파일이라고 칭한다. 복수의 소용량 정적 파일은 대용량 정적 파일로 번들링되고, P2P 저장 클라우드에 저장될 수 있다. 동적 파일에 대해서는, 단일의 파일의 변화는 전체 조합된 파일이 업데이트될 것을 필요로 하기 때문에, 동일한 전략이 효과적이지 않다.

[0096] 이러한 정책의 결과는, 만약 애플리케이션의 상태, 피어 상태 정보 등을 저장하기 위하여 P2P 네트워크를 사용하면, 정보를 네트워크 중 신뢰도가 가장 높은 피어로 향하게 해야 한다는 것이다. 만약 애플리케이션의 상태를 포함하는 파일이 매우 신뢰도 높은 피어(본질적으로, 매우 신뢰도 높은 피어는 확장된 P2P 네트워크의 코어를 구성하는 서브네트워크를 형성함)에만 배치되도록 제한하면, 복사율 및 상태 파일의 업데이트 비용을 크게 감소시킬 수 있고, 효율성을 향상시킬수 있다.

[0097] c) 신뢰도 낮은 피어에는 덜 분산되도록 하고, 신뢰도 높은 피어에는 더욱 분산되도록 하여야 한다.

[0098] d) 소용량의 파일에는 높은 분산 비용이 할당되고, 대용량의 파일에는 낮은 분산 비용이 할당되어야 한다.

[0099] 정책 c) 및 d)는 피어가 P2P 저장 클라우드에 콘텐츠를 분산시키고, 다른 피어들을 위해 콘텐츠를 저장시킬 수 있는 P2P 백업 및 리트리브 애플리케이션에 관한 것이다. 균형잡힌 P2P 저장 네트워크는 각각의 피어로 하여금 그 기여도 및 이득의 균형을 잡게 하여야 한다. 이전 내용에서는, 밴드폭이 P2P 저장 애플리케이션의 주요한

리소스라고 지적하였다. 피어가 다른 피어들을 위하여 수신하고 저장하는 코딩된 단편의 양을 그 피어의 기여도라고 하자. 피어가 P2P 클라우드에 분산하는 콘텐츠의 양을 그 피어의 이득이라고 하자. 신뢰도가 낮으면 데이터 저장을 더욱 중복되게 하는 것을 고려하여, 신뢰도 낮은 피어는 더 적게 분산하도록 제재(punish)를 가하고, 신뢰도 높은 피어는 더 많이 분산하도록 보상(reward)을 하여야 한다. 이러한 정책은 사용자가 P2P 애플리케이션을 온라인으로 유지하는 것을 장려하여, 이에 의해 P2P 네트워크의 전반적인 신뢰도를 향상시키고, 요구되는 복사율을 감소시키므로, P2P 경제학에서 긍정적인 이점을 가질 수 있다.

[0100] 또한, 소용량 파일의 분산에는 높은 분산 비용을 할당하여, 해당 피어가 그에 비례하여 더 많이 기여하도록 요구함으로써 소용량 파일의 분산에 제재를 가하고, 대용량 파일의 분산에는 낮은 분산 비용을 할당하여, 해당 피어가 그에 비례하여 더 적게 기여하도록 함으로써 대용량 파일의 분산에 보상을 할 수 있다. 결과적으로, P2P 백업 애플리케이션은 백업 빈도를 최소화하도록 설계되어야 한다. 파일이 변화된 직후에 파일을 즉시 업데이트하는 대신에, 복수의 변화들을 대용량 파일로 번들링하고, P2P 저장 클라우드에 단 한 번씩만, 예를 들어 매일 밤마다 업데이트하는 것을 고려할 수 있다.

[0101] 상술한 정책과 관련하여 설계되는 적응 코딩 저장 시스템 및 방법의 일 실시예가 도 11에 도시되어 있다. 프로세스 단계(1102)에 도시된 바와 같이, 분산형 또는 P2P 네트워크의 각각의 피어의 신뢰도가 판정된다. 분산되거나 저장될 파일이 입력된다(프로세스 단계(1104)). 파일의 크기가 평가되고(프로세스 단계(1106)), 수학식 7의 예상 저장 밴드폭에 기초하여 분산 비용이 파일에 할당된다(프로세스 단계(1108)). 만약 파일이 대용량 파일이면, 보다 높은 분산 비용이 할당될 수 있다. 만약 파일이 소용량이면, 파일에는 보다 작은 분산 비용이 할당될 수 있다. 파일의 크기에 기초하여, 적응 코딩 저장 시스템이 파일을 저장하기에 적절한 신뢰도를 갖는 피어를 선택할 것이다(프로세스 단계(1110)). 즉, 소정의 문턱치보다 낮은 신뢰도를 갖는 피어는 대용량 파일을 저장하고 분산하는 데 사용되고, 소정의 문턱치보다 높은 신뢰도를 갖는 피어는 소용량 파일을 저장하고 분산하는 데 사용된다.

