(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0040867
(43) 공개일자 2020년04월20일

(51) 국제특허분류(Int. Cl.)
H04W 12/00 (2019.01) *H04W 4/60* (2018.01)
H04W 76/14 (2018.01) *H04W 76/40* (2018.01)
H04W 84/18 (2009.01)
(52) CPC특허분류
H04W 12/002 (2019.01)
H04W 12/001 (2019.01)
(21) 출원번호 10-2020-7008574
(22) 출원일자(국제) 2018년08월09일
심사청구일자 없음
(85) 번역문제출일자 2020년03월24일
(86) 국제출원번호 PCT/CA2018/000151
(87) 국제공개번호 WO 2019/036791
국제공개일자 2019년02월28일
(30) 우선권주장
62/550,471 2017년08월25일 미국(US)

(71) 출원인
레프트 테크놀로지스 인코포레이티드
캐나다 브이2엑스 9이7 브리티쉬 컬럼비아 메이플 릿지 스튜어트 크레센트 20000 #4
(72) 발명자
언스트 제이슨 브루스
캐나다 브이3비 0에이3 브리티쉬 컬럼비아 코퀴틀럼 파인트리 웨이 1188, 1506
왕 쩌후아
캐나다 브이3비 5케이5 브리티쉬 컬럼비아 포트 코퀴틀럼 오리올 플레이스 1268
(74) 대리인
특허법인(유한)케이비케이

전체 청구항 수 : 총 17 항

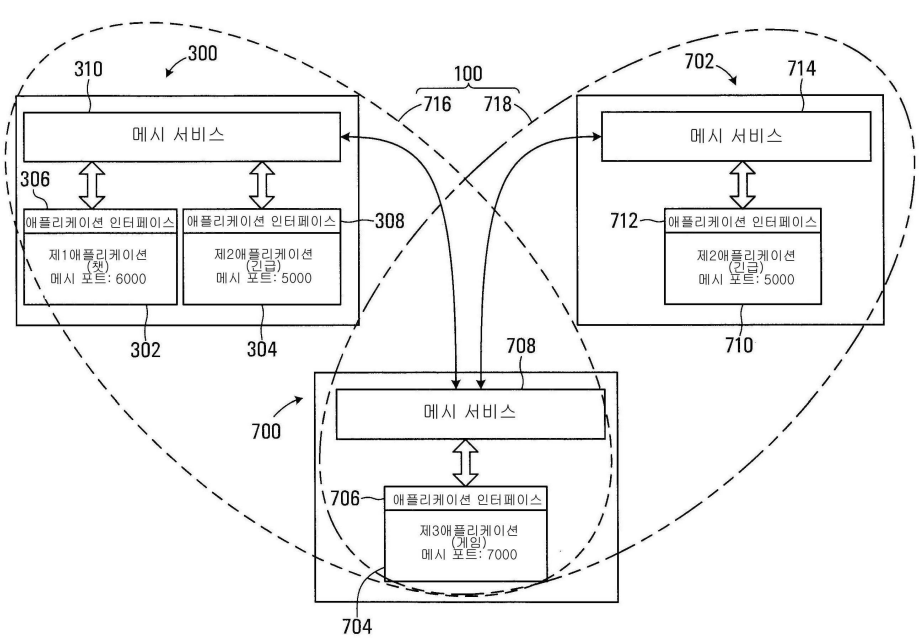
(54) 발명의 명칭 **메시 포트가 있는 메시 통신 네트워크**

(57) 요 약

복수의 디바이스 사이에 설정된 메시 네트워크를 통해 통신하는 방법이 개시된다. 각각의 디바이스는 무선 라디오를 가지며, 이 방법은 각 디바이스에서 메시 서비스를 론칭하는 단계를 포함하며, 상기 메시 서비스는 디바이스의 프로세서 회로가 메시 네트워크를 통해 디바이스들 간의 통신을 위해 무선 라디오를 컨트롤하기 위한 기능

(뒷면에 계속)

대 표 도



을 제공하도록 동작 가능하다. 각각의 디바이스는 디바이스상에서 실행되는 적어도 하나의 애플리케이션을 가지며, 상기 적어도 하나의 애플리케이션은 메시 포트와 관련되고, 상기 메시 포트는 복수의 디바이스들에서 적어도 일부의 디바이스상에서 실행되는 특정 애플리케이션의 인스턴스와 연관된 것으로 데이터 전송을 지정하는데 사용되며, 상기 적어도 하나의 애플리케이션 및 각각의 디바이스상의 메시 서비스는 데이터 통신하고 있다. 이 방법은 또한 특정 메시 포트를 통한 통신을 위해 메시 네트워크에 대한 액세스를 제공하기 위해 메시 서비스를 요청하는 디바이스에서 실행되는 특정 애플리케이션에 응답하여, 메시 서비스가 특정 애플리케이션이 특정 메시 포트에 통신을 위해 승인되는지 여부를 결정하게 하고, 특정 애플리케이션이 승인된 경우, 애플리케이션으로부터의 요청을 처리하여 메시 네트워크를 통해 특정 메시 포트와 통신하고 특정 메시 포트와 관련된 데이터 전송을 특정 애플리케이션으로 포워딩하며, 특정 애플리케이션이 승인되지 않은 경우, 애플리케이션으로부터의 요청을 거절하여 메시 네트워크를 통해 특정 메시 포트에서 통신하고 상기 특정 메시 포트와 관련된 데이터 전송에 대한 특정 애플리케이션의 액세스를 막는다.

(52) CPC특허분류
H04W 12/0051 (2019.01)
H04W 4/60 (2018.02)
H04W 76/14 (2018.02)
H04W 76/40 (2018.02)
H04W 84/18 (2013.01)

명 세 서

청구범위

청구항 1

각각이 무선 라디오를 갖는 복수의 디바이스들 사이에 확립된 메시 네트워크를 통해 통신하는 방법으로서, 상기 방법은:

상기 디바이스의 프로세스 회로가 메시 네트워크를 통한 디바이스들 간에 통신을 위해 무선 라디오를 컨트롤하기 위한 기능을 제공하도록 동작할 수 있는 메시 서비스를 각 디바이스 상에 론칭하는 단계;

특정 메시 포트를 통한 통신을 위해 상기 메시 네트워크에 액세스를 제공하기 위해 상기 메시 서비스를 요청하는 디바이스에 실행되는 특정 애플리케이션에 응답하여:

메시 서비스로 하여금 특정 애플리케이션이 특정 메시 포트상에서의 통신을 위한 승인되었는지를 결정하는 단계;

특정 애플리케이션이 승인된 경우, 메시 네트워크를 통해 특정 메시 포트에 통신하기 위해 애플리케이션으로부터의 요청을 처리하고 상기 특정 메시 포트와 관련된 데이터 전송을 상기 특정 애플리케이션으로 전달하는 단계; 및

특정 애플리케이션이 승인되지 않은 경우, 메시 네트워크를 통해 특정 메시 포트에 통신하기 위해 애플리케이션으로부터의 요청을 거절하고 상기 특정 메시 포트와 관련된 데이터 전송에 대한 특정 애플리케이션의 액세스를 방지하는 단계를 포함하고,

각각의 디바이스는 상기 디바이스 상에서 실행되는 적어도 하나의 애플리케이션을 가지며, 상기 적어도 하나의 애플리케이션은 메시 포트와 관련되고, 상기 메시 포트는 복수의 디바이스들 내의 디바이스들 중 적어도 일부에 실행되는 특정 애플리케이션의 인스턴스와 관련된 것으로 데이터 전송을 지시하는데 사용되며, 상기 적어도 하나의 애플리케이션 및 각 디바이스 상의 메시 서비스는 데이터 통신하는 메시 네트워크를 통해 통신하는 방법.

청구항 2

제 1 항에 있어서,

메시 서비스를 론칭하는 단계는 디바이스에서 운영 시스템을 부팅할 때 상기 메시 서비스를 론칭하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 3

제 1 항에 있어서,

메시 서비스를 론칭하는 단계는:

특정 애플리케이션이 디바이스에서 론칭된 것에 응답하여, 메시 서비스가 현재 디바이스에서 실행 중인지 여부를 결정하는 단계; 및

메시 서비스가 현재 실행되고 있지 않으면, 디바이스에 메시 서비스를 론칭하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 4

제 3 항에 있어서,

메시 서비스가 현재 디바이스 상에서 실행중인 경우, 메시 서비스 버전이 최신인지를 결정하는 단계 및 상기 실행중인 메시 서비스를 종료하지 않는 경우 업데이트된 메시 서비스를 다시 론칭하는 단계를 더 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 5

제 4 항에 있어서,

메시 서비스를 업데이트하기 위한 프로그램 코드를 수신하는 단계; 및

메시 서비스를 업데이트하기 전에, 프로그램 코드 내에서 암호화 코드를 판독하고, 상기 암호화 코드가 디바이스에 이전에 저장된 암호화 코드와 일치하는지를 결정하는 단계를 더 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 6

제 3 항에 있어서,

메시 서비스는 디바이스의 프로세서 회로가 특정 애플리케이션을 론칭하기 위해 상기 프로세서 회로에 의해 실행되는 컴퓨터 판독 가능 명령의 애플리케이션 세트 내에 포함된 컴퓨터 판독 가능 명령의 메시 서비스 세트를 실행하게 함으로써 론칭되는 메시 네트워크를 통해 통신하는 방법.

청구항 7

제 1 항에 있어서,

각 디바이스의 프로세서 회로에 의해 운영되는 운영 시스템은 디바이스 상에서 서비스를 실행하기 위한 별개의 기능을 제공하도록 동작 가능하게 구성되고, 상기 방법은 서비스를 실행하기 위한 별개의 기능을 사용하여 메시 서비스를 실행하는 단계 및 디바이스에서 서비스를 실행하기 위해 애플리케이션의 액세스를 별개의 기능으로 애플리케이션에 의한 액세스를 제한하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 8

제 1 항에 있어서,

각각의 메시 포트는 고유 메시 포트 식별자와 관련된 메시 네트워크를 통해 통신하는 방법.

청구항 9

제 1 항에 있어서,

각각의 특정 애플리케이션은 고유 애플리케이션 식별자와 관련되고, 상기 메시 서비스로 하여금 상기 특정 애플리케이션이 상기 특정 메시 포트를 통한 통신에 대해 권한이 부여되었는지를 결정하게 하는 단계는:

특정 애플리케이션으로부터 애플리케이션 식별자를 수신하는 단계; 및

애플리케이션 식별자가 특정 애플리케이션에 의해 액세스될 수 없는 메모리 위치에 승인된 애플리케이션 식별자 및 관련 메시 포트의 저장된 리스팅에서 애플리케이션 식별자와 일치하는지 여부를 결정하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 10

제 9 항에 있어서,

애플리케이션 식별자는 디지털 서명을 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 11

제 1 항에 있어서,

디바이스에 현재 실행되고 있지 않은 애플리케이션에 대한 메시 포트와 연관된 특정 디바이스에서 실행되는 메시 서비스에서 데이터 전송을 수신한 것에 응답하여, 디바이스에서 실행되는 다른 애플리케이션에 의한 데이터 전송에 대한 액세스를 차단하면서 상기 메시 서비스로 하여금 메시 네트워크를 통해 데이터 전송을 전달하게 하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 12

제 1 항에 있어서,

디바이스에서 현재 실행중인 특정 애플리케이션에 대한 메시 포트와 연관된 특정 디바이스에서 실행되는 메시 서비스에서 데이터 전송을 수신하는 것에 응답하여, 상기 메시 서비스가:

데이터 전송을 애플리케이션으로 포워딩하고;

메시 네트워크를 통해 데이터 전송을 다른 디바이스로 전달하게 하는 메시 네트워크를 통해 통신하는 방법.

청구항 13

제 1 항에 있어서,

각 디바이스의 무선 라디오는 복수의 무선 전송 링크 중 어느 하나를 사용하여 메시 네트워크를 통해 통신하도록 동작 가능하고,

메시 서비스가 메시 네트워크 통신을 위한 복수의 무선 전송 링크 중 적어도 일부에 액세스하는 것을 활성화 또는 비활성화하기 위해 사용자 선호를 수신하기 위한 액세스를 제공하게 하는 단계를 더 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 14

제 1 항에 있어서,

각각의 디바이스 상의 메시 서비스에서 데이터 전송을 수신하는 것에 응답하여, 상기 데이터 전송이 메시 네트워크 통신 컨트롤과 관련된 데이터를 포함하는지 여부를 결정하는 단계 및 상기 데이터 전송이 메시 네트워크 통신을 제어하는 것과 관련된 데이터를 포함하는 것으로 결정하는 것에 응답하여, 상기 데이터 전송에 다른 데이터 전송보다 높은 전송 우선순위를 할당하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 15

제 14 항에 있어서,

데이터 전송에 더 높은 전송 우선순위를 할당하는 단계는 복수의 디바이스 중 어느 한 디바이스에 의한 이전 전송의 수신을 확인하는 데이터, 상기 메시 네트워크에 액세스하기 위한 디바이스에서 실행되는 애플리케이션과 관련된 데이터, 및디바이스에서 애플리케이션이 종료되는 것과 관련된 데이터에 가장 높은 전송 우선순위를 할당하는 단계를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 16

제 1 항에 있어서,

각각의 특정 애플리케이션은 애플리케이션과 메시 서비스 사이에서 데이터의 전송을 위해 상기 메시 서비스와 인터페이스하도록 상기 디바이스 상의 프로세서 회로를 지시하기 위한 일련의 애플리케이션 인터페이스 코드 세트를 포함하는 메시 네트워크를 통해 통신하는 방법.

청구항 17

제 1 항에 있어서,

메시 서비스는 메시 네트워크를 사용하여 애플리케이션을 개발하는 애플리케이션 개발자에게 디버깅 기능을 제공하도록 동작 가능하고, 메시 서비스가 유효한 개발자 키 서명을 제공하는 애플리케이션 개발자에게 디버깅 기능을 제한하게 하는 단계를 더 포함하는 메시 네트워크를 통해 통신하는 방법.

발명의 설명

기 술 분 야

[0001] 본 개시는 일반적으로 복수의 네트워킹 디바이스들 간에 확립된 메시 네트워크를 통한 통신에 관한 것이다.

배 경 기 술

[0002] 메시 네트워크는 디바이스가 네트워크 인프라를 만들기 위해 디바이스 간에 데이터를 릴레이하기 위해 협력하는

네트워크이다. 디바이스는 유선 또는 무선 네트워크일 수 있다. 메시 네트워크는 일반적으로 디바이스가 지속적으로 네트워크에 접속하거나 네트워크를 떠나는 점에서 동적이므로, 네트워크를 통한 데이터 전송은 물리적 네트워크 인프라가 일반적 지역에서 디바이스들에 의한 연결을 용이하게 하기 위해 제공되는 종래의 네트워크에서 직면되지 않은 문제를 제시한다.

발명의 내용

해결하려는 과제

[0003] 본 발명은 상기의 단점을 극복하기 위한 것이다.

과제의 해결 수단

[0004] 하나의 개시된 태양에 따르면, 각각이 무선 라디오를 갖는 복수의 디바이스들 간에 설정된 메시 네트워크를 통해 통신하기 위한 방법이 제공된다. 이 방법은 각 디바이스에서 메시 서비스를 론칭하는 단계를 포함하며, 상기 메시 서비스는 디바이스의 프로세서 회로가 메시 네트워크를 통해 디바이스들 간의 통신을 위해 무선 라디오를 컨트롤하기 위한 기능을 제공하도록 동작될 수 있다. 각각의 디바이스는 디바이스상에서 실행되는 적어도 하나의 애플리케이션을 가지며, 적어도 하나의 애플리케이션은 메시 포트와 관련되고, 상기 메시 포트는 복수의 디바이스들에서 적어도 일부의 디바이스상에서 실행되는 특정 애플리케이션의 인스턴스와 연관된 것으로 데이터 전송을 지정하는데 사용되며, 상기 적어도 하나의 애플리케이션 및 각각의 디바이스상의 메시 서비스는 데이터 통신한다. 이 방법은 또한 특정 메시 포트를 통한 통신을 위해 메시 네트워크에 대한 액세스를 제공하기 위해 메시 서비스를 요청하는 디바이스에 실행되는 특정 애플리케이션에 응답하여, 메시 서비스로 하여금 특정 애플리케이션이 특정 메시 포트상에서의 통신을 위한 승인되었는지를 결정하는 단계; 특정 애플리케이션이 승인된 경우, 메시 네트워크를 통해 특정 메시 포트에 통신하기 위해 애플리케이션으로부터의 요청을 처리하고 상기 특정 메시 포트와 관련된 데이터 전송을 상기 특정 애플리케이션으로 전달하는 단계; 및 특정 애플리케이션이 승인되지 않은 경우, 메시 네트워크를 통해 특정 메시 포트에 통신하기 위해 애플리케이션으로부터의 요청을 거절하고 상기 특정 메시 포트와 관련된 데이터 전송에 대한 특정 애플리케이션의 액세스를 방지하는 단계를 포함한다.

[0005] 메시 서비스를 론칭하는 단계는 디바이스에 운영 시스템을 부팅할 때 메시 서비스를 론칭하는 단계를 포함할 수 있다.

[0006] 메시 서비스를 론칭하는 단계는 특정 애플리케이션이 디바이스에서 론칭된 것에 응답하여 메시 서비스가 현재 디바이스에서 실행 중인지 여부를 결정하고, 메시 서비스가 현재 실행되고 있지 않은 경우, 디바이스에 메시 서비스를 론칭하는 단계를 포함할 수 있다.

[0007] 이 방법은, 메시 서비스가 현재 디바이스에서 실행중인 경우, 메시 서비스 버전이 최신인지 여부를 결정하고, 그렇지 않으면, 실행중인 메시 서비스를 종료하지 않고 업데이트된 메시 서비스를 다시 론칭하는 단계를 포함할 수 있다.

[0008] 이 방법은 메시 서비스를 업데이트하기 위한 프로그램 코드를 수신하고, 메시 서비스를 업데이트하기 전에, 프로그램 코드 내에서 암호화 코드를 판독하고 상기 암호화 코드가 디바이스에 미리 저장된 암호화 코드와 일치하는지를 결정하는 단계를 포함할 수 있다.

[0009] 메시 서비스는 디바이스의 프로세서 회로가 특정 애플리케이션을 론칭기 위해 프로세서 회로에 의해 실행되는 컴퓨터 판독 가능 명령의 애플리케이션 세트 내에 포함된 컴퓨터 판독 가능 명령어의 메시 서비스 세트를 실행하게 함으로써 론칭될 수 있다.