[0102] 상술한 정책과 관련하여 설계되는 적응 코딩 저장 시스템 및 방법의 다른 실시예가 도 12에 도시되어 있다. 프로세스 단계(1202)에 도시된 바와 같이, 분산형 또는 P2P 네트워크의 각각의 피어의 신뢰도가 판정된다. 분산되거나 저장될 파일이 입력된다(프로세스 동작(1204)). 입력된 파일이 정적인지 아니면 동적인지를 판정하기 위하여, 이전에 저장되었던 동일한 파일과 입력된 파일이 비교된다(프로세스 단계(1206)). 파일이 처음으로 보관될 때, 그 파일은 동적이라고 가정한다. 파일에 빈번한 변화가 관찰되면, 그 파일은 여전히 동적인 것으로 지정된다. 만약 연장된 기간동안 파일이 변화되지 않음이 관찰되면, 파일은 정적인 것으로 지정된다. 동적 파일은 신뢰도가 매우 높은 피어에 저장된다(프로세스 단계(1210)). (따라서, 처음에는 파일이 서버 또는 신뢰도가 매우 높은 피어에 저장될 것이다.) 파일이 변화되지 않고 정적인 것으로 됨이 관찰되면, 이 정적 파일은 재분산되어 신뢰도가 낮은 피어에 저장될 것이다.

[0103] 단, 분산형 또는 피어 투 피어 네트워크의 전반적인 효율성 및 신뢰도를 증가시키기 위하여 도 11 및 도 12에 도시된 실시예는 단독으로 또는 조합되어 사용될 수 있다.

[0104] ### 3.3 서버 컴포넌트 지원을 이용한 P2P 저장

[0105] 만약 서버 컴포넌트가 P2P 네트워크의 보완에 사용되면, 대용량의 정적 파일을 위해서는 P2P 저장을 사용하고, 소용량의 동적 파일을 위해서는 서버를 사용할 수 있다. 서버 리소스의 대부분을 소비하는 것은 대용량 파일이기 때문에, P2P 저장은 서버를 잘 보완한다.

[0106] 도 13에 도시된 바와 같이, 적응 코딩 저장 시스템 및 프로세스의 일 실시예는 서버 지원을 이용한 P2P 백업을 사용한다. 도 13에 도시된 바와 같이, 네트워크의 동적 파일이 서버에 백업된다(프로세스 단계(1302)). 그 후, 클라이언트 및/또는 서버는 동적 파일이 더 이상 변화되지 않고, 정적 파일로 변하고 있음을 자동으로 감지할 수 있다(프로세스 단계(1304, 1306)). 그 후, 이 감지된 정적 파일은 프로세스 단계(1308)에 도시된 바와 같이 모두 함께 대용량 파일로 번들링되고, ERC를 이용하여 P2P 저장 클라우드로 분산될 수 있다(프로세스 단계(1310)). 이는 P2P 클라우드에 저장되는 파일의 크기를 효율적으로 증가시킨다. 이는 다수의 단편들의 ERC와 조합되어, 효율성을 향상시킬 수 있다.

[0107] 도 13에 도시된 실시예는 단독으로 사용될 수도 있고, 또는 분산형 또는 피어 투 피어 네트워크의 효율성 및 신뢰도를 전반적으로 증가시키기 위하여 도 11 및 도 12에 도시된 실시예와 조합하여 사용될 수도 있다. 또한, 이 실시예는 소거 회복 코딩을 이용하는 것뿐만 아니라 소거 회복 코딩 없이도 사용될 수 있음을 주의한다.

[0108] 추가적인 혼합 실시예를 형성하기 위하여 상술한 대안적인 실시예의 일부 또는 전부가 원하는 임의의 조합으로 사용될 수 있음을 주의하여야 한다. 비록 대상 문제가 구조적인 특징 및/또는 방법론적인 동작에 특정된 언어로 기술되어 있지만, 첨부되는 청구 범위에서 규정되는 대상 문제는 상술된 특정한 특징 또는 동작에 반드시 한정될 필요가 없음을 이해할 것이다. 오히려, 상술한 특정 특징 및 단계는 청구 범위를 구현하는 예시적인 형태로서 개시되어 있다.

도면의 간단한 설명

[0010] 적응 코딩 저장 시스템의 구체적인 특징, 양태 및 이점은 이하 명세서, 첨부된 청구항 및 첨부된 도면을 고려하면 더욱 잘 이해될 것이다.

[0011] 도 1은 여기서 설명되는 적응 코딩 저장 시스템 및 방법을 구현하는 예시적인 시스템을 구성하는 범용 컴퓨팅 장치를 도시하는 일반적인 시스템도.

[0012] 도 2는 여기서 설명되는 적응 코딩 저장 시스템 및 방법과 함께 사용될 수 있는 예시적인 피어 투 피어(P2P) 네트워크를 도시하는 도면.

[0013] 도 3은 10^{-6}의 원하는 신뢰도를 실현하기 위한 정보 저장 피어들의 수를 도시하는 그래프.

[0014] 도 4는 피어 신뢰도 및 원하는 복사율(replication ratio)을 도시하는 그래프.

[0015] 도 5는 소거 회복 코딩을 사용하여 10^{-6}의 원하는 신뢰도를 실현하기 위하여 필요한 정보 저장 피어들의 수를 도시하는 도면.

[0016] 도 6은 P2P 네트워크에서의 정보 저장을 위한 단편들의 ERC 수 및 연관된 적절한 파일 크기를 도시하는 그래프.

[0017] 도 7은 적응 ERC를 이용한 P2P 구성의 피어와 고정 ERC(피어 신뢰도 = 50%)를 이용한 P2P 구성의 피어 간의 밴드폭 사용을 도시하는 그래프.

[0018] 도 8은 적응 ERC를 이용한 P2P 구성의 피어와 고정 ERC(피어 신뢰도 = 99%)를 이용한 P2P 구성의 피어 간의 밴드폭 사용을 도시하는 그래프.

[0019] 도 9는 적응 코딩 저장 프로세스의 일 실시예를 도시하는 도면.

[0020] 도 10은 적응 코딩 저장 기술이 P2P 네트워크에서 사용되는 방식을 도시하는 예시적인 동작 흐름도.

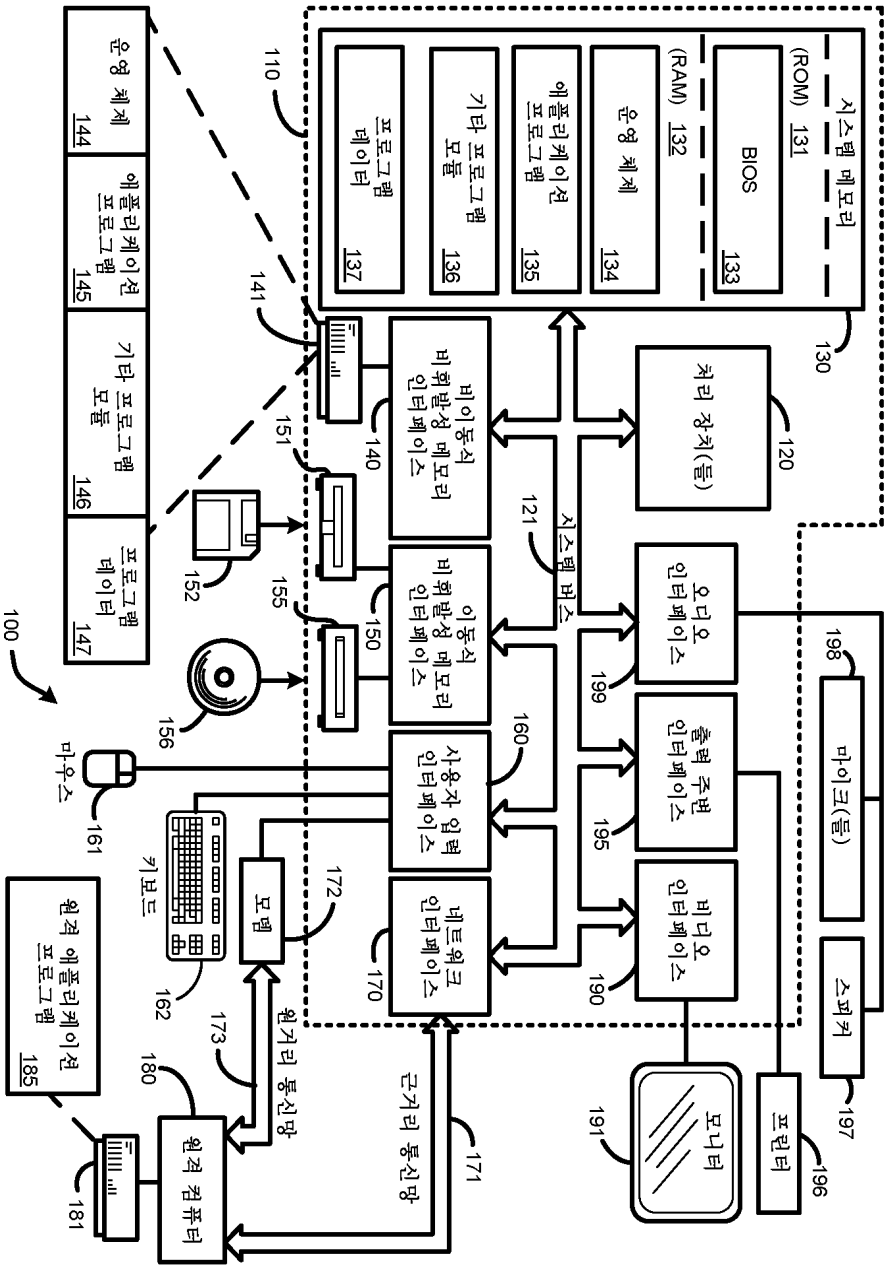
[0021] 도 11은 P2P 네트워크의 저장 효율을 최적화하기 위한 프로시저를 구현하는 적응 코딩 저장 시스템 및 프로세스의 일 실시예를 도시하는 도면.