[0010] 각각의 디바이스의 프로세서 회로에 의해 실행되는 운영 시스템은 디바이스상에서 서비스를 실행하기 위한 분리된 기능을 제공하도록 동작 가능하게 구성될 수 있고, 상기 방법은 서비스를 실행하기 위해 분리된 기능을 사용하여 메시 서비스를 실행하고 디바이스에서 서비스를 실행하기 위한 상기 분리된 기능에 애플리케이션에 의한 액세스를 제한하는 단계를 포함할 수 있다 .

[0011] 각 메시 포트는 고유 메시 포트 식별자와 연관될 수 있다.

[0012] 각각의 특정 애플리케이션은 고유 애플리케이션 식별자와 연관될 수 있고 메시 서비스로 하여금 특정 애플리케이션이 특정 메시 포트에 통신하기 위해 승인되었는지 여부를 결정하는 단계는 특정 애플리케이션으로부터 애플

리케이션 식별자를 수신하는 단계를 포함할 수 있어, 특정 애플리케이션이 액세스할 수 없는 메모리 위치에 승인된 애플리케이션 식별자 및 관련 메시 포트의 저장된 리스트에 있는 애플리케이션 식별자와 일치하는지 여부를 결정한다.

[0013] 애플리케이션 식별자는 디지털 서명을 포함할 수 있다.

[0014] 현재 디바이스에서 실행되고 있지 않은 애플리케이션의 메시 포트와 연결된 특정 디바이스 상에 실행되는 메시 서비스에서 데이터 전송을 수신한데 응답하여, 메시 서비스가 디바이스 상에 실행되는 다른 애플리케이션에 의한 데이터 전송에 대한 액세스를 차단하면서 메시 네트워크를 통해 데이터 전송을 전달하게 한다.

[0015] 현재 디바이스에서 실행중인 특정 애플리케이션의 메시 포트와 연결된 특정 디바이스 상에 실행되는 메시 서비스에서 데이터 전송을 수신한데 응답하여, 메시 서비스가 데이터 전송을 애플리케이션으로 전달하고, 메시 네트워크를 통해 다른 디바이스로 상기 데이터 전송을 전달하게 한다.

[0016] 각각의 디바이스 상의 무선 라디오는 복수의 무선 전송 링크 중 어느 하나를 사용하여 메시 네트워크를 통해 통신하도록 동작될 수 있고, 메시 서비스가 메시 네트워크 통신을 위한 복수의 무선 전송 링크들 중 적어도 일부에 액세스를 가능하게 하거나 불가능하게 하기 위한 사용자 선호를 수신하기 위해 액세스를 제공하게 하는 단계를 더 포함할 수 있다.

[0017] 이 방법은 각 디바이스 상의 메시 서비스에서 데이터 전송을 수신하는 것에 응답하여, 데이터 전송이 메시 네트워크 통신을 컨트롤하는 것과 관련된 데이터를 포함하는지 여부를 결정하는 단계, 및 데이터 전송이 메시 네트워크를 컨트롤하는 것과 관련된 데이터를 포함한다고 결정하는 것에 응답하여, 상기 데이터 전송에 다른 데이터 전송보다 높은 전송 우선순위를 할당하는 단계를 포함할 수 있다.

[0018] 데이터 전송에 더 높은 전송 우선순위를 할당하는 단계는 복수의 디바이스 중 어느 하나의 디바이스에 의해 이전 전송의 수신 확인한 데이터, 메시 네트워크에 액세스하기 위해 디바이스에서 론칭된 애플리케이션과 관련된 데이터, 및 디바이스 상에 애플리케이션이 종료된 것과 관련된 데이터에 최상위 전송 우선순위를 할당하는 단계를 포함할 수 있다.

[0019] 각각의 특정 애플리케이션은 애플리케이션과 메시 서비스 간의 데이터 전송을 위해 디바이스 상의 프로세서 회로가 메시 서비스와 인터페이스하도록 지시하기 위한 애플리케이션 인터페이스 코드 세트를 포함할 수 있다.

[0020] 메시 서비스는 메시 네트워크를 사용하여 애플리케이션을 개발하는 애플리케이션 개발자에게 디버깅 기능을 제공하도록 동작할 수 있으며, 메시 서비스가 유효한 개발자 키 서명을 제공하는 애플리케이션 개발자로 디버깅 기능을 제한하게 하는 단계를 더 포함할 수 있다.

[0021] 다른 태양들 및 특징들은 첨부도면과 결부하여 하기의 특정 개시된 실시예들의 설명을 검토할 때 당업자에게 명백해질 것이다.

발명의 효과

[0022] 본 발명의 내용에 포함됨.

도면의 간단한 설명

[0023] 개시된 실시예들을 예시하는 도면에서,

도 1은 복수의 디바이스들 간에 확립된 메시 네트워크의 개략도이다.

도 2는 도 1에 도시된 디바이스를 구현하기 위한 프로세서 회로의 블록도이다.

도 3은 도 1에 도시된 디바이스 상에 구현된 기능 블록의 개략도이다.

도 4는 도 3에 도시된 기능 블록을 론칭 및 구성하기 위해 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

도 5는 도 1에 도시된 복수의 디바이스 중 어느 한 디바이스 상에 디스플레이된 사용자 인터페이스의 스크린 샷이다.

도 6은 도 1에 도시된 디바이스에서 메시 서비스를 론칭하기 위해 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

도 7은 도 1에 도시된 메시 네트워크의 일부의 개략도이다.

도 8a, 8b는 애플리케이션 이벤트 처리 프로세스를 구현하도록 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

도 9는 콘텐츠 데이터 전송을 도시한 개략도이다.

도 10a, 10b는 메시 서비스 이벤트 핸들러를 구현하도록 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

도 11a 내지 도 11e는 도 1에 도시된 메시 네트워크에서 신뢰할 수 있는 전송 프로토콜을 구현하도록 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

도 12는 도 1에 도시된 메시 네트워크를 통해 전송된 데이터 패킷의 예를 도시한 개략도이다.

도 13은 도 1에 도시된 메시 네트워크에서 멀티 캐스트 전송 프로토콜을 구현하도록 도 2의 프로세서 회로를 지시하기 위한 코드 블록을 도시한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0024] 도 1을 참조하면, 복수의 디바이스 간에 확립된 메시 네트워크가 전반적으로 100으로 도시되어 있다. 도 1에 도시된 실시예에서, 메시 네트워크는 제 1 디바이스(102), 제 2 디바이스(104), 제 3 디바이스(106) 및 제 4 디바이스(112)를 포함한다. 도시된 실시예에서, 디바이스(102, 104, 106 및 108)는 스마트 폰 디바이스이며, 캘리포니아 주 마운틴 뷰의 구글(Google)이 이용 가능하게 하는 Android™ 또는 임의의 다른 적합한 운영 시스템와 같은 스마트 폰 운영 시스템을 실행하고 있을 수 있다. 이 실시예에서, 제 1 디바이스(102)는 사용자(114)에 의해 작동되고 있다. 추가 디바이스가 또한 사용자(116)에 의해 작동되는 각각이 무선 라디오를 갖는 태블릿 컴퓨터(110) 및 랩톱 컴퓨터(112)를 포함하는 메시 네트워크(100)에 참여하고 있다. 다른 실시예에서, 디바이스(102-112)는 예를 들어 스마트 폰, 태블릿 또는 랩톱 컴퓨터, 데스크탑 컴퓨터와 같은 복수의 네트워킹 디바이스, 라우터 또는 액세스 포인트와 같은 독립형 네트워킹 디바이스, 프린터와 같은 컴퓨터 주변 디바이스, 및 네트워킹 어플라이언스와 같은 물리적 객체 중 어느 하나일 수 있다. 유선 연결을 갖는 다른 디바이스들(미도시)도 메시 네트워크에 참여할 수 있지만, 이러한 디바이스들은 도 1에 도시된 디바이스들과는 다른 네트워크 역할을 담당할 수 있다. 일반적으로 각각의 디바이스들(102-112)은 상기 각각의 디바이스로 하여금 메시 네트워크를 확립하기 위한 기능을 제공하게 하는 컴퓨터 판독 가능 명령어 형태로 로딩되는 애플리케이션을 갖는다. 메시 네트워크(100)에서, 디바이스(102)는 각각의 디바이스(104, 106, 110 및 112)와 무선 통신한다. 또한, 디바이스(104 및 110)는 디바이스(108)를 통해 무선 연결된다. 디바이스들(102-112)의 능력 및 서로에 대한 디바이스들의 지리적 위치에 따라 메시 네트워크(100) 내에 다양한 다른무선 링크들이 확립될 수 있다.

[0025] 디바이스(102-112) 중 어느 한 디바이스를 구현하기 위한 프로세서 회로의 블록도가 도 2에서 200으로 도시되어 있다. 도 2를 참조하면, 프로세서 회로(200)는 다중 프로세싱 코어를 포함할 수 있는 마이크로 프로세서(202)를 포함한다. 프로세서 회로(200)는 또한 디스플레이(204) 및 사용자 입력을 수신하기 위한 입력 디바이스(206)를 포함한다. 일부 실시예에서, 입력 디바이스(206)는 디스플레이(204) 상에 터치 스크린으로서 제공될 수 있다. 이 실시예에서, 프로세서 회로(200)는 디바이스 상에서 실행되는 애플리케이션과 관련된 데이터를 저장하기 위한 메모리(210)를 포함한다. 메모리(210)는 랜덤 액세스 메모리, 비휘발성 플래시 메모리, 하드 드라이브 또는 이들 및 다른 메모리 유형의 조합을 사용하여 구현될 수 있다. 메모리(210)는 프로그램 코드 및/또는 데이터를 저장하기 위해 사용되며, 도시된 실시예에서 운영 시스템 저장 위치(240), 메시 서비스 프로그램 코드를 저장하기 위한 메시 서비스 저장 위치(280), 애플리케이션 프로그램 코드를 저장하기 위한 애플리케이션 저장 위치(244), 사용자 선호 위치(246), 사용자 식별자(uuid) 파일 위치(248), 메시 포트 테이블 위치(250), 암호화 키 저장 위치(252), 메시 서비스 애플리케이션 데이터 버퍼 위치(256), 메시 서비스 전송 데이터 버퍼 위치(258), 메서 서비스 전송 큐 위치(260), 무선 링크 큐 위치(262), 라우팅 테이블 위치(264), 카운터 위치(266) 및 전달 버퍼 위치(268)를 이용해 구현될 수 있다.

[0026] 프로세서 회로(200)는 이동통신 네트워크에 연결하기 위한 RF 기저대역 라디오(212) 및 안테나(214)를 더 포함한다. RF 기저대역 라디오(212)는 2G, 3G, 4G 또는 다른 통신 표준을 포함하는 다양한 통신 표준 중 어느 하나를 사용하여 데이터 통신을 제공하도록 구성될 수 있다.

[0027] 프로세서 회로(200)는 또한 IEEE 802.11 WLAN 로컬 네트워크와 같은 로컬 네트워크에 연결하기 위한 무선 라디오(216) 및 안테나(218)를 포함한다. 무선 라디오(216)는 또한 블루투스, Wi-Fi 다이렉트 또는 근거리 통신과

같은 다른 무선 링크 또는 프로토콜을 통한 연결을 제공할 수 있다.

[0028] 프로세서 회로(200)는 선택적으로 위치 수신기(230)를 더 포함한다. 위치 수신기(230)는 GPS(global positioning system) 신호를 수신하기 위한 안테나(218)를 포함하고, 위치 수신기(230)는 네트워크 디바이스의 위치를 결정하기 위해 셀룰러 서비스 공급자가 제공하는 특정 로컬 네트워크 액세스 포인트 또는 셀룰러 신호 삼각측량 정보의 알고 있는 위치와 같은 다른 위치 정보와 조합하여 상기 GPS 정보를 사용할 수 있다.

[0029] 프로세서 회로(200)는 오디오 프로세서(220), 마이크로폰(222) 및 스피커(224)를 더 포함한다. 오디오 프로세서(220)는 마이크로폰(222)으로부터 오디오 입력 신호를 수신 및 처리하고 스피커(224)에서 오디오 출력을 생성한다. 프로세스 회로(200)는 또한 비디오/이미지 프로세서(226) 및 카메라(228)를 포함한다. 비디오/이미지 프로세서(226)는 카메라(228)로부터 이미지 및/또는 비디오 신호를 수신하고 처리한다.

[0030] 디스플레이(204), 입력 디바이스(206), 메모리(210), RF 기저대역 라디오(212), 무선 라디오(216), 오디오 프로세서(220) 및 비디오/이미지 프로세서(226)는 모두 마이크로 프로세서(202)와 통신한다.

[0031] 운영 시스템 저장 위치(240)는 마이크로 프로세서(202)로 하여금 운영 시스템을 구현하도록 지시하기 위한 코드를 저장하며, 스마트 폰 디바이스(102-106)의 경우 Android ™ 기반 운영 시스템, iOS 기반 운영 시스템, 또는 임의의 다른 운영 시스템일 수 있다. 태블릿 컴퓨터(110) 및 랩톱 컴퓨터(112)는 Android, iOS, Windows®, Linux 또는 다른 적절한 운영 시스템을 실행하고 있을 수 있다. 본 명세서의 나머지 개시는 일반적으로 안드로이드 기반 운영 시스템 하에서 다양한 개시된 실시예의 구현에 관한 것이지만, 동일한 원리가 일부 구현 차이를 갖는 다른 운영 시스템에도 적용된다.

[0032] 디바이스들(102-108)은 각각 도 2에 도시된 것과 매우 유사한 프로세서 회로(200)를 사용하여 구현될 수 있다. 랩탑 컴퓨터 디바이스(112)는 도 2에 도시된 많은 컴포넌트들을 포함할 수 있고, 일부 컴포넌트들은 위치 수신기(230) 및 RF 기저대벽 라디오(212)와 같이 가능하게는 생략될 수 있으나, 이들 부품들은 그럼에도 불구하고 랩탑 컴퓨터에 포함될 수 있다. 실시예가 도 2의 프로세서 회로(200) 아키텍처를 참조하여 여기에서 설명되지만, 설명된 시스템 실시예 및/또는 프로세스 실시예는 다른 유형의 디바이스 사이의 통신에도 적용 가능하다. 일반적으로, 각각의 디바이스(102-112)는 플래시 메모리(210)(또는 다른 메모리 유형)에 로딩된 컴퓨터 판독 가능 코드를 가지며, 이는 각각의 디바이스가 메시 네트워크(100)를 확립하기 위해 필요한 기능을 제공하게 한다. 일부 실시예에서, 메시 네트워크(100)에 참여하는 디바이스는 도 2에 도시된 부품들 중 다수를 생략할 수 있다. 예를 들어, 디바이스는 스마트 기기, 차량 또는 다른 물리적 물체 내에 연결된 디바이스로서 통합될 수 있고 디스플레이(204), 입력 디바이스(206) 또는 도 2에 도시된 다른 도시된 부품들과 같은 요소를 포함하지 않을 수 있다. 연결된 디바이스는 예를 들어 java, c 또는 c ++ 코드를 실행할 수 있으며 IEEE 802.11, Bluetooth, Wi-Fi 다이렉트 또는 무선 링크를 확립하기 위한 근거리 통신 프로토콜을 구현하는 무선 라디오(216)를 포함할 수 있다.

[0033] 일 실시예에서, 메시 네트워크(100)는 2016년 5월 30일자로 출원된 "METHOD FOR ESTABLISHING NETWORK CLUSTERS BETWEEN NETWORKED DEVICES"이라는 제목의 공동 소유 특허 미국 가출원 제62/343,056호에 개시된 바와 같이 일반적으로 확립될 수 있으며, 그 전체가 본 명세서에 참조로 포함된다. 이와 같이, 메시 네트워크(100)의 디바이스는 액세스 포인트, 클라이언트 또는 라우터로서 작용하도록 구성될 수 있다. 액세스 포인트 디바이스(또한 본 명세서에서 "마스터 모드"라고 함)는 클라이언트로 구성된 다른 디바이스(본 명세서에서 "클라이언트 모드"라고 함)가 메시 네트워크(100)에서 액세스 포인트에 접속하여 상기 액세스 포인트를 통해 다른 클라이언트로 데이터를 송수신할 수 있게 한다. 일부 클라이언트 디바이스는 라우터(여기서는 "라우팅 모드"라고 함)로 더 구성되며, 각 액세스 포인트에 연결을 번갈아 가면서 액세스 포인트 모드로 구성된 두 디바이스들 간에 링크를 제공하도록 작동될 수 있다.

[0034] 도 3을 참조하면, 메시 네트워크와 상호 작용하기 위해 도 1에 도시된 임의의 디바이스 상에 구현된 기능 블록의 개략도가 300에 도시되어 있다. 디바이스(300)의 기능 블록도에서 블록은 프로세서 회로(200)의 마이크로 프로세서(202)가 각각의 기능을 구현하기 위해 디바이스 상에 필요한 기능을 구현하도록 지시하는 컴퓨터 판독 가능 코드를 나타낸다. 도시된 실시예에서, 제 1 애플리케이션(302) 및 제 2 애플리케이션(304)은 디바이스상에서 실행되는 것으로 도시되고 컨텐츠 공유, 채팅, 응급 서비스 연락처 정보 등과 같은 다양한 작업을 수행하기 위해 디바이스상에서 기능을 구현할 수 있다. 도시된 실시예에서, 제 1 애플리케이션(302)이 채팅 애플리케이션이고, 제 2 애플리케이션(304)이 긴급 애플리케이션이다. 애플리케이션(302)은 애플리케이션 인터페이스(306)를 가지며, 이는 디바이스 상에 실행되는 메시 서비스(310)와 통신한다. 마찬가지로, 제 2 애플리케이션(304)은 애플리케이션 인터페이스(308)를 가지며, 이는 디바이스 상에 실행되는 메시 서비스(310)와 통신한다. 메시 서비

스(310)는 프로세서 회로(200)의 마이크로 프로세서(202)가 메시 네트워크(100)를 통한 디바이스들 간의 통신을 위해 무선 라디오(216)를 컨트롤하게 한다. 디바이스상에서 실행되고 애플리케이션(302 및 304) 모두(및 디바이스에서 실행중인 임의의 추가 애플리케이션)와 통신하는 단일 메시 서비스(310)만이 있다.