[0022] 도 12는 P2P 시스템의 저장 효율을 최적화하기 위한 프로시저를 구현하는 적응 코딩 저장 시스템 및 방법의 다른 실시예를 도시하는 도면.

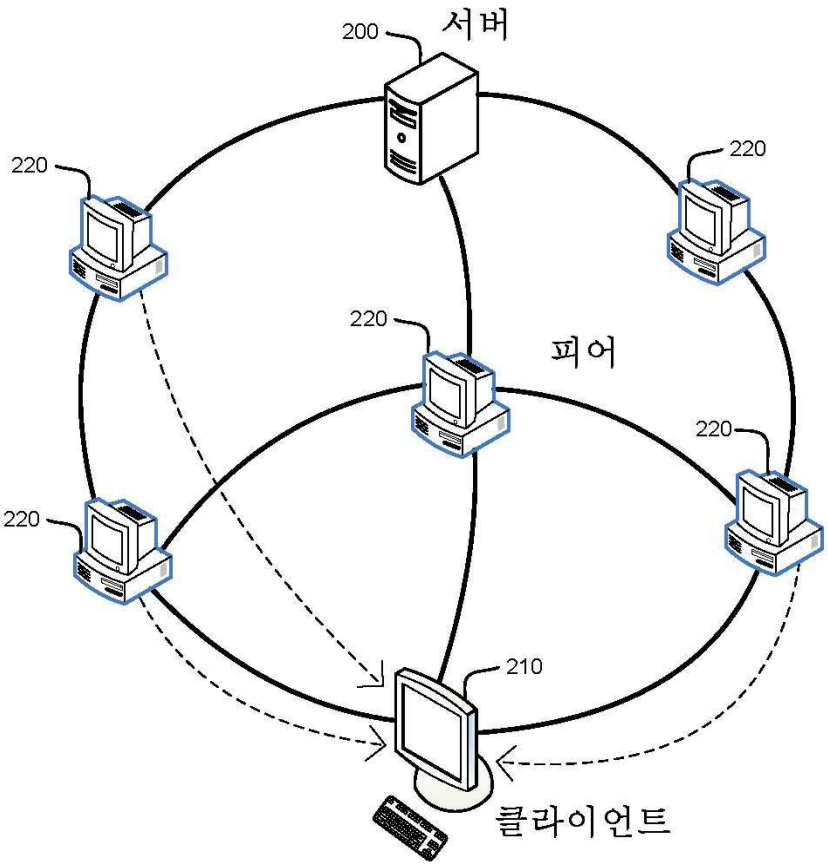
[0023] 도 13은 서버 지원에 의한 P2P 백업을 사용하는 적응 코딩 저장 시스템 및 방법의 일 실시예를 도시하는 도면.

도면

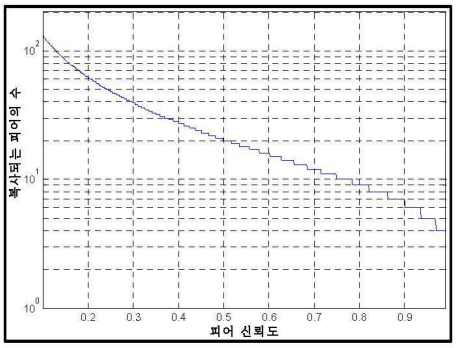
도면1


시스템 메모리
(ROM) 131
BIOS 133
(RAM) 132
운영 체제 134
애플리케이션 프로그램 135
기타 프로그램 모듈 136
프로그램 데이터 137
130
110
처리 장치(들) 120
121
시스템 버스
비이동식 비휘발성 메모리 인터페이스 140
이동식 비휘발성 메모리 인터페이스 150
오디오 인터페이스 199
출력 주변 인터페이스 195
사용자 입력 인터페이스 160
네트워크 인터페이스 170
비디오 인터페이스 190
마이크(들) 198
스피커 197
모니터 191
프린터 196
141
151
155
152
156
100
마우스 161
키보드 162
모뎀 172
173
원거리 통신망
171
원격 컴퓨터 180
원격 애플리케이션 프로그램 185
181
운영 체제 144
애플리케이션 프로그램 145
기타 프로그램 모듈 146
프로그램 데이터 147

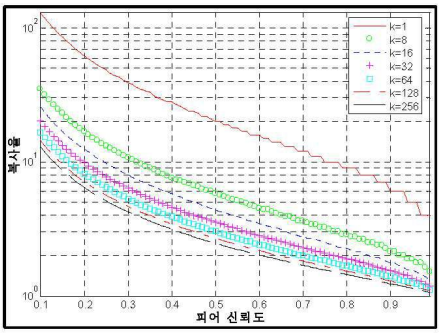
도면2


200
서버
220
220
220
피어
220
220
210
클라이언트

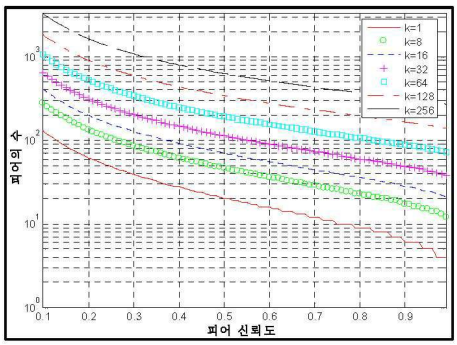
도면3


복사되는 파어의 수
피어 신뢰도
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

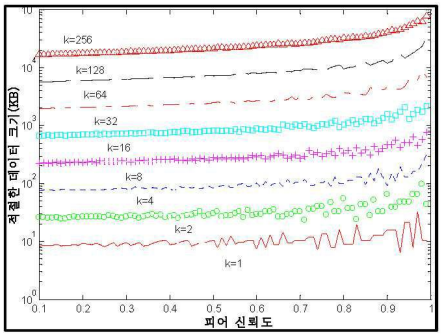
도면4


k=1
k=8
k=16
k=32
k=64
k=128
k=256
복사율
피어 신뢰도

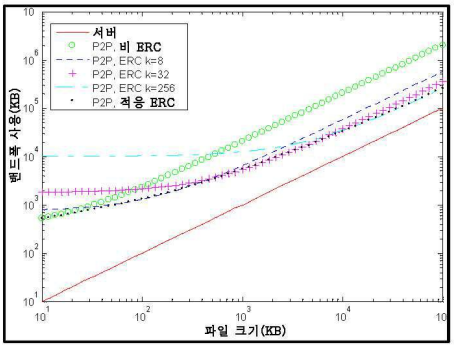
도면5


k=1
k=8
k=16
k=32
k=64
k=128
k=256
피어의 수
피어 신뢰도

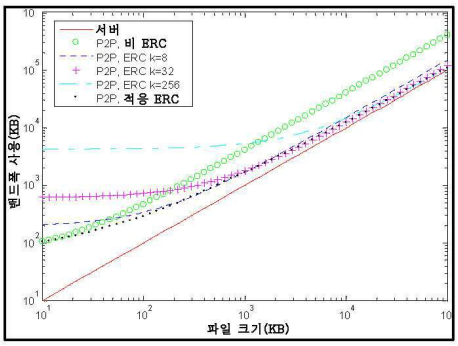
도면6


k=256
k=128
k=64
k=32
k=16
k=8
k=4
k=2
k=1
제공한 데이터 크기(KB)
피어 신뢰도

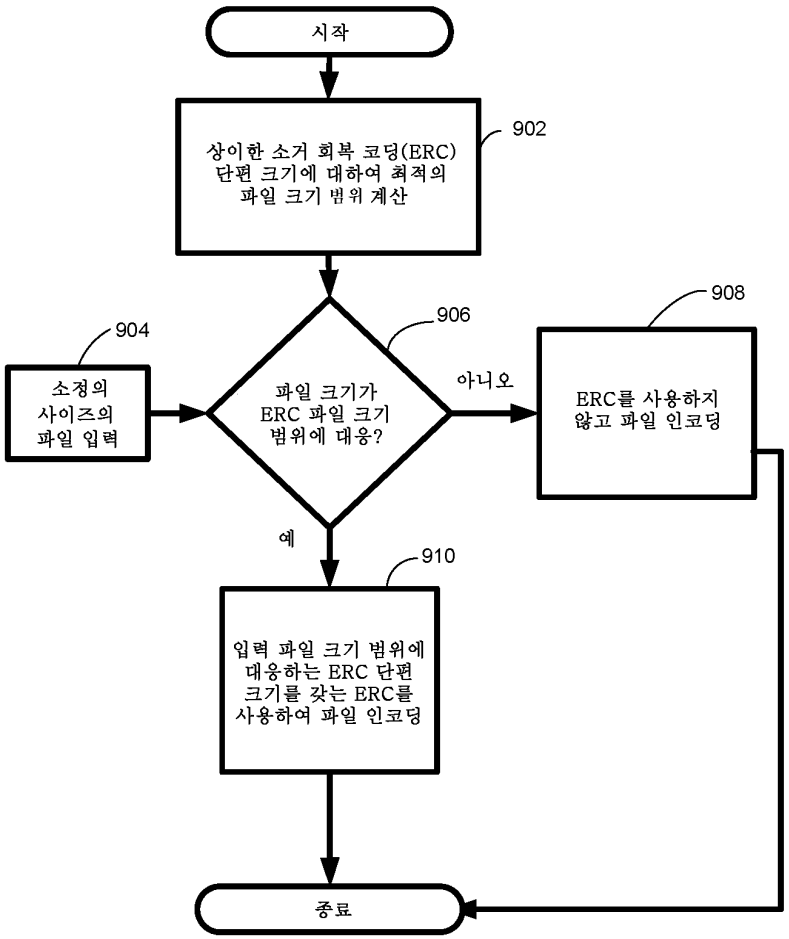
도면7


서버
P2P, 비 ERC
P2P, ERC k=8
P2P, ERC k=32
P2P, ERC k=256
P2P, 적응 ERC
벤드폭 사용(KB)
파일 크기(KB)