[0035] 도 4를 참조하면, 디바이스의 프로세서 회로(202)로 하여금 도 3에 도시된 기능 블록을 론칭하고 구성하도록 지시하기 위한 코드 블록을 도시한 흐름도가 전반적으로 400으로 도시되어 있다. 도 4의 블록은 일반적으로 마이크로 프로세서(202)가 도 3에 도시된 기능 블록들을 구현하도록 지시하기 위한 메모리(210)로부터 판독할 수 있는 코드를 나타낸다. 각각의 블록을 구현하기 위한 실제 코드는 가령 Java, C, Objective-C, C ++, C# 및/또는 어셈블리 코드와 같은 임의의 적절한 프로그램 언어로 작성될 수 있다. 프로세스는 블록(402)에서 시작하는데, 이는 애플리케이션 저장 위치(244)에 저장된 제 1 애플리케이션에 대한 컴퓨터 판독 가능 코드를 실행함으로써 마이크로 프로세서(202)가 애플리케이션(이 경우 제 1 애플리케이션(302))을 론칭하도록 지시한다. 상기 애플리케이션이 론칭되면, 제 1 애플리케이션(302)에 대한 애플리케이션 인터페이스(306)가 애플리케이션 저장 위치(244)에서 컴퓨터 판독 가능 코드에 저장된 암호화 코드 또는 서명을 판독한다. 암호화 코드는 애플리케이션의 메시 서비스 기능을 구현하는 데 사용될 수 있는 함수 라이브러리를 포함하는 소프트웨어 개발자 툴 세트를 수신할 때 제 1 애플리케이션의 개발자에 의해 획득될 수 있다. 이와 같이, 암호화 코드는 디바이스(300)상에서 애플리케이션을 실행하는데 사용되는 애플리케이션 저장 위치(244)로부터 판독된 컴퓨터 판독 가능 코드의 일부로서 포함될 수 있다.

[0036] 블록(404)은 이어서 마이크로 프로세서(202)로 하여금 메시 서비스(310)가 디바이스에서 이미 실행 중인지 여부를 결정하도록 지시한다. 메시 서비스(310)가 이미 실행중인 경우, 블록(404)은 마이크로 프로세서(202)를 블록(406)으로 지시하여, 마이크로 프로세서가 이벤트를 기다리게 한다.

[0037] 블록(404)에서 메시 서비스(310)가 아직 디바이스에서 실행되고 있지 않으면, 블록(404)은 마이크로 프로세서(202)를 블록(408)으로 지시하여, 마이크로 프로세서로 하여금 디바이스에 메시 서비스가 가능한지 여부를 결정하도록 지시한다. 도 5를 참조하면, 사용자 선호를 컨트롤하기 위해 사용자 인터페이스의 디바이스에 디스플레이된 스크린 샷이 500으로 도시되어 있다. 사용자 선호 인터페이스(500)는 디바이스의 사용자가 메시 서비스(310)가 활성화 또는 비활성화되는지 여부에 관해 선호를 설정할 수 있게 하는 "메시 서비스" 컨트롤(502)을 포함한다("메시 서비스" 컨트롤은 도 5에서 활성화된 것으로 도시되어 있다). 사용자 선호 인터페이스(500)는 또한 "Wi-Fi 메시" 컨트롤(504), "블루투스 메시" 컨트롤(506) 및 "마스터 모드", "클라이언트", "라우터 모드" 및 "자동 모드" 중에서 선택하기 위한 라디오 버튼을 포함하는 네트워크 역할 선택기(508)를 포함하는 다른 사용자 선호 컨트롤을 포함한다. 사용자 선호 인터페이스(500)는 또한 사용자가 인터넷에 대한 연결을 공유하기 원하는지를 나타내는 인터넷 공유 컨트롤(510)을 포함한다. 예를 들어, 디바이스들 중 하나에 대한 RF 기저대역 라디오(212) 또는 다른 인터페이스가 데이터 네트워크를 통해 인터넷에 대한 액세스를 제공할 수 있고, 사용자는 이 액세스를 메시 네트워크(100)상의 다른 디바이스와 공유하도록 선택할 수 있다.

[0038] 사용자 선호가 사용자 선호 인터페이스(500)에서 수신되고 메모리(210)의 사용자 선호 위치(246)에 저장된다. 도 4를 다시 참조하면, 블록(408)은 마이크로 프로세서로 하여금 사용자 선호 위치(246)에 저장된 "메시 서비스" 컨트롤의 상태를 읽도록 지시하고, 가능하다면, 마이크로 프로세서를 블록(410)으로 지시한다. 블록(410)은 또한 마이크로 프로세서(202)가 메시 서비스 저장 위치(280)에 저장된 메시 서비스 프로그램 코드를 실행함으로써 메시 서비스(310)를 론칭하도록 지시한다. 블록(405)은 또한 마이크로 프로세서(202)가 메시 서비스(310)에 연결하고 서명을 교환하도록 지시한다. 그 후 프로세스는 블록(406)에서 계속되며, 이는 마이크로 프로세서(202)가 다음 이벤트를 기다리도록 지시한다.

[0039] 도시된 실시예에서, 메시 서비스(310)는 디바이스 상에 서비스로서 론칭되며, 이는 하나 이상의 애플리케이션에 의해 사용될 수 있는 소프트웨어 기능을 가리킨다. 프로세서 회로(200)는 서비스에 대한 손상 및/또는 금지된 액세스를 방지하기 위해 애플리케이션에 의한 액세스로부터 보호되는 모드로 서비스를 실행할 수 있다. 서비스는 애플리케이션에 의해 사용하기 위해 서비스에 의해 제공되는 기능을 노출하는 API(application programming interface)를 통해 애플리케이션에 기능을 제공할 수 있다.

[0040] 블록(408)에서, 사용자 선호 위치(246)에 저장된 "메시 서비스" 컨트롤의 상태가 비활성화되면, 마이크로 프로세서는 서비스가 사용자에 의해 활성화될 때까지 대기하도록 블록(412)으로 지시된다. 따라서, 메시 네트워크(100)로의 연결은 사용자가 "메시 서비스" 컨트롤(502)의 상태를 활성으로 변경할 때까지 정지된다. 그러나, 애플리케이션은 계속 오프라인 기능을 수행할 수 있다. 마이크로 프로세서(202)가 "메시 서비스" 컨트롤(502)이 활성인 것을 검출하면, 마이크로 프로세서는 다시 블록(404)으로 지시된다.

[0041] 블록(406)은 마이크로 프로세서(202)가 이벤트를 수신했는지 여부를 결정하도록 지시한다. 이벤트가 수신되지 않았다면, 마이크로 프로세서(202)는 반복 블록(406)으로 지시된다. 블록(406)에서, 이벤트가 검출되면, 블록(406)은 마이크로 프로세서(202)를 도 8에 도시된 애플리케이션 이벤트 처리 프로세스(800)의 블록(802)으로 지시한다.

[0042] 도 4에 도시된 실시예에서, 메시 서비스(310)의 론칭은 특정 애플리케이션(이 경우에 제 1 애플리케이션(302))이 디바이스에서 론칭되는 것에 응답한다. 메시 서비스(310)가 현재 실행되고 있지 않은 경우, 메시 서비스가 실행되도록 사용자 선호가 설정되면 메시 서비스가 디바이스에 론칭된다. 다른 실시예에서, 메시 서비스(310)는 디바이스의 운영 시스템을 부팅할 때(즉, 메모리(210)의 운영 시스템 저장 위치(240)로부터) 론칭될 수 있다.

[0043] 일 실시예에서, 메시 서비스 저장 위치(280)에 저장된 메시 서비스 프로그램 코드는 애플리케이션(302 및 304) 중 하나 또는 둘 모두와 관련된 애플리케이션 코드와 함께 또는 그 일부로서 제공될 수 있다. 메시 서비스(310)가 현재 디바이스상에서 실행중인 것을 블록(404)에서 발견하면, 프로세스(400)는 실행중인 메시 서비스 버전이 최신인지를 결정하는 단계를 추가로 포함할 수 있다. 버전이 최신이 아닌 경우, 메시 서비스(310)의 실행 버전이 종료될 수 있고 블록(410)은 업데이트된 메시 서비스를 재론칭하기 위해 반복될 수 있다. 예로서, 제 2 애플리케이션(304)은 제 1 애플리케이션(302) 이전에 메모리(210)로 다운로드될 수 있고, 따라서 제 1 애플리케이션은 업데이트된 버전의 메시 서비스(310)와 함께 패키지될 수 있다. 대안으로, 제 1 애플리케이션(302)의 업데이트된 버전에 대한 프로그램 코드가 디바이스(300) 상에 수신되어 설치될 수 있고, 업데이트된 프로그램 코드는 업데이트된 메시 서비스 프로그램 코드를 포함할 수 있다. 따라서, 메시 서비스에 대한 업데이트가 새로운 애플리케이션을 로딩하는 디바이스에 자동으로 롤아웃될 수 있다는 점에서, 애플리케이션 코드와 함께 메시 서비스 프로그램 코드를 배포하는 것과 관련된 이점이 있다.

[0044] 도 3을 다시 참조하면, 애플리케이션 인터페이스(306 및 308)와 메시 서비스(310) 사이의 프로세스 간 통신이 화살표(312 및 314)로 표시된다. 이러한 프로세스 간 통신(312)은 디바이스상에서 실행되는 운영 시스템에 의존하며 디바이스와 관련된 하나 이상의 프로토콜을 사용하여 구현될 수 있다. 예를 들어, 안드로이드 운영 시스템에는 메시 서비스(310)와 애플리케이션 인터페이스(306, 308) 사이의 프로세스 간 통신에 사용될 수 있는 4개의 다른 프로토콜이 있다. 안드로이드 기반 디바이스에서, 프로세스는 다른 프로세스에 할당된 메모리에 직접 액세스할 수 없고 프로세스 간의 통신은 기능 및 프로토콜을 제공하는 운영 시스템을 사용하여 수행해야 한다.

[0045] (Android 플랫폼에 특정한) 브로드캐스트 인텐트(*broadcast intents*)로 알려진 제 1 프로토콜은 사용자가 Wi-Fi를 비활성화하거나 블루투스를 활성화하거나 가령 사용자 선호 인터페이스(500)에 대한 다른 권한을 변경했을 때 애플리케이션 인터페이스(306 및 308)와 메시 서비스(310) 사이에 소량의 데이터를 전송하는데 사용될 수 있다. 다른 운영 시스템 플랫폼은 가령 Linux 운영 시스템용의 POSIX 메시지 큐, 스레드 메시지 또는 Windows 운영 시스템용의 Microsoft 메시지 큐잉의 유사한 기능의 프로토콜을 제공한다.

[0046] 애플리케이션 인터페이스들(306 및 308)과 메시 서비스(310) 사이에서 데이터를 교환하기 위한 제 2 통신 프로토콜은 동일한 애플리케이션을 실행할 수 있는 다른 디바이스들의 리스트와 같은 더 큰 데이터 교환에 사용되는 메신저 송수신(*send/recv*) 기능을 통해 이루어진다. 송수신 기능은 또한 메시 서비스(310)에 의해 전송될 콘텐츠 데이터를 교환하거나 메시 서비스로부터 애플리케이션 인터페이스(306 및 308)에서 수신된 데이터에 사용될 수 있다. 메신저 송수신 기능을 사용하는 데이터 전송의 크기가 일반적으로 제한되며, 더 큰 콘텐츠 데이터 교환은 복수의 메시지를 사용하여 분할되고 전송될(즉, 패킷화될) 필요가 있을 것이다.

[0047] 제 3 통신 프로토콜은 공통 SQLite(SQL) 데이터베이스 또는 파일을 사용하여 데이터를 교환하여, 데이터베이스 또는 파일이 메모리(210)의 공통으로 액세스 가능한 위치에 저장되고 메시 서비스(310)에 의해 또는 애플리케이션 인터페이스(306 및 308)에 의해 판독될 수 있기 때문에 더 빠른 데이터 전송을 가능하게 한다. SQLite 프로토콜은 더 큰 파일 크기를 갖는 콘텐츠 데이터, 예를 들어 비디오 또는 더 큰 이미지 콘텐츠 데이터를 교환하는데 효과적일 수 있다. SQL 또는 파일을 사용하여 데이터를 교환하는 것은 또한 데이터의 영구 저장을 제공한다는 이점을 가지며, 이는 메시 서비스(310)가 셧다운되는 경우에 유리하다. 상기 메신저 송수신 프로토콜하에, 메시 서비스(310)에 의해 아직 전송되지 않은 데이터는 어떤 이유로 서비스가 셧다운될 경우 손실될 수 있다.

[0048] 일 실시예에서, 송수신 프로토콜은 애플리케이션 인터페이스(306 및 308)와 메시 서비스(310) 사이에서 데이터를 전송하기 위한 이 프로토콜의 사용이 전송 병목현상을 야기하는 메시 네트워크(100)를 통한 통신 속도가 충분히 높을 때까지 사용될 수 있다. 그러한 경우에, 제 3 통신 프로토콜은 전송 병목현상을 제거하는데 사용될 수 있다.

[0049] AIDL(Android Interface Definition Language)로 알려진 제 4 프로토콜은 예를 들어 동일한 애플리케이션을 실행할 수 있는 다른 디바이스 리스트 또는 전송할 파일의 리스트와 같이 데이터 교환을 위한 메신저 송수신 기능의 대안으로 사용될 수 있다.

[0050] 도 6을 참조하면, 제 1 애플리케이션(302) 이후에 메시 서비스(310)에 의해 실행된 프로세스는 마이크로 프로세서(202)가 프로세스(400)의 블록(410)에서 메시 서비스를 론칭하도록 지시한다. 프로세스는 600으로 도시된다. 프로세스는 블록(602)에서 시작하고, 디바이스의 마이크로 프로세서(202)가 메모리(210)의 uuid 저장 위치(248)에서 디바이스 상에 저장된 메시 네트워크 식별자(uuid) 파일이 존재하는지 여부를 결정하게 한다. uuid 파일이 존재하지 않는 경우, 블록(602)은 마이크로 프로세서(202)를 블록(604)으로 지시하고, 마이크로 프로세서가 디바이스에 대한 메시 네트워크 식별자를 생성하게 한다. 일 실시예에서, uuid는 결과 식별자가 고유할 매우 높은 가능성을 제공하는 블록체인 호환 uuid 생성기를 사용하여 생성될 수 있다. 그런 후, 블록(606)은 마이크로 프로세서(202)가 사용자에게 메모리(210)의 uuid 저장 위치(248)에 uuid 파일을 기록하기 위한 액세스를 승인해 주도록 요청하게 한다. 블록(606)에서 사용자에 의해 권한이 부여되지 않으면, 마이크로 프로세서(202)는 블록(612)으로 지시되고 세션은 생성된 uuid로 계속된다. 그러나, 권한이 주어지지 않은 경우, 디바이스 사용자가 개시한 후속 세션에는 현재 활성화된 uuid와 다른 uuid 생성이 다시 필요하다.

[0051] 블록(606)에서 사용자가 uuid를 uuid 저장 위치(248)에 기록하는 권한이 주어지면, 마이크로 프로세서(202)는 블록(608)으로 지시되고 uuid가 메모리(210)에 저장된다. 사용자가 uuid 파일을 삭제하지 않는 한, 저장된 uuid 파일은 애플리케이션(302 및 304)이 제거 되더라도 사용 가능하게 유지될 것이다. 사용자가 이어서 메시 네트워크(100)와 함께 사용하기 위해 새로운 애플리케이션을 설치하면, uuid는 사용 가능한 상태로 유지된다. 일 실시예에서, uuid는 이더리움 호환가능 암호화된 지갑으로서 uuid 저장 위치(248)에 저장될 수 있다. 그런 후, 블록(608)은 마이크로 프로세서(202)를 블록(612)으로 지시한다.

[0052] 블록(612)은 마이크로 프로세서(202)가 디바이스에 설정된 위치 권한이 Wi-Fi, Wi-Fi 다이렉트 및 Bluetooth 무선 프로토콜을 통해 메시 네트워크(100)에 대한 액세스를 허용하는지 여부를 결정하도록 한다. 액세스가 허용되지 않으면, 블록(612)은 마이크로 프로세서(202)를 블록(614)으로 지시하고, Wi-Fi, Wi-Fi 다이렉트 및 블루투스 사용자 선호가 메모리(210)의 사용자 선호 위치(246)에 저장된다. 도 5를 다시 참조하면, 이는 사용자 선호 인터페이스(500)에서 "Wi-Fi 메시" 컨트롤(504)을 비활성화로 설정하고 "블루투스 메시" 컨트롤(506)을 비활성화로 설정하는 것에 해당한다. 블록(612)에서 액세스가 허용되면, 블록(612)은 마이크로 프로세서(202)를 블록(616)으로 지시한다.

[0053] 블록(616)은 마이크로 프로세서(202)로 하여금 디바이스가 Wi-Fi 통신 설정의 변경을 허용하는 운영 시스템 버전을 갖는지 여부 및 사용자가 디바이스에서 이러한 설정의 변경을 허용했는지 여부를 결정하도록 지시한다. 예를 들어 Android 버전 5.x 이상에서는 Wi-Fi 설정을 변경할 수 있지만 사용자가 그러한 변경을 수행할 권한을 부여해야 하는 것을 필요로 한다. 메시 네트워크(100)의 확립은 디바이스 및 무선 라디오(216)의 특정 설정의 조작을 필요로 하기 때문에, 이러한 설정에 대한 액세스가 비활성화될 때, 디바이스는 블루투스를 통해서만 메시 네트워크와의 연결로 제한될 수 있다. 블록(616)에서 Wi-Fi 설정의 변경이 허용되지 않으면, 프로세스는 블록(618)에서 계속되고 여기서 마이크로 프로세서(202)는 무선 라디오(216)를 통한 Wi-Fi 및 Wi-Fi 다이렉트 통신을 비활성화도록 지시된다. 이는 도 5에 도시된 사용자 기본 설정 인터페이스(500)에서 "Wi-Fi 메시" 컨트롤(504)"을 비활성화로 설정하는 것에 해당한다

[0054] 블록(616)에서 Wi-Fi 설정의 변경이 허용되면, 프로세스는 블록(620)에서 계속되고, 이는 마이크로 프로세서(202)로 하여금 이벤트가 애플리케이션 인터페이스(306 또는 308)(도 3) 또는 무선 라디오(216)를 통해 메시 네트워크(100)로부터 수신되었는지를 지시한다. 이벤트가 수신되지 않으면, 마이크로 프로세서(202)는 반복 블록(620)으로 지시된다. 이벤트가 수신되면, 블록(620)은 마이크로 프로세서(202)를 도 10의 블록(1002)으로 지시한다.