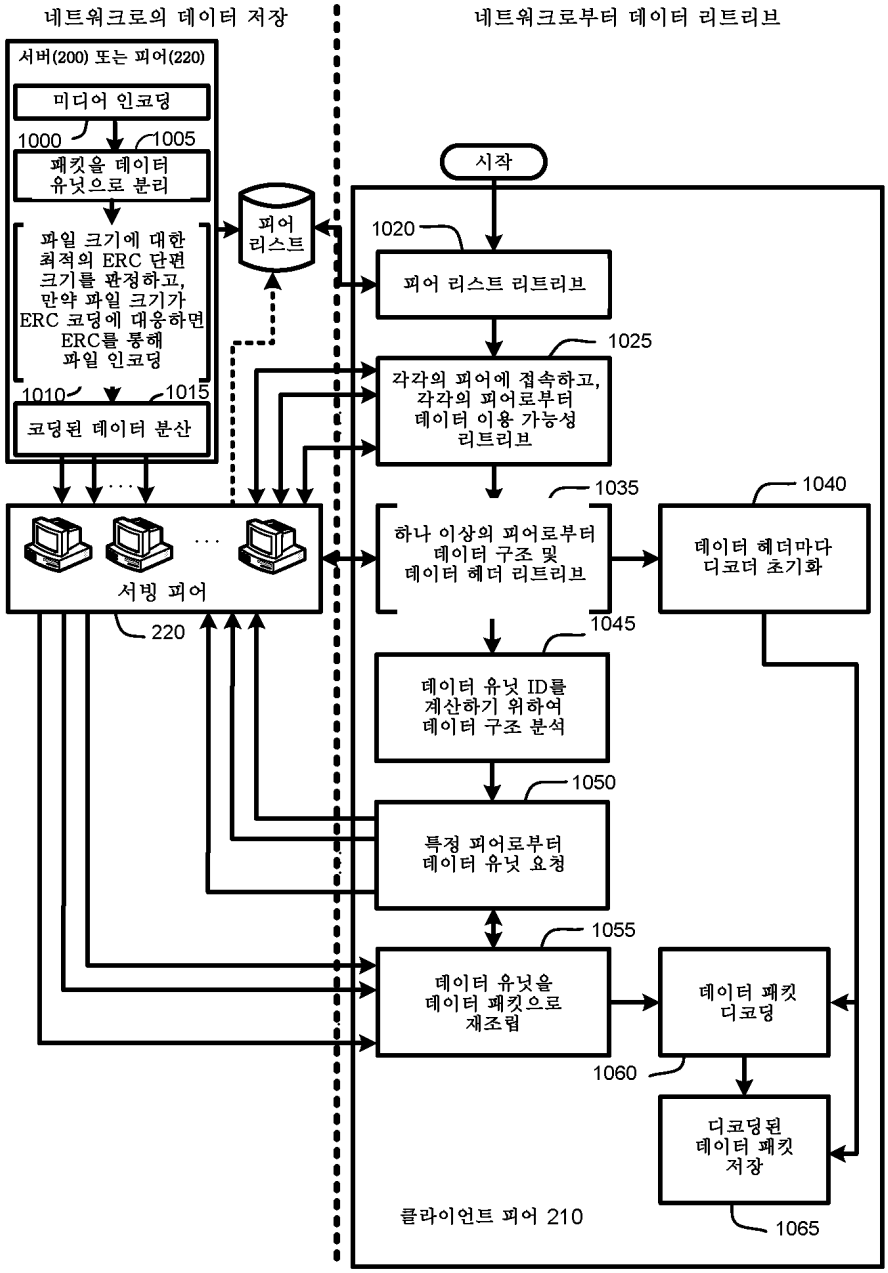
도면8


서버
P2P, 비 ERC
P2P, ERC k=8
P2P, ERC k=32
P2P, ERC k=256
P2P, 적응 ERC
벤드폭 사용(KB)
파일 크기(KB)

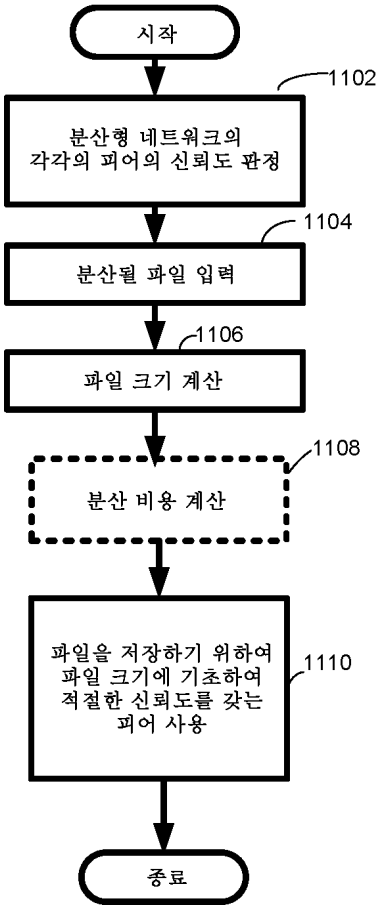
도면9


시작
상이한 소거 회복 코딩(ERC) 단편 크기에 대하여 최적의 파일 크기 범위 계산
902
904
소정의 사이즈의 파일 입력
906
파일 크기가 ERC 파일 크기 범위에 대응?
아니오
908
ERC를 사용하지 않고 파일 인코딩
예
910
입력 파일 크기 범위에 대응하는 ERC 단편 크기를 갖는 ERC를 사용하여 파일 인코딩
종료