[0055] 도 3에 도시된 디바이스 구성의 하나의 장점은 특정 애플리케이션과 관련된 데이터 트래픽의 분리를 허용하기 위해 특정 애플리케이션(예를 들어, 도 3에 도시된 디바이스(300)에 대한 기능 블록도에서 제 1 애플리케이션(302) 및 제 2 애플리케이션(304))과 관련된 데이터 흐름을 추가로 고유하게 식별함으로써 제공된다. 도 3에 도시된 디바이스(300)는 메시 네트워크(100)를 통해 다른 디바이스와 통신하는 도 7에 도시되어 있다. 도 7을 참조하면, 디바이스(300)는 디바이스(702)와 통신하는 디바이스(700)와 무선 통신한다. 디바이스(300)는 제 1 애플리케이션(302) 및 제 2 애플리케이션(304) 모두를 실행하는 반면, 디바이스(702)는 제 2 애플리케이션의 인스턴스(710)만을 실행하고 있다. 디바이스(700)는 메시 네트워크(100)를 통해 네트워크 디바이스에 의해 재생되는

게임과 관련된 제 3 애플리케이션(704)을 실행하고 있다. 도 7에서, 디바이스(300 및 700)는 무선 통신 범위(716) 내에 있는 반면, 디바이스(700 및 702)는 무선 통신 범위(718) 내에 있다. 그러나, 도시된 실시예에서, 디바이스들(300 및 702)은 무선 통신 범위를 벗어나 있다. 무선 통신 범위(716 및 718)는 따라서 이 경우 3개의 디바이스를 포함하는 메시 네트워크(100)를 정의한다.

[0056] 도 7에 도시된 실시예에서, 디바이스(300)의 메시 서비스(310) 및 디바이스(702)상에서 실행되는 메시 서비스(714)는 메시 서비스(714)를 통해 제 2 애플리케이션(304 및 710) 사이의 전송을 위한 데이터를 통신해야 한다. 이 실시예에서, 각각의 데이터 흐름은 데이터 전송과 관련된 특정 애플리케이션을 식별하기 위한 수단을 제공하는 식별자와 관련된다. 본 개시에서 식별자를 "메시 포트"라 하고 제 1 애플리케이션에는 메시 포트(6000)가 할당되며, 제 2 애플리케이션에는 메시 포트(5000)가 할당되고, 제 3 애플리케이션에는 메시 포트(7000)가 할당된다. 할당된 메시 포트는 디바이스(702)상에서 실행되는 제 2 애플리케이션(710)의 인스턴스로만 전달되도록 디바이스(300)상에서 실행중인 제 2 애플리케이션(304)의 인스턴스와 연관된 데이터 흐름을 허용한다. 5000의 메시 포트를 갖는 데이터를 수신한 디바이스(700)에서 실행되는 메시 서비스(708)는 데이터를 디바이스(702)로 여전히 보낼 수 있으나, 디바이스(700) 상에 실행되는 제 3 애플리케이션(704)으로는 데이터를 보내지 않을 수 있다. 따라서, 디바이스(700)는 메시 네트워크(100)에 참여하지만, 제 3 애플리케이션(704)의 인스턴스를 실행하는 다른 디바이스가 메시 네트워크(100)에 합류하지 않거나 디바이스(300 또는 702) 중 하나가 제 3 애플리케이션의 인스턴스를 론칭하지 않는 한, 제 3 애플리케이션(704)은 어떠한 데이터도 수신하지 않을 것이다. 이는 메시 네트워크(100)를 통해 제 1, 제 2 및 제 3 애플리케이션 각각에 관련 데이터를 오직 제공하는 이점을 갖는다. 또한, 제 3 애플리케이션(704)이 메시 포트(5000 또는 6000) 중 하나를 청취하려고 시도하면, 디바이스(700) 상에 실행되는 메시 서비스(708)는 그러한 액세스를 금지할 것이다.

[0057] 프로세스(400)의 블록(406)에서 수신된 이벤트를 처리할 때 프로세서 회로(200)에 의해 실행되는 애플리케이션 이벤트 처리 프로세스는 도 8에 전반적으로 800으로 도시되어 있다. 애플리케이션 이벤트 처리 프로세스(800)는 디바이스(300)를 참조로 설명되지만, 동일한 프로세스가 또한 각각의 디바이스(700 및 702)에 실행된다. 애플리케이션 이벤트 처리 프로세스(800)는 디바이스(300)의 애플리케이션 인터페이스(306 및 308)를 구성하여 이벤트를 처리하기 위한 기능을 제공한다. 메시 서비스(310) 또는 제 1 또는 제 2 애플리케이션(302 및 304)으로부터 애플리케이션 인터페이스(306 및 308)에 의해 다양한 이벤트가 수신될 수 있다. 도 8a를 참조하면, 애플리케이션 이벤트 처리 프로세스(800)는 블록(802)에서 시작하여, 사용자가 디바이스가 동작하는 모드와 관련된 사용자 선호 위치(246)에 저장된 사용자 선호를 변경했는지 여부를 결정하도록 마이크로 프로세서(202)를 지시한다. 도 5를 다시 참조하면, 사용자 선호 인터페이스(500)는 사용자가 메시 네트워크(100)에서 디바이스에 대한 네트워크 역할을 선택할 수 있게 하는 복수의 버튼 컨트롤(508)을 포함한다. 사용자는 마스터 모드(즉, Wi-Fi 액세스 포인트로서), 클라이언트 모드, 라우팅 모드 또는 자동 모드로 동작하는 것 중에서 선택할 수 있다. 네트워크 역할의 변화가 블록(802)에서 마이크로 프로세서(202)에 의해 감지되면, 마이크로 프로세서는 블록(804)으로 지시된다. 블록(804)은 마이크로 프로세서(202)가 메시 네트워크(100)에서 네트워크 역할에 대한 변경된 사용자 선호를 메모리(210)의 사용자 선호 위치(246)에 저장하도록 지시한다. 블록(804)은 또한 마이크로 프로세서(202)가 브로드캐스트 인텐트 프로토콜을 통해 네트워크 역할의 변화를 메시 서비스(310)로 통신하도록 지시한다. 그 후, 프로세스는 프로세스(400)의 블록(406)으로 되돌아 가고, 여기서 마이크로 프로세서(202)는 다음 이벤트를 기다리도록 지시된다.

[0058] 네트워크 역할의 변화가 블록(802)에서 감지되지 않으면, 프로세스는 블록(806)에서 계속되며, 이는 마이크로 프로세서(202)로 하여금 사용자가 메시 서비스(310)를 비활성화했는지 여부를 결정하도록 지시한다. 메시 서비스(310)가 블록(806)에서 사용자에 의해 비활성화된 경우, 마이크로 프로세서(202)는 블록(804)으로 다시 지시되어, 여기서 사용자 선호 위치(246)에 저장된 사용자 선호가 업데이트되고 메시 서비스(310)는 브로드캐스트 인텐트 프로토콜 메시지를 통해 변경을 통지 받는다. 그 후 프로세스는 프로세스(400)의 블록(406)으로 되돌아 가고, 여기서 마이크로 프로세서(202)는 다음 이벤트를 기다리도록 지시된다.

[0059] 메시 서비스(310)가 블록(806)에서 비활성화되지 않으면, 프로세스는 블록(808)에서 계속되어 마이크로 프로세서(202)가 애플리케이션이 메시 포트의 바인딩 또는 언바인딩을 요청했는지를 결정하도록 지시한다. 상술한 바와 같이, 제 1 애플리케이션(302)은 메시 포트(6000)를 통한 통신용으로 구성되는 반면, 제 2 애플리케이션(304)은 메시 포트(5000)를 통한 통신용으로 구성된다. 예를 들어, 제 1 애플리케이션(302)이 메시 포트(6000)에 대한 바인딩을 요청하면, 블록(808)은 마이크로 프로세서(202)를 블록(810)으로 지시한다. 그런 후, 블록(810)은 마이크로 프로세서(202)가 메시 포트(6000)에 대한 바인딩을 요청하기 위해 메시 서비스(310)에 의해 제공되는 메시 포트 바인딩 API 기능을 호출하도록 지시한다.

[0060] 애플리케이션(302, 304, 및 704)과 같은 애플리케이션은 다양한 상이한 애플리케이션 개발자에 의해 생성될 수 있고 메시 네트워크(100)의 사용자에게 이용 가능하게 될 수 있다. 일 실시예에서, 각각의 애플리케이션 개발자는 메시 네트워크(100) 상의 사용자에 애플리케이션을 제공하도록 허용되기 전에 등록 프로세스를 거쳐야한다. 등록시, 이 실시예의 애플리케이션 개발자에게는 개발자 키 서명이 발행되고, 상기 키 서명은 연이어 애플리케이션의 프로그램 코드에 내장된다. 이 실시예에서, 블록(810)은 마이크로 프로세서(202)가 메모리(210)의 애플리케이션 저장 위치(244)에서 애플리케이션을 위한 프로그램 코드에 있는 개발자 키 서명을 판독하고 메시 서비스(310)에 의해 제공된 메시 포트 바인딩 API 함수에 대한 호출에서 상기 개발자 키 서명을 포함시키도록 더 지시한다. 메시 서비스(310)에 의한 메시 포트 바인딩 API 기능에 대한 호출의 처리는 도 10b의 블록(1034-1040)을 참조하여 본 명세서에서 나중에 설명된다.

[0061] 애플리케이션 이벤트 처리 프로세스(800)는 블록(812)에서 계속되며, 이는 마이크로 프로세서(202)가 애플리케이션이 메시 네트워크(100)를 통한 데이터의 안전한 전송을 위해 암호화 키 교환을 요청했는지 여부를 결정하도록 지시한다. 암호화 키 교환이 블록(812)에서 요청된 경우, 마이크로 프로세서(202)는 암호화 키 교환을 개시하는 메시 서비스 API 기능이 호출되는 블록(814)으로 지시된다. 메시 서비스(310)는 암호화 키를 교환하기 위해 호출되는 API 기능을 구현하며, 호출될 때, 자신의 암호화 키를 지정된 uuid를 갖는 타겟 디바이스로 전송한다. 디바이스가 암호화 키를 수신하면, 이는 로컬 암호화 키 저장 위치(252)에 저장되고, 그 자신의 암호화 키를 키 교환을 시작한 디바이스로 다시 전송함으로써 응답한다. 개시 디바이스에 의해 수신될 때, 암호화 키는 로컬 암호화 키 저장 위치(252)에 저장된다.

[0062] 블록(812)에서 암호화 키 교환이 요청되지 않은 경우, 애플리케이션 이벤트 처리 프로세스(800)는 블록(816)에서 계속되어, 마이크로 프로세서(202)가 관련 애플리케이션이 메시 네트워크(100)를 통해 데이터를 메시 서비스(310)로 전송하고자 하는 지를 결정하도록 한다. 예로서, 채팅 애플리케이션(302)에 대해, 데이터는 텍스트 및/또는 오디오 콘텐츠, 이미지 또는 비디오 콘텐츠와 같은 다른 콘텐츠를 포함하는 채팅 메시지일 수 있다. 메시 애플리케이션(302 및 304)으로부터 메시 서비스(310)로의 콘텐츠 데이터의 전송은 애플리케이션 인터페이스(306 및 308)에 의해 관리된다. 블록(816)은 마이크로 프로세서(202)가 콘텐츠 데이터를 메시 서비스(310)로 전송하기 위한 메시 서비스 기능을 호출하도록 지시한다. 메시 서비스에 대한 호출은 상기 호출에 uuid를 포함시킴으로써 데이터의 의도된 목적지(즉, 도 7의 하나 이상의 디바이스(700 또는 702))를 식별한다.

[0063] 위에서 언급 한 바와 같이, 이미지 또는 채팅 메시지와 같은 더 작은 크기의 콘텐츠 데이터에 대해, 메신저 송수신 기능은 메시 서비스(310) 로의 전송을 위해 사용될 수 있다. 블록(818)은 마이크로 프로세서(202)가 메신저 송수신 기능을 통해 메시 서비스(310)로 전달하기 위한 복수의 데이터 청크(chunk)로 데이터를 분할하도록 지시한다. 디바이스(300)상의 제 2 애플리케이션(304)으로부터의 컨텐츠 데이터 전송의 예가 도 9에 900으로 개략적으로 도시되어 있다. 도 9를 참조하면, 5MB의 이미지(902)가 904에 도시된 n개의 청크로 분할되는 것으로 도시되어 있다. 일 실시예에서, 애플리케이션 인터페이스들(306 및 308)은 데이터 청크들에 대한 최대 데이터 크기를 정의하는 데이터 청크 크기 제한(MAX_CHUNK)을 구현할 수 있다.

[0064] 도 8을 다시 참조하면, 이 실시예에서, 블록(818)은 또한 마이크로 프로세서(202)가 이미지(902)의 전체 데이터 길이를 "Chunk 1"의 제 1 필드로서 기록하도록 지시한다. 따라서, 제 1 "Chunk 1"는 전송 데이터 길이를 메시 서비스(310)에 통지하는 역할을 하며, 나머지 데이터 청크(2 내지 n)는 데이터만을 포함할 것이다. 그런 후, 블록(818)은 마이크로 프로세서(202)가 메신저 송수신 기능을 사용하여 청크(1)를 메시 서비스(310)로 전송하도록 지시한다.

[0065] 블록(820)은 마이크로 프로세서(202)가 메시 서비스(310)가 다음 청크를 수신할 준비가 되었는지 여부를 결정하도록 지시한다. 컨텐츠 데이터를 전송하기 위해 애플리케이션(302 또는 304)으로부터 호출이 수신될 때, 메시 서비스(310)는 데이터 흐름을 위해 애플리케이션 버퍼(320)(도 9에 도시됨)를 할당한다. 애플리케이션 버퍼(320)는 메모리(210)의 메시 서비스 애플리케이션 버퍼 위치(256)에 보유된다. 더 상세히 후술될 바와 같이, 청크(1)를 수신하면, 메시 서비스(310)는 메시 네트워크(100)를 통한 추가 데이터의 전송을 위해 적용 가능한 애플리케이션 버퍼(320)에 공간이 남아 있는지를 결정하고 이에 따라 적용가능한 애플리케이션 인터페이스(306 또는 308)에 알린다. 블록(820)에서, 메시 서비스(310)가 아직 더 많은 데이터를 수신할 준비가 되지 않았다면, 블록(820)은 마이크로 프로세서(202)가 블록(820)을 반복하도록 지시한다.

[0066] 블록(820)에서, 메시 서비스(310)가 더 많은 데이터를 수신할 준비가 되면, 프로세스는 블록(822)에서 계속되고 다음 청크(이 경우 chunk 2)가 전송된다. 블록(824)은 마이크로 프로세서(202)로 하여금 추가 청크가 계속 전송될지 여부를 결정하도록 지시하고, 이 경우 마이크로 프로세서는 다시 블록(820)으로 지시된다. 블록(824)에서,

추가 청크가 전송되지 않으면, 마이크로 프로세서는 406으로 되돌아가 다음 이벤트를 기다린다.

[0067] 블록(816)에서, 메시 네트워크(100)를 통해 데이터를 전송하라는 애플리케이션으로부터의 요청이 없는 경우, 프로세스는 도 8b의 블록(826)에서 계속된다. 도 8b를 참조하면, 블록(826)은 메시 서비스(310)가 peerChanged 이벤트를 발행했는지 여부를 마이크로 프로세서(202)가 결정하도록 지시한다. peerChanged 이벤트는 메시 서비스(310)가 메시 네트워크(100)상의 디바이스들(102-112) 중 하나의 상태가 변경되었음을 감지할 때마다 생성된다.

[0068] 블록(826)에서 peerChanged 이벤트가 수신되지 않으면, 애플리케이션 이벤트 처리 프로세스(800)는 블록(828)에서 계속되며, 여기서 마이크로 프로세서(202)는 메시 서비스(310)가 디바이스 상에 실행되는 애플리케이션 중 하나와 관련된 dataReceived 이벤트를 발행했는지 여부를 결정하도록 지시된다. 예를 들어, 메시 포트(6000)와 관련된 데이터가 메시 네트워크(100)를 통해 수신될 때, dataReceived 이벤트가 메시 서비스(310)에 의해 애플리케이션 인터페이스(306)로 전송된다.

[0069] peerChanged 이벤트가 블록(826)에서 수신되거나 dataReceived 이벤트가 블록(828)에서 수신되면, 프로세스는 블록(830)에서 계속되고, 이는 마이크로 프로세서(202)가 이벤트를 처리하게 한다. 각각의 애플리케이션은 일반적으로 메모리(210)의 저장 위치(244)에 저장된 애플리케이션에 대한 코드에 의해 정의된 구성된 거동에 따라 데이터 및 다른 이벤트를 처리하고 디스플레이할 것이다. 예를 들어, 채팅 애플리케이션(302)은 디바이스들(102-112) 중 하나의 사용자가 채팅 애플리케이션을 종료한 것을 나타내는 *peerChanged* 통지를 수신한 경우, 디바이스(300)의 디스플레이는 그 사용자와 관련된 리스팅을 제거하도록 업데이트될 수 있거나 대안으로 사용자가 비활성임을 표시할 수 있다. 컨텐츠 데이터가 수신되는 경우, 디바이스(300)의 디스플레이는 채팅 메시지 또는 메시 네트워크(100)의 다른 사용자에 의해 전송된 다른 컨텐츠를 디스플레이하도록 업데이트될 수 있다. 블록(830)은 마이크로 프로세서(202)를 프로세스의 블록(406)으로 지시해 다음 이벤트를 기다리게 한다.