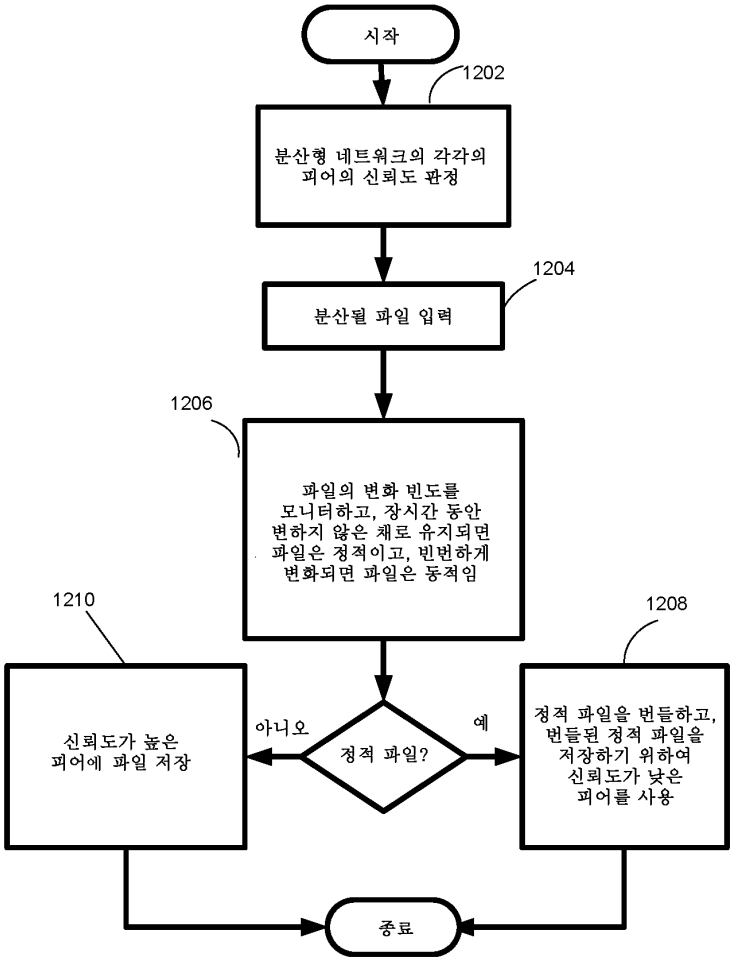
도면10


네트워크로의 데이터 저장
네트워크로부터 데이터 리트리브
서버(200) 또는 피어(220)
미디어 인코딩
1000
1005
패킷을 데이터 유닛으로 분리
파일 크기에 대한 최적의 ERC 단편 크기를 판정하고, 만약 파일 크기가 ERC 코딩에 대응하면 ERC를 통해 파일 인코딩
1010
1015
코딩된 데이터 분산
피어 리스트
서빙 피어
220
시작
1020
피어 리스트 리트리브
1025
각각의 피어에 접속하고, 각각의 피어로부터 데이터 이용 가능성 리트리브
1035
하나 이상의 피어로부터 데이터 구조 및 데이터 헤더 리트리브
1040
데이터 헤더마다 디코더 초기화
1045
데이터 유닛 ID를 계산하기 위하여 데이터 구조 분석
1050
특정 피어로부터 데이터 유닛 요청
1055
데이터 유닛을 데이터 패킷으로 재조립
데이터 패킷 디코딩
1060
디코딩된 데이터 패킷 저장
1065
클라이언트 피어 210

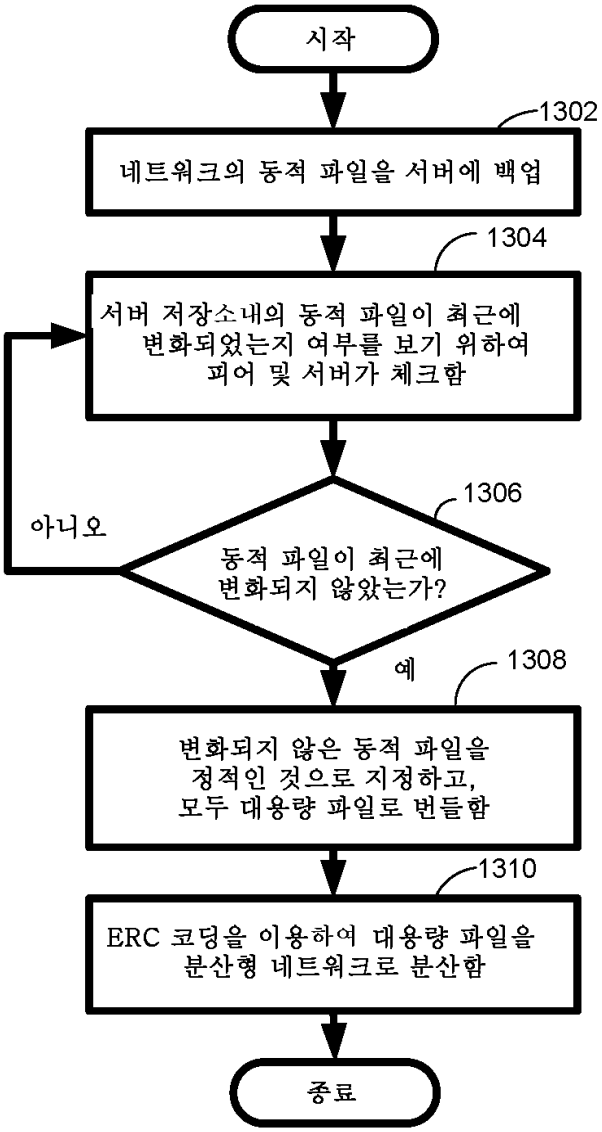
도면11


시작
1102