[0070] 블록(828)에서, dataReceived 이벤트가 수신되지 않으면, 프로세스는 블록(832)에서 계속된다. 상술한 바와 같이, 애플리케이션 개발자는 메시 네트워크(100)상에서 사용하기 위한 애플리케이션을 제공하도록 허용되기 전에 등록 프로세스를 거쳐야 할 수 있다. 일 실시예에서, 테스트 및 성능 평가를 위해 프로그래밍 방식으로 디바이스의 네트워크 역할을 설정할 수 있도록 등록된 애플리케이션 개발자에게 추가 기능이 노출될 수 있다. 필요한 기능은 메모리(210)의 애플리케이션 저장 위치(244)에 저장하기 위한 개발자 버전의 코드를 통해 제공될 수 있다. 일 실시예에서, 애플리케이션이 네트워크를 확립하기 위한 자신의 자원 중 어느 하나를 공유하지 않고도 메시 네트워크(100)에 참여하도록 프로그래밍될 수 있기 때문에, 개발자 코드는 디바이스가 라이브 메시 네트워크(100)에 참여하는 것을 제한할 수 있다. 메시 네트워크(100)의 성공적인 확립은 무선 통신 범위 내에 있지 않은 다른 디바이스들 간에 메시지의 전송을 가능하게 하도록 메시 네트워크 내의 디바이스들의 참여에 의존한다.

[0071] 따라서, 블록(832)은 (상술한 바와 같은) 유효한 개발자 키 서명을 갖는 등록된 애플리케이션 개발자에게 제공되는 코드의 버전을 실행하는 디바이스에서만 액세스할 수 있다. 블록(832)은 관련 애플리케이션이 메시 네트워크(100)에 대한 상태 정보를 요청했는지 여부를 마이크로 프로세서(202)가 결정하도록 지시한다. 이러한 상태 정보는 디바이스 및 네트워크 역할(라우팅 모드, 클라이언트 모드, 액세스 포인트 모드 등)의 리스트, 특정 기기에 대한 연결 정보 및 기타 성능 평가 측정 항목을 포함할 수 있다. 블록(832)에서, 상태 정보가 요청된 경우, 블록(834)은 마이크로 프로세서(202)가 메시 서비스(310)로부터 상태 정보 데이터를 요청하도록 지시한다. 상태 정보 데이터는 상기 정보 타입 중 어느 하나 또는 모두를 포함하는 구조화된 데이터 메시지의 형태로 제공될 수 있다. 블록(834)은 마이크로 프로세서(202)를 프로세스(400)의 블록(406)으로 다시 지시해 다음 이벤트를 기다리게 한다.

[0072] 블록(832)에서 상태 정보에 대한 요청이 수신되지 않으면, 프로세스는 블록(836)에서 계속되며, 이는 유효한 개발자 키 서명을 갖는 등록된 애플리케이션 개발자에게 제공된 코드 버전을 실행하는 디바이스에서만 액세스 가능하다. 블록(836)은 관련 애플리케이션이 특정 메시 네트워크 통신 파라미터를 설정하기 위해 액세스를 요청했는지 여부를 마이크로 프로세서(202)가 결정하도록 지시한다. 예를 들어, 애플리케이션은 테스트 네트워크 설정과 관련된 여러 디바이스에 대해 네트워크 역할, 무선 SSID 또는 Bluetooth 식별자를 프로그래밍 방식으로 조작할 수 있길 바란다. 블록(836)에서 애플리케이션에 의해 그러한 요청이 이루어진 경우, 블록(838)은 마이크로 프로세서(202)가 그러한 기능을 제공하는 메시 서비스 기능에 호출을 전송하도록 지시한다. 블록(838)은 마이크로 프로세서(202)를 프로세스(400)의 블록(406)으로 다시 지시해 다음 이벤트를 기다린다.

[0073] 블록(836)에서 특정 메시 네트워크 통신 파라미터를 설정하기 위한 요청이 수신되지 않으면, 프로세스는 블록(840)에서 계속된다. 블록(840)은 애플리케이션의 사용자가 도 5에 도시된 사용자 선호 인터페이스(500)의 디스플레이를 요청했는지를 마이크로 프로세서(202)가 결정하도록 지시한다. 블록(842)은 마이크로 프로세서(202)로

하여금 사용자 선호 인터페이스(500)가 예를 들어, 메시 네트워크상에서 디바이스의 네트워크 역할의 변화와 같은 사용자 입력을 수신하도록 디스플레이하게 지시한다. 그 후, 블록(844)은 마이크로 프로세서(202)로 하여금 사용자가 용자 선호 중 어느 하나를 변경했는지 여부를 결정하도록 지시하고, 이 경우 프로세스는 블록(846)으로 계속된다. 블록(846)은 마이크로 프로세서(202)가 브로드캐스트 인텐트 프로토콜을 사용하여 사용자 선호 인터페이스(500)에서 수신된 변경된 사용자 선호를 메시 서비스(310)에 통지하도록 지시한다. 그런 다음, 블록(846)은 마이크로 프로세서(202)를 다시 프로세스(400)의 블록(406)으로 지시해 다음 이벤트를 기다린다.

[0074] 블록(844)에서, 사용자 선호가 수신되지 않으면, 마이크로 프로세서(202)는 사용자 선호 인터페이스(500)를 닫도록 지시된 후 프로세스(400)의 블록(406)으로 다시 지시되어 다음 이벤트를 기다린다. 블록(840)에서, 사용자 선호 인터페이스(500)의 디스플레이가 요청되지 않은 경우, 블록(840)은 마이크로 프로세서(202)를 프로세스(400)의 블록(406)으로 다시 지시해 다음 이벤트를 기다린다.

[0075] 애플리케이션 이벤트 처리 프로세스(800)는 따라서 애플리케이션 인터페이스(306, 308, 700 및 712)가 각각의 애플리케이션 및 각 디바이스에 실행되는 메시 서비스(310, 708, 및 714)로부터 발생하는 이벤트뿐만 아니라 메시 포트 식별에 기초하여 애플리케이션 인터페이스로 전달되는 다른 디바이스에서 발생하는 이벤트를 처리하기 위해 디바이스(102-112)의 각각의 마이크로 프로세서가 필요한 기능을 구현하도록 지시한다.

[0076] 메시 서비스(310)(및 메시 서비스(708 및 714))가 도 6에 도시된 프로세스(600)의 블록(620)에서 수신된 이벤트를 처리하도록 메시 서비스 기능을 구현하기 위해 프로세서 회로(200)에 의해 실행되는 메시 서비스 프로세스가 도 10에 1000으로 도시되어 있다. 도 10a를 참조하면, 메시 서비스 프로세스(1000)는 블록(1002)에서 시작하고, 이는 마이크로 프로세서(202)가 메시 네트워크(100)를 통해 다른 디바이스로부터 peerChanged 이벤트가 수신되었는지를 결정하도록 지시한다. 블록(1002)에서, peerChanged 이벤트가 수신된 경우 프로세스는 블록(1004)에서 계속되며, 이는 마이크로 프로세서(202)가 peerChanged 이벤트와 관련된 메시 포트를 결정하고 대응하는 메시 포트를 갖는 애플리케이션이 peerChanged 이벤트를 통지하게 하게 한다. 예를 들어, peerChanged 이벤트가 디바이스(702)로부터 메시 서비스(310)에서 수신되면, 메시 포트 식별자는 5000이 되고 peerChanged 이벤트는 제 2 애플리케이션(304)의 애플리케이션 인터페이스(308)로 전송될 것이다. 상술한 바와 같이, 애플리케이션 인터페이스(308)는 도 8에 도시된 애플리케이션 이벤트 처리 프로세스(800)의 블록(826)에 따라 peerChanged 이벤트를 처리한다. 그 후, 블록(1004)은 마이크로 프로세서(202)가 프로세스(600)의 블록(620)으로 복귀시켜 추가 이벤트를 기다리도록 지시한다.

[0077] 블록(1002)에서, peerChanged 이벤트가 수신되지 않으면, 마이크로 프로세서(202)는 블록(1006)으로 지시되고 콘텐츠 데이터가 메시 네트워크(100)를 통해 다른 디바이스로부터 수신되었는지를 결정하도록 지시된다. 블록(1006)에서 콘텐츠 데이터가 수신되었다면, 마이크로 프로세서(202)는 블록(1008)으로 지시되고, 이는 마이크로 프로세서가 콘텐츠 데이터와 관련된 메시 포트를 결정하게 한다. 블록(1010)은 결정된 메시 포트에 대응하는 애플리케이션이 디바이스상에서 실행 중인지 여부를 마이크로 프로세서(202)가 결정하도록 지시한다. 블록(1012)은 마이크로 프로세스(202)가 메시 네트워크(100)를 통해 데이터를 수신하고 상기 데이터를 상기 개시된 프로세스 간 통신 프로토콜 중 하나를 사용하여 적용 가능한 애플리케이션으로의 전달하게 한다. 일 실시예에서, 데이터는 메모리(210)의 애플리케이션 버퍼 위치(254)에서 수신 데이터 버퍼에 기록될 수 있다. 수신 데이터 버퍼는 적용 가능한 애플리케이션 인터페이스(306 또는 308)에 의해 관리된다. 블록(1012)은 또한 마이크로 프로세서(202)가 메시 네트워크(100)를 통한 데이터 전송이 완료될 때까지 대기하고 그런 후전송이 완료되었음을 적용 가능한 애플리케이션에 통지하게 지시한다. 적용 가능한 애플리케이션은 그 후 애플리케이션 버퍼 위치(254)의 수신 데이터 버퍼에서 데이터를 처리할 수 있다. 다른 실시예에서, 수신 데이터 버퍼가 전송 완료 전에 데이터 전송에 의해 채워지면, 블록(1012)은 또한 마이크로 프로세서(202)가 여전히 수신되어야 할 추가 데이터 양을 적용 가능한 애플리케이션에 통지하게 지시할 수 있다. 따라서, 큰 데이터 전송으로 인해 애플리케이션 수신 데이터 버퍼가 압도되는 것이 방지된다. 블록들(1006, 1008, 1010 및 1012)은 따라서 마이크로 프로세서(202)가 메시 포트에 대응하는 애플리케이션을 위해 애플리케이션 인터페이스로 데이터를 전달하도록 지시한다. 예를 들어, 컨텐츠 데이터가 디바이스(702)상에서 실행되는 제 2 애플리케이션(710)으로부터 수신되면, 메시 포트는 5000이 되고 컨텐츠 데이터는 제 2 애플리케이션(304)과 연관된 애플리케이션 인터페이스(308)로 전달될 것이다. 그런 후, 블록(1012)은 마이크로 프로세서(202)가 프로세스(600)의 블록(620)으로 복귀시켜 추가 이벤트를 기다리기 한다.

[0078] 블록(1010)에서, 결정된 메시 포트에 대응하는 애플리케이션이 디바이스상에서 실행되고 있지 않으면, 마이크로 프로세서(202)는 블록(1014)으로 지시되고, 여기서 마이크로 프로세서는 본 명세서에 후술된 라우팅 프로토콜에 따라 메시 네트워크(100)를 통해 콘텐츠를 전달하도록 지시된다. 그 후, 블록(1014)은 마이크로 프로세서(202)

가 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리게 지시한다.

[0079] 블록(1006)에서, 콘텐츠 데이터가 수신되지 않으면, 메시 서비스 프로세스(1000)는 블록(1016)에서 계속되며, 이는 암호화 키 교환 요청이 다른 디바이스로부터 메시 네트워크(100)를 통해 수신되었는지를 마이크로 프로세서(202)가 결정하도록 지시한다. 암호화 키 교환 요청이 수신되었다면, 블록(1016)은 마이크로 프로세서(202)를 블록(1018)으로 지시한다. 암호화 키 요청은 요청 디바이스의 공개 키를 포함할 수 있으며, 이 경우 블록(1018)은 마이크로 프로세서(202)가 공개 암호화 키를 메모리(210)의 암호화 키 저장 위치(252)에 추가하도록 지시한다. 블록(818)은 또한 마이크로 프로세서(202)가 메시 네트워크(100)를 통해 디바이스 공개 암호화 키를 암호화 키 교환 요청을 발행한 디바이스로 전송하도록 지시한다. 마지막으로, 블록(818)은 디바이스가 암호화된 통신을 요청했음을 적용 가능한 애플리케이션(302 또는 304)에 마이크로 프로세서(202)가 통지하도록 지시한다. 암호화된 통신은 블록(812)에서 메시 네트워크(100)상의 개시 디바이스와 타켓 디바이스 사이의 애플리케이션에 의해 요청된 암호화 키 교환에 의존한다. 일단 키 교환이 완료되고 키가 로컬 암호화 키 저장 위치(252)에 저장되면, 상기 키는 공통 메시 포트를 갖는 타겟 디바이스상의 애플리케이션으로 애플리케이션들 간에 전송된 데이터를 암호화하는데 사용될 수 있다. 그렇지 않으면, 전송된 데이터는 상기 데이터가 암호화되었음을 표시하도록 설정된 바이트 값을 갖는 것을 제외하고는 본 명세서에 후술된 바와 같이 데이터가 전송될 수 있다(데이터 패킷(1300)의 isEncrypted 필드(1326)가 도 12에 도시되어 있다. 수신 측에서, 타겟 디바이스는 isEncrypted 필드(1326)를 판독하고, 그 로컬 암호화 키 저장 위치(252)에 관련 암호화 키를 찾는다. 발견되면, 암호화 키는 데이터를 해독하는데 사용된다).

[0080] 블록(1016)에서, 암호화 키 교환 요청이 수신되지 않은 경우, 메시 서비스 프로세스(1000)는 블록(1020)에서 계속되며, 이는 애플리케이션(302 또는 304) 중 하나가 도 5에 도시된 사용자 선호 인터페이스(500) 상에 네트워크 역할 선택기 버튼 컨트롤(508)을 통해 디바이스(300)에 대한 네트워크 역할을 변경했는지를 마이크로 프로세서(202)가 결정하도록 지시한다. 블록(1020)에서 네트워크 역할의 변화가 수신되지 않은 경우, 프로세스는 블록(1022)에서 계속되고, 이는 마이크로 프로세서(202)가 애플리케이션(302 또는 304) 중 하나가 사용자 선호 인터페이스(500)상의 "Wi-Fi 메시" 컨트롤(504) 또는 "블루투스 메시" 컨트롤(506)을 통한 Wi-Fi 또는 블루투스 통신을 비활성화 또는 활성화했는지 결정하게 지시한다.

[0081] 블록(1020 및 1022)의 경우, 네트워크 역할 또는 통신 설정이 변경된 경우, 프로세스는 블록(1024)에서 계속되며, 이는 마이크로 프로세서로 하여금 적용 가능한 메시 통신 설정을 업데이트하도록 지시한다. 메시 통신 설정은 메시 서비스(310)가, 예를 들어 도 2에 도시된 무선 라디오(216)의 Wi-Fi 또는 블루투스 기능을 비활성화할 수 있게 하는 것과 같이 메시 네트워크(100)와 상호 작용하는 방법을 결정한다. 따라서, 디바이스(300)의 사용자는 네트워크 역할 및 무선 라디오(216)의 사용을 선택할 능력이 있다. 예를 들어, 디바이스(300)가 배터리 충전량이 낮으면, 사용자는 Wi-Fi 통신을 비활성화하여 배터리 전력을 보존하면서 블루투스 통신이 계속 진행되도록 할 수 있다. 그런 후, 블록(1024)은 마이크로 프로세서(202)로 하여금 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리게 지시한다.

[0082] 블록(1022)에서, 통신 설정에 변화가 없다면, 프로세스는 블록(1026)에서 계속된다. 블록(1026)은 마이크로 프로세서(202)로 하여금 애플리케이션이 도 5의 사용자 선호 인터페이스상의 "메시 서비스" 컨트롤(502)을 통해 메시 서비스(310)를 비활성화했는지 여부를 결정하도록 지시한다. 메시 서비스(310)가 비활성화된 경우, 블록(1026)은 마이크로 프로세서(202)를 블록(1028)으로 지시하고, 이는 마이크로 프로세서가 메시 네트워크(100)를 통해 "제거된" 상태를 갖는 peerChanged 통지를 생성하여 전송하도록 지시하여, 메시 네트워크(100)상의 다른 디바이스가 상기 디바이스(300)가 더 이상 네트워크에서 이용 가능하지 않을 것임이 업데이트될 수 있다. 블록(1028)은 마이크로 프로세서(202)가 모든 바운드 메시 포트(즉, 포트 (5000 및 6000))를 해제하고 메시 서비스(310)를 셧다운하도록 더 지시한다. 메시 서비스가 종료되면, 디바이스(300)는 더 이상 메시 네트워크(100)에 참여할 수 없다. 일 실시예에서, 애플리케이션(302 및 304)은 사용자가 메시 서비스(310)를 다시 활성화하기로 결정한 경우에 실행중인 상태로 유지될 수 있다. 대안으로, 애플리케이션(302 및 304)은 메시 서비스(310)와 동시에 셧다운될 수 있다. 블록(1026)에서, 메시 서비스(310)가 비활성화되지 않았으면, 메시 서비스 프로세스(1000)는 도 10b의 블록(1030)에서 계속된다.

[0083] 도 10b를 참조하면, 블록(1030)은 상술한 바와 같이 개발자 버전의 코드에서만 구현되고, 애플리케이션(302 또는 304) 중 어느 하나가 상태 정보를 요청했는지를 결정하도록 마이크로 프로세서(202)에 지시한다. 상태 정보에 대한 요청은 이전에 애플리케이션 이벤트 처리 프로세스(800)의 블록(832)과 관련하여 설명되었다. 블록(1030)에서, 상태 정보에 대한 요청이 애플리케이션 인터페이스(306 및 308) 중 하나에 의해 발행된 경우, 프로세스는 블록(1032)에서 계속되고, 이는 요청된 상태 정보를 정의하는 구조화된 데이터를 애플리케이션 인터페이

스로 전송하도록 마이크로 프로세서(202)에 지시한다. 블록(1032)은 마이크로 프로세서(202)가 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리게 지시한다.

[0084] 블록(1030)에서, 상태 정보에 대한 요청이 수신되지 않은 경우, 프로세스는 블록(1034)에서 계속되며, 이는 마이크로 프로세서(202)로 하여금 메시 포트를 바인딩하기 위한 요청이 애플리케이션 인터페이스(306 및 308) 중 하나로부터 수신되었는지를 결정하도록 지시한다. 메시 포트 바인딩 요청이 수신되면, 프로세스는 블록(1036)에서 계속되며, 이는 마이크로 프로세서(202)로 하여금 애플리케이션 프로세서(306 또는 308)에 의해 제공된 개발자 키 서명이 바인딩 요청시 특정 메시 포트에 바인딩하기 위해 인증되는지를 결정하도록 지시한다. 메시 서비스(310)는 개발자 키 서명에 대응하는 키 서명 데이터와 함께 다양한 어플리케이션에 할당된 메시 포트 테이블을 유지한다. 테이블은 메모리(210)의 메시 포트 테이블 위치(250)에 유지된다. 개발자 키 서명이 블록(1036)에서 인증되면, 메시 포트 바인딩을 요청하는 애플리케이션이 요청된 메시 포트에 바인딩되도록 허용되고 메시 포트 테이블이 업데이트되어 특정 메시 포트의 성공적인 바인딩을 반영한다. 예로서, 메시 포트(6000)가 할당된 애플리케이션(302)은 메시 서비스(310)가 개발자 키 서명이 이 특정 포트에 바인딩될 수 있음을 나타내는 대응하는 키 서명을 갖는 경우에만 이 메시 포트에 바인딩하는 것이 허용될 것이다(6000). 비록 디바이스(300)가 애플리케이션(304)을 메시 포트(5000)에 성공적으로 바인딩할 수 있지만, 애플리케이션(302)은 메시 포트(5000)에 바인딩하는 것이 허용되지 않을 것이다. 이는 트래픽을 분리하고 애플리케이션들이 다른 애플리케이션용으로 의도된 데이터 트래픽에 악의적으로 수신하는 것을 방지하는 이점을 갖는다. 블록(1038)은 또한 마이크로 프로세서(202)로 하여금 메시 포트 바인딩 요청을 시작한 애플리케이션 인터페이스에 통지를 전송하도록 지시한 다음, 마이크로 프로세서가 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리도록 지시한다.

[0085] 블록(1036)에서 개발자 키 서명이 인증되지 않으면, 블록(1040)은 마이크로 프로세서로 하여금 메시 포트 바인딩 요청을 시작한 애플리케이션 인터페이스에 이 효과에 대한 통지를 전송하도록 지시한 다음 마이크로 프로세서가 프로세스(600)의 블록(620)으로 복귀시켜 추가 이벤트를 기다리게 지시한다.

[0086] 블록(1034)에서 메시 포트를 바인딩하기 위한 요청이 수신되지 않은 경우, 블록(1042)은 마이크로 프로세서(202)로 하여금 메시 포트 바인딩 해제 요청이 수신되었는지를 결정하도록 지시한다. 메시 포트 바인딩 해제 요청이 수신되면, 블록(1042)은 마이크로 프로세서(202)로 하여금 바운드 메시 포트를 해제하고 메모리(210)의 메시 포트 테이블 위치(250)에 저장된 메시 포트 테이블을 업데이트하도록 지시한다. 블록(1044)은 그런 후 마이크로 프로세서로 하여금 프로세스(600)의 블록(620)으로 되돌아가 다른 이벤트를 기다리도록 지시한다.

[0087] 블록(1042)에서, 메시 포트 바인딩 해제 요청이 수신되지 않은 경우, 메시 서비스 프로세스(1000)는 블록(1046)에서 계속된다. 블록(1046)은 마이크로 프로세서(202)로 하여금 콘텐츠 데이터 청크가 각각의 애플리케이션(302 및 304)의 애플리케이션 인터페이스(306 및 308) 중 하나로부터 메시 서비스(310)에 의한 전송을 위해 수신되었는지를 결정하도록 지시한다. 콘텐츠 데이터 청크가 수신되면, 블록(1046)은 마이크로 프로세서(202)로 하여금 애플리케이션과 uuid에 의해 식별된 목적지 사이의 데이터 흐름에 대한 메시 서비스 애플리케이션 데이터 버퍼 위치(256)에 애플리케이션 데이터 버퍼(즉, 도 9에 도시된 애플리케이션 버퍼(302))를 할당하도록 지시한다. 블록(1046)은 또한 마이크로 프로세서(202)로 하여금 데이터를 애플리케이션 버퍼(320)에 기록하도록 지시한다.

[0088] 상술한 바와 같이, 애플리케이션 버퍼는 애플리케이션이 콘텐츠를 전송하고자하는 메시 네트워크(100)상의 특정 애플리케이션 및 특정 목적지에 대해 애플리케이션 버퍼가 할당된다. 이와 같이 고유 애플리케이션 버퍼는 디바이스(300)상의 제 2 애플리케이션(304)이 디바이스(702)상의 제 2 애플리케이션(710)으로 데이터를 전송하는 데 할당될 것이다. 제 2 애플리케이션(304)이 또한 다른 디바이스로 데이터를 전송하는 경우, 메시 서비스 애플리케이션 버퍼 위치(256) 내에 추가 애플리케이션 버퍼가 할당될 것이다. 따라서, 각각의 애플리케이션 버퍼는 발신 애플리케이션, 상기 발신 애플리케이션의 메시 포트 및 목적지 디바이스와 연관된다. 다시 말해서, 각각의 애플리케이션 버퍼는 소스 디바이스(예를 들어, 디바이스(300))로부터 목적지 디바이스(예를 들어, 디바이스(702))로의 특정 데이터 흐름과 관련되고 또한 특정 메시 포트와 관련된다.

[0089] 블록(1046)은 마이크로 프로세서(202)를 블록(1048)으로 지시하고, 이는 블록(1048)으로 하여금 추가 데이터 청크가 계속 전송될지 여부를 결정하도록 지시한다. 애플리케이션 이벤트 처리 프로세스(800)의 블록(818)과 관련하여 상술한 바와 같이, 제 1 데이터 청크는 데이터 길이를 데이터 전송의 전체 길이를 정의하는 제 1 필드로서 포함한다. 블록(1048)에서 전송을 위한 단일 데이터 청크만이 있다면(즉, 데이터 길이가 MAX_CHUNK 파라미터보다 작으면), 마이크로 프로세서는 블록(1050)으로 지시된다. 블록(1050)은 마이크로 프로세서(202)로 하여금 데이터 청크를 데이터 흐름과 관련된 애플리케이션 버퍼(320)에 추가하도록 지시한다. 블록(1050)은 마이크로 프

로세서(202)로 하여금 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리도록 지시한다.

[0090] 블록(1048)에서 전송을 위한 하나 이상의 데이터 청크가 있는 경우(즉, 데이터 길이가 MAX_CHUNK 파라미터보다 큰 경우), 마이크로 프로세서는 블록(1052)으로 지시된다. 블록(1052)은 마이크로 프로세서(202)로 하여금 특정 데이터 길이를 갖는 콘텐츠 데이터에 대해 애플리케이션 버퍼(320) 내에 공간이 있는지를 결정하도록 지시한다. 디바이스(300)는 일반적으로 제한된 메모리 크기를 가질 것이기 때문에, 메시 서비스 애플리케이션 버퍼 위치(256) 내의 애플리케이션 버퍼는 디바이스의 자원 과부하를 방지하기 위해 적당한 크기로 유지될 필요가 있을 것이다. 애플리케이션 버퍼의 나머지 공간이 MAX_CHUNK 파라미터의 두 배 미만인 경우, 블록(1052)은 마이크로 프로세서(202)를 블록(1054)으로 지시하고 마이크로 프로세서는 데이터 청크를 애플리케이션 버퍼(320)에 저장하도록 지시되고, 컨텐츠 데이터 전송 요청을 발생시킨 애플리케이션 인터페이스에 메시 서비스(310)가 더 많은 데이터 청크를 수신할 준비가 되지 않았다는 것을 통지하도록 지시한다. 그 후, 블록(1050)은 마이크로 프로세서(202)가 프로세스(600)의 블록(620)으로 되돌아 가서 추가 이벤트를 기다리도록 지시한다.

[0091] 블록(1052)에서, 애플리케이션 버퍼의 나머지 공간이 MAX_CHUNK 파라미터의 두 배 이상인 경우, 마이크로 프로세서(202)는 블록(1056)으로 지시되고, 여기서 마이크로 프로세서는 데이터 청크를 애플리케이션 버퍼(320)에 저장하도록 지시되고, 콘텐츠 데이터 전송 요청을 발생시킨 애플리케이션 인터페이스에 메시 서비스(310)가 더 많은 데이터 청크를 수신할 준비가 되었다는 것을 통지하도록 지시한다. 그 후, 블록(1056)은 마이크로 프로세서(202)로 하여금 프로세스(600)의 블록(620)으로 돌아가서 추가 이벤트를 기다리도록 지시한다. 블록(1046 내지 1056)은 따라서 각각의 데이터 청크가 애플리케이션(302 및 304)의 애플리케이션 인터페이스(306 또는 308)로부터 메시 서비스(310)에서 수신될 때 반복된다.

[0092] 도 8과 관련하여 위에서 개시된 바와 같이, 애플리케이션 이벤트 처리 프로세스(800)의 블록(818 내지 824)은 메시 네트워크(100)를 통한 전송을 위해 데이터가 애플리케이션(302 및 304)으로부터 메시 서비스(310)로 얼마나 빨리 푸시되는지를 관리하도록 마이크로 프로세서(202)에 지시한다. 메시 서비스 프로세스(1000)의 블록(1054 및 1056)은 애플리케이션 인터페이스(306 및 308)로부터 메시 서비스로의 데이터 청크의 전송 속도를 컨트롤하기 위해 애플리케이션 인터페이스(306 및 308)에 신호를 제공한다.

[0093] 도 9를 다시 참조하면, 상술한 바와 같이, 이미지(902)는 따라서 제 2 애플리케이션(304)의 애플리케이션 인터페이스(308)에 의해 904에 도시된 바와 같이 n개의 데이터 청크로 분할된다. 메신저 송수신 프로세스 간 통신 프로토콜(312)은 애플리케이션 이벤트 처리 프로세스(800)의 블록(816-824) 및 메시 서비스 프로세스(1000)의 블록(1046-1056)과 관련하여 상술한 바와 같이 애플리케이션 인터페이스(308)와 메시 서비스(310) 사이의 청크(904)를 송신하는 데 사용된다. 따라서, 이들 블록은 애플리케이션(304)과 메시 서비스(310) 사이의 컨텐츠 데이터를 전송하고 메시 서비스가 메시 네트워크(100)를 통한 전송을 위한 데이터로 넘치는 것을 방지하기 위해 협력한다.

[0094] 따라서, 메시 서비스(310)는 메모리(210)의 메시 서비스 애플리케이션 버퍼 위치(256)에서 복수의 애플리케이션 및 목적지 특정 데이터 버퍼에서 메시 네트워크(100)를 통한 전송을 위한 데이터를 축적한다. 데이터는 데이터 청크로 수신되지만, 연속된 일련의 데이터 바이트로 애플리케이션 버퍼에 저장된다. 메시 서비스 프로세스(1000)의 블록(1046)에서, 전송을 위한 컨텐츠 데이터 청크가 적용 가능한 애플리케이션으로부터 수신되고 특정 데이터 흐름에 대한 메시 서비스 애플리케이션 버퍼 위치(256)에서 애플리케이션 버퍼(256) 중 하나에 기록될 때, 메시 서비스(310)는 전송을 위해 데이터를 처리한다.

[0095] 안정적인 데이터 전송

[0096] 신뢰할 수 있는 데이터 전송 프로세스 실시예가 도 11에 도시되어 있다. 신뢰할 수 있는 데이터 전송 프로세스는 적어도 소스 디바이스(예를 들어 도 7에 도시된 디바이스(300)) 및 목적지 디바이스(예를 들어 디바이스(702))를 포함하고, 하나 이상의 라우팅 디바이스(예컨대, 디바이스(700))를 더 포함할 수 있다. 신뢰할 수 있는 데이터 전송은 일반적으로 데이터가 목적지 디바이스(702)에 의해 수신되었음을 확인하기 위해 소스 디바이스(300)에 의해 메시 네트워크(100)를 통한 데이터 흐름의 모니터링을 포함한다. 다른 실시예에서, 본 명세서에서 후술된 바와 같이, 신뢰할 수 없는 전송 프로토콜이 또한 일부 경우에 사용될 수 있다. 신뢰할 수 있는 데이터 전송 프로세스는 일반적으로 유니캐스트 데이터(즉, 소스 디바이스(300)로부터 단일 목적지 디바이스(702)로 전송된 데이터)에 사용된다.

[0097] 소스 디바이스 전송

[0098] 메시 서비스(310)는 각각의 할당된 애플리케이션 버퍼(320, 322 및 324)에 대해 메시 서비스 전송 데이터 버퍼

위치(258)에 전송 데이터 버퍼(도 9에서 326, 328 또는 330로 도시됨)를 할당한다. 이와 같이, 애플리케이션에서 특정 메시 포트와 관련된 목적지 디바이스로의 각 데이터 흐름을 위해 전송 버퍼가 할당된다. 소스 디바이스(300)와 목적지 디바이스 사이에 하나 이상의 데이터 흐름이 있을 수 있다. 예를 들어, 도 7에 도시된 메시 네트워크(100)가 제 1 및 제 2 애플리케이션(302 및 304)의 인스턴스가 모두 실행되는 다른 디바이스를 포함한다면, 디바이스(300)의 메시 서비스(310)는 애플리케이션(302)과 목적지 사이의 데이터 흐름을 위해 제 1 애플리케이션 버퍼 및 애플리케이션(304)과 목적지 사이의 데이터 흐름을 위한 제 2 애플리케이션 버퍼를 할당할 것이다. 이 제 1 및 제 2 애플리케이션 버퍼는 소스 uuid와 목적지 uuid가 동일하지만 메시 포트가 다를 수 있다(예를 들어, 6000 및 5000).

[0099] 도 11a를 참조하면, 신뢰할 수 있는 데이터 전송 프로세스는 각각의 데이터 흐름(즉, 디바이스(300)와 같은 디바이스 상에 각각의 애플리케이션 버퍼(256))에 대해 실행되고 1102에서 시작하는 프로세스 스레드(1100)를 포함한다. 블록(1104)은 소스 디바이스(300)의 마이크로 프로세서(202)로 하여금 데이터 흐름에 대해 할당된 대응하는 전송 버퍼(즉, 전송 버퍼(326))가 1/3미만인지를 결정하도록 지시한다. 블록(1104)에서, 전송 버퍼(326)가 1/3이상이면, 마이크로 프로세서(202)는 스레드(1100)의 시작으로 되돌아 가게 지시된다. 블록(1104)에서, 전송 버퍼(326)가 1/3미만이면, 마이크로 프로세서(202)는 블록(1106)으로 지시되고, 이는 마이크로 프로세서로 하여금 애플리케이션 버퍼(320)로부터 데이터를 판독하고 패킷화하여 상기 데이터를 전송 버퍼(326)에 기록하게 지시한다. 일 실시예에서, 메시 서비스(310)는 메시 네트워크(100)를 통한 전송을 용이하게 하도록 구성된 데이터 패킷에 상기 메시 네트워크(100)를 통해 전송하기 위한 데이터를 인코딩한다. 메시 서비스(310)가 네트워크를 통해 데이터 패킷을 보내도록 시도할 때 메시 네트워크(100) 상의 목적지 디바이스의 네트워크 어드레스가 결정될 것이기 때문에 이번에 목적지는 목적지 디바이스의 uuid를 통해서만 식별될 수 있지만, 데이터 패킷들은 일반적으로 UDP(User Datagram Protocol)를 준수할 수 있다. 일반적으로, 전송 버퍼(326)에 기록된 데이터 패킷의 크기는 디바이스의 무선 라디오(216)에 의해 구현되는 다양한 무선 링크에 의해 전송될 수 있는 최대 전송 유닛(MTU)의 제한 내에서 가능한 한 클 것이다. 일 실시예에서, 전송 버퍼(326)는 약 300개의 데이터 패킷을 보유하도록 크기가 정해질 수 있고, 따라서 블록(1106)은 전송 버퍼에 100개 미만의 데이터 패킷이 있다고 결정될 때마다 실행된다. 스레드(1100)는 메시 네트워크(100)를 통한 데이터 전송 속도에 상응하는 시간 간격으로 반복될 수 있다.

[0100] 이 실시예에서, TCP(Transmission Control Protocol)는 메시 네트워크(100)가 수용될 다양한 상이한 어드레스에 의존하는 반면 TCP는 기본적으로 인터넷 프로토콜(IP) 어드레싱을 지원하기 때문에 메시 네트워크(100)를 통한 데이터 전송에는 사용되지 않는다. TCP는 또한 연결 기반 프로토콜이며, 라우팅 모드로 구성된 메시 네트워크(100)를 구성하는 디바이스는 주기적으로 마스터 모드에서 다른 디바이스에 연결되는 것 사이를 전환하므로, 각 스위치는 TCP 연결을 중단시키며 재확립하기 위해 추가 오버헤드를 필요로 할 수 있다. 또한, TCP는 Android 운영 시스템에서 지원되지 않는 수정된 버전의 TCP를 사용하지 않는 한 다중 경로 라우팅을 지원하지 않는다.

[0101] 여전히 도 11a를 참조하면, 신뢰할 수 있는 데이터 전송 프로세스는 또한 각 데이터 흐름에 대해 실행되고 1112에서 시작하는 프로세스 스레드(1110)를 포함한다. 각 데이터 흐름에 대해, 메시 서비스(310)는 각각의 애플리케이션 버퍼(326, 328 및 330)에 대응하는 전송 큐(332, 334 또는 336)를 할당한다. 전송 큐(332, 334, 336)는 메모리(210)의 메시 서비스 전송 큐 위치(260)에 저장된다. 이 실시예에서, 프로세스(1110)는 메시 네트워크(100)상의 전송 혼잡을 피하기 위한 기능을 제공하기 위한 가변 혼잡 윈도우와 결부하여 전송 큐를 이용한다. 메시 네트워크(100)상의 각각의 디바이스(300, 700, 702)가 가능한 한 빨리 데이터를 전송한 경우, 심한 혼잡으로 인해 메시 네트워크(100)를 작동 불가능하게 할 수 있다. 혼잡 윈도우의 크기는 전송 디바이스가 전송된 패킷이 목적지 디바이스에 의해 수신되었다는 것을 확인할 수 있기 전에 메시 네트워크(100)를 통해 전송될 최대 데이터 패킷 수를 결정한다. 일 실시예에서, 혼잡 윈도우 크기는 초기에 1로(즉, 단일 데이터 패킷이 전송되게) 설정되고, 이후에 후술되는 바와 같이 목적지에 의해 전송된 패킷의 수신의 성공적인 확인에 기초하여 증가된다.