분산형 네트워크의 각각의 피어의 신뢰도 판정
1104
분산될 파일 입력
1106
파일 크기 계산
1108
분산 비용 계산
파일을 저장하기 위하여 파일 크기에 기초하여 적절한 신뢰도를 갖는 피어 사용
1110
종료

도면12


시작
1202
분산형 네트워크의 각각의 피어의 신뢰도 판정
1204
분산될 파일 입력
1206
파일의 변화 빈도를 모니터하고, 장시간 동안 변하지 않은 채로 유지되면 파일은 정적이고, 빈번하게 변화되면 파일은 동적임
1210
신뢰도가 높은 피어에 파일 저장
아니오
정적 파일?
예
1208
정적 파일을 번들하고, 번들된 정적 파일을 저장하기 위하여 신뢰도가 낮은 피어를 사용
종료

도면13


시작
1302
네트워크의 동적 파일을 서버에 백업
1304
서버 저장소내의 동적 파일이 최근에 변화되었는지 여부를 보기 위하여 피어 및 서버가 체크함
1306
동적 파일이 최근에 변화되지 않았는가?
아니오
예
1308
변화되지 않은 동적 파일을 정적인 것으로 지정하고, 모두 대용량 파일로 번들함
1310
ERC 코딩을 이용하여 대용량 파일을 분산형 네트워크로 분산함
종료