[0102] 프로세스(1110)는 각각의 특정 데이터 흐름에 대해 실행되고 블록(1112)에서 시작한다. 블록(1114)은 소스 디바이스(300)의 마이크로 프로세서(202)가 데이터 흐름에 대한 전송 큐(예를 들어, 전송 큐(332)) 내의 다수의 데이터 패킷이 현재 혼잡 윈도우(CW) 크기보다 작은지 여부를 결정하게 지시한다. 예를 들어, 전송이 전송 큐(332)와 연관된 데이터 흐름에 대해 방금 시작된 경우, 전송 큐는 비워지고 블록(1114)은 마이크로 프로세서(202)를 블록(1116)으로 지시한다. 블록(1116)은 마이크로 프로세서(202)가 전송 버퍼(326)로부터 혼잡 윈도우 크기에 해당하는 다수의 데이터 패킷를 읽고 데이터 패킷을 전송 큐(332)에 추가하도록 지시한다. 블록(1116)은 또한 마이크로 프로세서(202)가 전송 버퍼(326)로부터 데이터 패킷을 제거하도록 지시한다. 상술한 바와 같이,

초기에 혼잡 윈도우 큐 크기가 1로 설정될 수 있고, 따라서 단일 데이터 패킷이 전송 큐(332)에 추가될 수 있다. 블록(1114)에서, 데이터 흐름에 대한 전송 큐(332)의 데이터 패킷의 수가 현재 혼잡 윈도우(CW)보다 적지 않으면, 마이크로 프로세서(202)는 블록(1118)으로 지시된다.

[0103] 프로세스(1110)는 블록(1118)에서 계속되고 상기 블록은 마이크로 프로세서(202)로 하여금 목적지 디바이스로의 라우팅 정보가 있는지 결정하도록 지시한다. 메시 서비스(310)는 상기에서 참조된 바와 같이 미국 가출원 제 62/343,056 호에 전반적으로 설명되고 그 전체가 본 명세서에 참조로 포함된 바와 같이 메모리(210)의 라우팅 테이블 위치(264)에 라우팅 테이블을 유지한다. 라우팅 테이블 리스트는 그들의 메시 네트워크 어드레스에 의해 메시 네트워크(100)에서 이전에 발견된 목적지 디바이스들을 나열한다.

[0104] 각각의 디바이스는 메시 네트워크(100)에서 클라이언트, 라우팅 디바이스 및 액세스 포인트일 수 있는 하나 이상의 역할을 갖는다. 클라이언트는 "hello" 메시지를 액세스 포인트 디바이스로 전송하여 액세스 포인트와의 연결을 요청한다. 액세스 포인트 디바이스는 연결 요청을 승인하는 확인응답("Hello ACK")을 전송할 수 있다. 클라이언트는 현재 액세스 포인트 디바이스에서 지원하는 무선 프로토콜에 따라 WiFi, Wi-Fi 다이렉트 또는 Bluetooth를 통해 연결할 수 있다. 라우팅을 위한 메시 네트워크의 토폴로지는 마스터 피어를 중심으로 클러스터로 구성되고 마스터 또는 라우터를 통해 이중 역할로 메시에 연결된다. 라우팅 설정을 위해 기본 패턴 패킷이 이전 패턴에 도입되었다.

[0105] 각 목적지 디바이스는 관련된 넥스트 홉 어드레스를 가지며, 목적지 디바이스가 전송 디바이스와 직접 연결되어 있다면 넥스트 홉 어드레스가 목적지 디바이스의 어드레스가 되고 따라서 홉 카운터는 1로 설정될 것이다. 목적지 디바이스가 직접 통신이 아니라 메시 네트워크(100)상의 하나 이상의 다른 라우팅 디바이스를 통해 연결된 경우, 넥스트 홉 어드레스는 라우팅 디바이스의 어드레스일 것이다. 목적지가 하나의 라우팅 디바이스로만 분리된 경우, 홉 카운터는 2로 설정된다. 넥스트 홉 카운터가 3이상이면 전송 디바이스와 목적지 디바이스 사이에 2 이상의 라우팅 디바이스가 있음을 나타낸다. 따라서, 전송 디바이스는 넥스트 홉 디바이스에만 관련되어 있으며, 넥스트 홉 디바이스 뒤에 남아있는 메시 네트워크(100)를 메시 네트워크(100)상의 디바이스의 어드레스 및 이용 가능한 적용 가능 홉 카운트만을 갖는 블랙 박스로서 간주한다. 일 실시예에서, 목적지 디바이스로 이어지는 하나 이상의 넥스트 홉 디바이스가 있고 하나 이상의 패킷이 전송되는 경우, 패킷은 상이한 경로를 통해 동시에 전송되어 네트워크 레이턴시를 감소시킬 수 있다.

[0106] 프로세스(1110)의 블록(1118)은 마이크로 프로세서(202)가 목적지 디바이스가 라우팅 테이블(264)에 나열되어 있는지 여부를 결정하도록 지시하고, 추가로 마이크로 프로세서(202)가 넥스트 홉이 현재 연결되어 있는지를 결정하도록 지시한다. 상술한 바와 같이, 라우팅 모드의 디바이스는 상이한 액세스 포인트에 접속하는 것 사이에서 교대하고, 따라서 라우팅 테이블(264)에 가능한 넥스트 홉 디바이스로서 나타나는 것이라도, 현재 데이터를 라우팅하는데 이용 가능하지 않을 수 있다. 블록(1118)에서, 목적지 디바이스가 라우팅 테이블(264)에 나열되지 않거나 라우팅 테이블의 넥스트 홉이 현재 연결 해제된 경우, 마이크로 프로세서(202)는 블록(1112)으로 다시 지시된다. 블록(1118)에서 목적지 디바이스는 라우팅 테이블(264)에 나열되고 현재 연결되어 있다면, 마이크로 프로세서(202)는 블록(1120)으로 지시된다.

[0107] 블록(1120)은 마이크로 프로세서(202)가 무선 라디오(216)와 관련된 무선 링크(즉, Wi-Fi, Wi-Fi 다이렉트 또는 Bluetooth)가 전송에 이용 가능한지 여부를 결정하도록 지시하고, 이 경우프로세스는 블록(1122)에서 계속된다. 블록은 마이크로 프로세서(202)로 하여금 전송 큐(332)가 비 었는지 여부를 결정하도록 지시하고, 이 경우 마이크로 프로세서(202)는 블록(1112)으로 다시 지시되고, 블록(11114-1120)은 전송 큐에 데이터가 전송될 때까지 반복된다. 블록(1122)에서 전송 큐(332)가 비어 있지 않으면, 마이크로 프로세서(202)는 블록(1124)으로 지시된다.

[0108] 상술한 바와 같이, 각 디바이스의 무선 라디오(216)는 Wi-Fi, Wi-Fi 다이렉트 또는 Bluetooth 등과 같은 여러 다른 무선 링크 또는 프로토콜을 통한 연결을 제공할 수 있다. 메시 서비스(310)는 각 무선 링크에 대한 링크 큐를 할당한다. 링크 큐는 메모리(210)의 무선 링크 큐 위치(262) 내에 유지된다. 예를 들어, 메시 서비스(310)는 Wi-Fi 큐(338), Wi-Fi 다이렉트 큐(340) 및 Bluetooth 큐(342)를 할당 및 유지할 수 있다. 상술한 바와 같이, 상기 일부 디바이스에서, 하나 이상의 무선 링크가 일시적으로 또는 영구적으로 이용 불가능할 수 있으며, 그러한 큐는 이용 가능한 무선 링크에 대해서만 할당될 것이다.

[0109] 블록(1124)은 마이크로 프로세서(202)가 전송을 위한 데이터 패킷을 생성하도록 지시한다. 데이터 패킷의 예는 도 12에 1300으로 도시되어 있다. 도 12를 참조하면, 데이터 패킷(1300)은 MAX_SIZE의 전체 크기를 가지며 순차 블록으로 도시된 복수의 데이터 필드를 포함한다. 데이터 패킷(1300)이 UDP 데이터 패킷으로서 전송될 때, 패킷

은 UDP 헤더(1302)로 시작한다. UDP 헤더는 Wi-Fi 및 Wi-Fi 다이렉트 전송에 사용되지만 다른 프로토콜을 따르는 블루투스 전송에는 사용되지 않는다.

[0110] 데이터 패킷(1300)은 전송이 신뢰할 수 있는 전송인지 신뢰할 수 없는 전송인지를 나타내는 1 바이트 request_type 필드(1304)로 시작한다. 데이터 패킷(1300)은 또한 source_uuid_type 필드(1306) 및 source_uuid 필드(1308)를 포함한다. source_uuid 필드(1308)는 요청을 하는 디바이스에 대한 (source_uuid_type의) 어드레스를 보유하는데 사용된다. 유사하게, 데이터 패킷(1300)은 또한 destination_uuid_type 필드(1310) 및 destination_uuid 필드(1312)를 포함하며, 여기서 destination_uuid 필드는 요청이 전송되는 디바이스에 대한 destination_uuid_type의 uuid를 보유하는데 사용된다. 일 실시예에서 request_type 필드(1304), source_uuid _type 필드(1306) 및 destination_uuid _type 필드(1310)는 안드로이드 운영 시스템에서 제공되는 Varint 데이터 타입을 사용하여 1 바이트 열거형(enumerations)으로서 저장될 수 있다. 이 실시예에서 source_uuid 필드(1308) 및 destination_uuid 필드(1312)는 데이터 패킷(1300)의 20 바이트를 차지한다. 데이터 패킷(1300)은 또한, 예를 들어, UDP 프로토콜에 대한 최신 개정판과의 호환성을 제공하기 위해 사용될 수 있는 프로토콜 버전을 나타내는 1 바이트 값을 보유하는 protocol_version 필드(1314)를 포함한다.

[0111] 블루투스 프로토콜을 통해 전송할 때, UDP 헤더(1302)는 사용되지 않는다. 오히려 앞으로 올 바이트 수가 있는 정수가 전송된다. 수신 Bluetooth 디바이스는 지정된 바이트 수가 수신될 때까지 데이터 바이트를 읽는다. 데이터 패킷이 Bluetooth를 통해 전달되면 적절한 Bluetooth 무선 링크가 선택되고 라우팅 테이블로부터 목적지 MAC 어드레스를 기반으로 데이터가 올바른 큐에 대기된다. Wi-Fi 및 Wi-Fi 다이렉트 링크를 통한 인터넷 프로토콜 전송의 경우, 데이터 패킷이 싱글-홉 큐에 대기될 때 목적지 어드레스가 추가된다.

[0112] 데이터 패킷(1300)은 또한 싱글-홉을 통한 메시 네트워크(100)를 통해 무선 링크에 의해 전송된 일련의 데이터 패킷을 추적하기 위한 single-hop_seq# 필드(1316)를 포함한다. 블록(1124)에서 무선 링크의 레벨에서 UDP 데이터 패킷(1300)을 생성할 때, 순차적인 single-hop_seq# 값이 필드(1316)에 기록되어 아래에 설명된 바와 같이 어떤 데이터 패킷이 성공적으로 전송되었는지를 식별할 수 있게 한다. 데이터 패킷(1300)은 또한 상술한 바와 같이 요청을 생성하는 애플리케이션에 대한 메시 포트 식별자를 유지하기 위한 mesh_port 필드(1318)를 포함한다. 이 필드는 각각 Varint 데이터 형식을 사용하여 인코딩되며, 이는 1 내지 4 바이트 사이를 사용하여 정수를 직렬화하고 더 적은 숫자들은 통신 효율을 위해 적은 수의 바이트를 사용한다.

[0113] 데이터 패킷(1300)은 또한 데이터 페이로드의 바이트 길이를 특정하는 값을 보유하는 data_length 필드(1320)를 포함하여, 데이터 패킷에서 운반될 페이로드와 관련된 다수의 필드를 포함한다. 이 실시예에서, 일련의 데이터 패킷 내의 제 1 데이터 패킷만이 data_length 필드(1320)를 포함할 것이다. 메시 네트워크(100)를 통한 데이터 통신의 무결성을 검증하기 위해 선택적인 checksum 필드(1322)가 포함될 수 있다. 데이터 패킷(1300)은 또한 전송된 데이터 패킷의 시퀀스에서 각 데이터 패킷의 순서를 식별하는 multi-hop_seq 필드(1324)를 포함한다. 일 실시예에서, multi-hop seq는 가능한 최대 시퀀스 번호가 충분히 큰 한 명백히 순환 방식으로 숫자 시퀀스를 재사용할 수 있다.

[0114] data_payload 필드(1328)는 데이터 패킷의 나머지 필드에서의 헤더 바이트 수보다 적은 DATA_MAX의 크기를 갖는다. 이 실시예에서, 데이터 패킷(1300)은 또한 데이터 패킷(1300) 내의 데이터 페이로드가 암호화되었는지의 표시를 보유하기 위한 isEncrypted 필드(1326)를 포함한다.

[0115] 데이터 패킷(1300)은 또한 데이터 패킷과 관련된 전송 시간을 유지하기 위한 timestamp 필드(1330)를 포함한다. timestamp 필드(1330)가 도 12에 도시된 데이터 패킷의 일부로서 도시되어 있지만, 타임스탬프(timestamp)는 다른 디바이스로 전송되는 것이 아니라 디바이스상에서의 전송을 위해 대기중인 데이터 패킷에만 유지된다.

[0116] 블록(1124)은 또한 마이크로 프로세서(202)가 현재 시간을 데이터 패킷(1300)의 timestamp 필드(1330)에 기록하도록 지시한다(그러나 상술한 바와 같이, 이 타임 스탬프는 메시 네트워크(100)를 통해 전송되지 않고, 후술하는 바와 같이 추적을 위해 소스 디바이스에 의해 사용된다). 그런 후, 프로세스 스레드(1110)는 블록(1126)에서 계속되며, 상기 블록은 마이크로 프로세서(202)가 링크 큐(332)의 컨텐츠(즉, 데이터 패킷 또는 데이터 패킷들)을 적용 가능한 무선 링크 큐(예를 들어, Wi-Fi 큐(338))에 기록하도록 지시한다.

[0117] 그 후, 블록(1128)은 마이크로 프로세서(202)가 엔드-투-엔드 타임아웃 타이머가 데이터 흐름에 대해 시작되었는지를 결정하도록 지시한다. 데이터 흐름에서 이전에 전송된 데이터 패킷이 Wi-Fi 링크에 의한 전송을 위해 큐(338)에 이미 기록 되었다면 엔드-투-엔드 타임아웃 타이머는 시작되었을 것이다. 이 경우에, 블록(1128)은 마이크로 프로세서(202)를 다시 블록(1112)으로 지시한다. 엔드-투-엔드 타임아웃 타이머가 아직 시작되지 않았다

면, 큐(338)에 기록된 데이터 패킷은 데이터 흐름의 제 1 데이터 패킷이고 마이크로 프로세서(202)는 블록(1130)으로 지시되며, 여기서 엔드-투-엔드 타임아웃 타이머가 시작된다. 블록(1130)은 마이크로 프로세서(202)를 다시 블록(1112)으로 지시하고 스레드(1110)는 데이터 흐름에서 추가 데이터 패킷에 대해 반복된다.

[0118] 도 11b를 참조하면, 신뢰할 수 있는 데이터 전송 프로세스는 또한 1142에서 시작하는 싱글-홉 전송 프로세스 스레드(1140)를 포함한다. 싱글-홉 전송 프로세스 스레드(1140)는 무선 라디오(216)가 각각의 링크 큐(338, 340 및 342)에서 동작하고 데이터 흐름을 위해 데이터 패킷을 전송하도록 지시한다. 블록(1144)은 소스 디바이스(300)의 마이크로 프로세서(202)로 하여금 링크 큐(예를 들어, Wi-Fi 큐(338))가 비어 있는지를 결정하도록 지시하고, 이 경우 마이크로 프로세서는 다음 링크 큐 및 프로세스를 처리하기 위해 블록(1146)으로 지시된 다음 1142에서 재개한다. 블록(1144)에서 링크 큐가 비어 있지 않은 경우, 마이크로 프로세서(202)는 블록(1148)으로 지시되고, 여기서 전송 재시도 카운터 x_r은 값 1로 초기화된다. 전송 재시도 카운터는 메모리(210)의 카운터 위치(266)에 유지되고, 무선 링크에 의한 특정 데이터 패킷의 전송을 위한 전송 시도를 모니터링하는데 사용된다.

[0119] 그 후, 블록(1150)은 마이크로 프로세서(202)가 전송을 위해 링크 큐(338)의 헤드에서 데이터 패킷을 처리하도록 지시한다. 각각의 무선 링크는 특정 전송 프로토콜 및 전송 포맷을 가질 것이며, 무선 링크는 자신의 특정 전송 포맷 내에서 데이터 패킷(1300)의 필요한 캡슐화를 수행할 것이다. 예를 들어, 인터넷 프로토콜(IP)을 사용하여 데이터를 전송하는 Wi-Fi 및 Wi-Fi 다이렉트 링크는 도 12에 도시된 UDP 헤더(1302)와 함께 데이터 패킷(1300)을 캡슐화한다. 일부 실시예에서, 데이터 패킷(1300)은 무선 링크에 의해 전송될 수 있는 것보다 더 클 수 있으며, 이는 더 작은 데이터 패킷으로 전송하기 위해 데이터 패킷을 분할해야 할 수 있다. 데이터 패킷이 도 12에 1300으로 도시된 경우, single-hop_seq# 필드(1316)는 싱글-홉 확인응답에 사용되어 데이터 패킷이 넥스트 홉으로 이동하게 하여 패킷이 목적지에 도착하도록 보장한다. 무선 링크는 싱글-홉 큐에서 작동하여 가능한 모든 패킷을 전송하고 각 싱글-홉 패킷은 전송과 관련된 시간을 갖는다. 싱글-홉 ACK를 수신하기 전에 타임아웃이 있는 경우, 전송 디바이스는 무선 링크에 대해 설정된 MAX_RETRIES에 도달할 때까지 데이터를 다시 보내며, 이 시점에서 패킷이 삭제된다. 무선 링크는 single-hop_seq #(1316)을 설정하고 유지한다.

[0120] 블록(1150)은 마이크로 프로세서(202)로 하여금 무선 라디오(216)가 무선 링크(이 경우 Wi-Fi 무선 링크)를 통해 넥스트 홉 디바이스로 데이터 패킷을 전송하게 지시한다. 싱글-홉 전송 프로세스 스레드(1140)는 블록(1152)에서 계속되며, 이는 마이크로 프로세서(202)가 싱글-홉 확인응답(싱글-홉 ACK)이 일정 기간 내에 넥스트 홉 디바이스로부터 다시 수신되는지 여부를 모니터링하도록 지시한다. 기간은 미리 결정된 최대 시간으로서 구현되지만, 송신 디바이스는 재전송 전에 최대 시간까지 랜덤 시간 동안 대기한다. 랜덤 시간은 라우터 모드에서 한 액세스 포인트 디바이스와 주기적으로 연결을 끊어 다른 액세스 포인트 디바이스에 연결하는 디바이스와의 연결 실패 가능성을 줄이는 데 사용된다. 각 재전송 후에, MAX_RETRIES 임계값에 도달하면 패킷화된 데이터를 드롭하기 전에 더 긴 대기 시간이 만료될 수 있다.

[0121] 싱글-홉 ACK는 수신된 데이터 패킷의 필드(1316)로부터 판독된 single-hop_seq#을 포함하고, 식별된 데이터 패킷이 의도된 목적지에 도달했다는 확인을 전송 디바이스에 제공한다. 블록(1152)은 또한 마이크로 프로세서(202)로 하여금 전송 재시도 카운터 x_r가 최대 재시도 임계치 r_{max}에 도달했는지를 결정하도록 지시한다. 일 실시예에서, r_{max}의 값은 각각의 데이터 패킷 전송에 대한 3개의 재시도 시도에 대응하여 3으로 설정된다. 블록(1152)에서, 싱글-홉 ACK가 수신되거나 전송 재시도 카운터 x_r이 최대 재시도 임계치 r_{max}에 도달 한 경우, 싱글-홉 전송 프로세스 스레드(1140)는 블록(1154)에서 계속되고 상기 블록은 마이크로 프로세서(202)가 링크 큐(338)로부터 데이터 패킷을 제거하도록 지시한다. 이와 같이, 무선 링크에 의한 전송이 성공하지 못하면, 데이터 패킷은 큐에서 제거되고 더 이상 전송을 시도하지 않는다. 따라서, 신뢰할 수 있는 전송 프로세스는 본 명세서에서 후술하는 엔드-투-엔드 확인 프로세스에 의지한다. 싱글-홉 확인 프로세스의 한 가지 장점은 프로세스가 중단된 싱글-홉 링크를 통해 끝없이 재전송을 시도함이 없이 제한된 재시도 메커니즘을 통해 허위 전송 실패가 방지된다는 것이다. 메시 네트워크(100)를 거쳐 다수의 홉을 통해 데이터 패킷을 전송할 때, 홉 중 하나에서 전송 실패가 있을 가능성이 높다. 싱글-홉 전송 프로세스 스레드(1140)는 따라서 재전송을 시도함으로써 메시 네트워크(100)가 장애로부터 복구할 수 있게 하고, 이에 따라 이하에서 더 상세히 설명되는 엔드-투-엔드 재전송의 회수를 줄인다. 블록(1154)은 마이크로 프로세서(202)를 다시 블록(1144)으로 지시하고 블록(1144-1156)은 상술한 바와 같이 반복된다.

[0122] nexthop uuid는 목적지 디바이스에 대한 라우팅이 발견될 때 결정된다. 무선 링크가 Wi-Fi 링크인지 Bluetooth 링크인지에 따라 넥스트 홉에 MAC 어드레스 또는 IP 어드레스가 필요할 수 있다. MAC 어드레스인 경우, Bluetooth 링크 구현은 모든 1 : 1 Bluetooth 링크의 MAC 어드레스를 판별하고 데이터 패킷이 올바른

Bluetooth 링크 큐에 대기한다. Wi-Fi 또는 Wi-Fi 다이렉트 무선 링크를 통한 IP 전송의 경우, 라우팅 테이블 위치(264)에서 넥스트 홉 디바이스의 조회를 기반으로 넥스트 홉 IP 어드레스가 UDP 데이터 패킷에 추가된다. 이 IP 어드레스는 넥스트 홉 디바이스로의 전송을 위해 UDP 헤더(1302)에 채워진다. 따라서, 데이터 패킷의 destination_uuid 필드(1312)는 데이터 패킷(1300)의 최종 목적지 디바이스를 식별하는 반면, 메시 네트워크(100)의 넥스트 홉 IP 어드레스는 최종적으로 목적지 디바이스에 도달하기 위해 데이터 패킷이 전송될 다음 디바이스를 식별한다.

[0123] 블록(1152)에서, 싱글-홉 ACK가 아직 수신되지 않고 전송 재시도 카운터 x_r가 아직 최대 재시도 임계치 r_{max}에 도달하지 않은 경우, 마이크로 프로세서(202)는 전송 재시도 카운터 x_r가 증가되는 블록(1156)으로 지시된다. 그런 후, 블록(1156)은 마이크로 프로세서(202)를 다시 블록(1150)으로 지시하고, 무선 링크를 통해 데이터 패킷을 전송하려는 추가 시도가 이루어진다.

[0124] 싱글-홉 프로세스 스레드(1140)는 데이터가 의도된 목적지로 연속적인 싱글-홉 전송을 통해 메시 네트워크(100)를 통해 전파되도록 각각의 무선 링크 큐 및 메시 네트워크(100)의 각 디바이스에 대해 구현된다.

[0125] 목적지/라우팅

[0126] 도 11c를 참조하면, 신뢰할 수 있는 데이터 전송 프로세스는 또한 메시 네트워크(100)상의 디바이스를 수신하는 메시 서비스에 의해 구현되는 엔드-투-엔드 확인응답 프로세스 스레드(1160)를 포함한다. 엔드-투-엔드 확인응답 프로세스 스레드(1160)는 메시 네트워크(100) 상의 임의의 디바이스에서 데이터 패킷이 수신되면 1162에서 시작한다. 블록(1164)은 수신 디바이스(즉, 라우팅 디바이스 또는 목적지 디바이스(702))의 마이크로 프로세서(202)로 하여금 싱글-홉 ACK를 데이터 패킷이 수신되었음을 확인하는 데이터 패킷을 전송한 디바이스로 다시 전송하도록 지시한다. 싱글-홉 전송 프로세스 스레드(1140)의 블록(1152)과 관련하여 상술한 바와 같이, 싱글-홉 ACK는 각 디바이스에서 무선 링크 큐(262)를 관리하기 위해 소스 디바이스(300)(또는 전송을 위해 메시 네트워크(100)를 가로 질러 멀티 홉이 있는 경우 다른 라우팅 디바이스)에 의해 사용된다.

[0127] 그런 후, 블록(1164)은 마이크로 프로세서(202)로 하여금 데이터 패킷의 destination_uuid 필드(1312)를 읽도록 지시하고, 블록(1166)은 마이크로 프로세서로 하여금 destination_uuid 필드의 컨텐츠를 메시 네트워크(100) 상의 디바이스의 자신의 어드레스와 비교함으로써 수신 디바이스가 데이터 패킷의 최종 목적지인지를 결정하도록 지시한다. 디바이스가 데이터 패킷의 최종 목적지가 아니라 라우팅 디바이스로서 작용하는 경우, 프로세스 스레드는 블록(1168)에서 계속되며, 여기서 마이크로 프로세서(202)는 데이터 패킷을 메모리(210)의 포워딩 버퍼(268)에 기록하도록 지시된다. 포워딩 버퍼(268)를 관리하도록 라우팅 디바이스를 지시하기 위한 스레드는 도 11d를 참조하여 본 명세서에서 나중에 설명된다. 그 후 프로세스(1160)는 블록(1168)에서 계속되며, 상기 블록은 무선 링크를 통한 다음 데이터 패킷의 수신을 대기하도록 마이크로 프로세서(202)를 1162로 다시 지시한다. 따라서, 디바이스가 데이터 흐름에 대한 라우팅 디바이스로서 작용하는 경우, 데이터 패킷을 포워딩할 때 프로세스(11160)의 블록(1162-1168)만이 실행될 것이다.

[0128] 블록(1166)에서 수신 디바이스가 데이터 패킷의 최종 목적지인 경우(즉, destination_uuid 필드(1312)가 메시 네트워크(100)상의 디바이스의 자신의 어드레스와 일치하는 경우), 마이크로 프로세서(202)는 블록(1170)으로 지시된다. 블록(1170)은 마이크로 프로세서(202)로 하여금 수신된 데이터 패킷이 multi-hop_seq 필드(1324)를 판독함으로써 진행중인 데이터 흐름과 관련된 순서 데이터 패킷인지의 여부를 결정하도록 지시한다. 디바이스에서 수신된 각각의 데이터 패킷은 source_uuid 필드(1308), destination_uuid 필드(1312), 및 mesh_port 필드(1318)에 기초하여 특정 데이터 흐름과 연관되는 것으로 결정될 수 있다. 진행중인 데이터 흐름과 관련된 하나 이상의 데이터 패킷이 디바이스에 의해 이미 수신된 경우, 다음 데이터 패킷은 데이터 흐름에 대해 마지막에 수신된 패킷 위에 1만큼 증가된 멀티 홉 seq 필드(1324)를 가질 것으로 예상될 것이다. 블록(1170)에서, 마이크로 프로세서(202)는 수신된 데이터 패킷이 진행중인 데이터 흐름에 대한 다음의 예상되는 멀티 홉 seq와 일치하는지 또는 데이터 흐름의 제 1 패킷(multi-hop_seq = 1)인지를 결정하도록 지시된다. 어느 경우이든, 프로세스는 블록(1172)에서 계속되며, 상기 블록은 마이크로 프로세서(202)가 임계값 카운터 x_0를 증가시키도록 지시한다. 임계값 카운터 x_0는 메모리(210)의 카운터 위치(266)에 저장되고 순차적으로 수신된 순차적 데이터 패킷의 카운트를 유지하는데 사용된다. 블록(1174)은 임계값 카운터 x_0가 임계값 x_{0max}에 도달했는지 여부를 결정하도록 마이크로 프로세서(202)에 지시한다.

[0129] 그 후, 프로세스 스레드(1160)는 마이크로 프로세서(202)가 임계값 카운터 x_0을 0으로 리셋하도록 지시되는 블

록(1176)에서 계속된다. 블록(1176)은 또한 마이크로 프로세서(202)로 하여금 데이터 흐름에서 수신된 마지막 순서 데이터 패킷의 시퀀스 번호를 결정하도록 지시한다. 그 후, 블록(1178)은 마이크로 프로세서(202)로 하여금 엔드-투-엔드 ACK를 생성하여 데이터 패킷의 source_uuid 필드(1308)에 의해 데이터 흐름의 소스로서 식별된 디바이스로 다시 전송하도록 지시한다. 엔드-투-엔드 ACK는 최종 수신된 순서 데이터 패킷의 시퀀스 번호를 증가시킴으로써 결정된 다음 예상 데이터 패킷의 시퀀스 번호를 포함한다. 이 실시예에서, 성공적으로 수신된 최종 패킷을 식별하는 대신에 엔드-투-엔드 ACK는 시퀀스 번호에 의해 다음의 예상되는 데이터 패킷을 식별한다. 다른 실시예들에서, 엔드-투-엔드 ACK는 성공적으로 수신된 마지막 패킷을 식별할 수 있다. 블록(1178)은 마이크로 프로세서(202)를 다시 블록(1162)으로 보내서 추가 데이터 패킷의 수신을 기다린다.

[0130] 임계값 x_{0max}는 일반적으로 데이터 흐름과 관련된 하나 이상의 데이터 패킷이 수신된 후에 엔드-투-엔드 ACK가 메시 네트워크(100)를 통해서만 전송되도록 적어도 2 또는 3의 값으로 설정될 것이다. 엔드-투-엔드 ACK 프로세스 스레드(1160)는 따라서 각각의 단일 패킷보다는 여러 데이터 패킷의 수신을 확인하기 위해 ACK 메시지를 생성함으로써 엔드-투-엔드 ACK 메시지로 메시 네트워크(100)를 플러딩(flooding)하는 것을 방지한다.

[0131] 블록(1174)에서, 임계값 카운터 x_0가 아직 임계값 x_{0max}에 도달하지 않은 경우, 프로세스는 블록(1182)에서 계속되어, 상기 블록은 마이크로 프로세서(202)로 하여금 데이터 흐름에 대해 엔드-투-엔드 ACK 타이머(ACK 타이머)가 시작되었는지를 결정하도록 지시한다. 데이터 패킷이 데이터 흐름에서 제 1 패킷인 경우, 엔드-투-엔드 ACK 타이머는 시작되지 않을 것이며, 블록(1184)에서 타이머는 데이터 흐름과 관련되어 0으로 초기화된다. 블록(1182)에서 엔드-투-엔드 ACK 타이머가 이전에 시작되었다면, 마이크로 프로세서(202)는 블록(1186)으로 보내지고 엔드-투-엔드 ACK 타이머는 0으로 리셋된다. 엔드-투-엔드 ACK 타이머는 엔드-투-엔드 ACK 타이머 만료 임계값과 결부하여 사용되어 디바이스가 데이터 흐름과 관련된 더 많은 데이터 패킷을 수신하기 위해 대기하는 동안의 기간을 설정한다.

[0132] 엔드-투-엔드 ACK 프로세스 스레드(1160)와 관련된 별도의 스레드(1190)에서, 블록(1192)은 엔드-투-엔드 ACK 타이머를 모니터링하고 상기 타이머가 만료 임계값에 도달하면, 수신 디바이스의 마이크로 프로세서(202)는 블록(1194)으로 지시된다. 블록(1194)은 마이크로 프로세서(202)로 하여금 수신된 마지막 순서의 데이터 패킷의 multi-hop_seq 필드(1324)로부터의 값을 결정하도록 지시한다. 블록(1194)은 또한 마이크로 프로세서(202)를 블록(1178)으로 지시하고, 여기서 다음으로 예상되는 데이터 패킷을 식별하는 엔드-투-엔드 ACK가 메시 네트워크(100)를 통해 소스 디바이스로 다시 전송된다. 엔드-투-엔드 ACK 프로세스 스레드(1160)는 또한 x_{0max}의 값에 의해 설정된 데이터 패킷의 수가 수신되어야하는 시간 내에 타임아웃을 구현한다. 타임아웃에 도달하면, 목적지 디바이스는 더 이상 추가 데이터 패킷을 기다리지 않고 다음 예상 패킷을 식별하는 소스 디바이스로 엔드-투-엔드 ACK를 다시 전송한다.

[0133] 블록(1170)에서 순차적 데이터 패킷이 수신되면, 마이크로 프로세서(202)는 블록(1180)으로 지시되고, 여기서 마이크로 프로세서는 마지막 수신된 순서의 데이터 패킷에서 멀티-홉 seq 필드(1324)의 값을 결정하도록 지시된다. 블록(1180)은 마이크로 프로세서(202)를 블록(1178)으로 지시하고, 여기서 마이크로 프로세서는 데이터 흐름에서 다음으로 예상되는 패킷(즉, multi-hop seq +1)에 대한 엔드-투-엔드 ACK를 전송하도록 지시된다. 이는 데이터 패킷이 재전송될 수 있도록 데이터 흐름에서 하나 이상의 누락된 데이터 패킷의 제 1 패킷의 multi-hop_seq를 소스 디바이스에 통지하는 효과가 있다.

[0134] 상술한 바와 같이, 블록(1168)에서, 데이터 패킷이 수신 디바이스 이외의 목적지를 갖는 것으로 결정되면, 디바이스는 라우팅 디바이스(예를 들어, 도 3의 디바이스(700))로서 작용하여 데이터 패킷을 포워딩 버퍼에 기록한다. 도 11d를 참조하면, 포워딩 버퍼(268)를 처리하기 위해 라우팅 디바이스(700)상에서 실행되는 포워딩 프로세스 스레드가 1200으로 도시되고 1202에서 시작한다. 포워딩 프로세스 스레드(1200)는 디바이스상의 각각의 포워딩 버퍼(268)에서 실행된다. 블록(1204)은 라우팅 디바이스의 마이크로 프로세서(202)로 하여금 포워딩 버퍼(268)에 임의의 데이터 패킷이 있는지 여부를 결정하도록 지시한다. 포워딩 버퍼(268)에 패킷이 없으면, 마이크로 프로세서(202)는 상기 마이크로 프로세서가 위치(268)의 다음 전달 버퍼를 처리하도록 지시하는 블록(1206)으로 지시된다.

[0135] 블록(1204)에서, 포워딩 버퍼에 하나 이상의 데이터 패킷이 있으면, 마이크로 프로세서(202)는 블록(1208)으로 지시되고, 상기 블록은 마이크로 프로세서가 데이터 패킷에서 destination_uuid 필드(1312)를 읽도록 지시한다. 포워딩 프로세스 스레드(1200)는 블록(1210)에서 계속되며, 상기 블록은 목적지 디바이스가 디바이스 라우팅 테이블(264)에 열거되는지 여부를 결정함으로써 목적지 정보에 대한 라우팅 정보가 존재하는지 여부를 마이크로

프로세서(202)가 결정하도록 지시한다. 블록(1210)은 추가로 마이크로 프로세서(202)로 하여금 넥스트 홉이 현재 연결되어 있는지 여부를 결정하도록 지시한다. 블록(1210)에서, 목적지 디바이스가 라우팅 테이블(264)에 열거되고 현재 연결되어 있다면, 마이크로 프로세서(202)는 블록(1212)으로 지시된다.

[0136] 블록(1212)은 무선 라디오(216)와 연관된 무선 링크 인터페이스가 이용 가능한지 여부를 결정하도록 마이크로 프로세서(202)에 지시하고, 이 경우 프로세스는 블록(1214)에서 계속된다. 블록(1214)은 마이크로 프로세서(202)로 하여금 전송을 위해 선택된 무선 링크에 대한 링크 큐에 데이터 패킷을 기록하도록 지시한다. 블록(1214)은 또한 마이크로 프로세서(202)가 포워딩 버퍼로부터 데이터 패킷을 제거하도록 지시한다. 전달된 데이터 패킷의 전송은 도 11b에 도시된 싱글-홉 전송 처리 스레드(1140)에 따르며, 무선 링크는 데이터 패킷을 포워딩기 위해 여러 번(r_{max}) 시도할 것이다. 포워딩 전송이 실패하면, 패킷은 링크 큐로부터 제거되고, 신뢰할 수 있는 전송을 위한 추가 처리는 본 명세서에서 후술하는 바와 같이 소스 디바이스로 되돌아 간다.

[0137] 도 11c를 다시 참조하면, 블록(1178)에서, 신뢰할 수 있는 데이터 전송 프로세스의 엔드-투-엔드 확인 프로세스 스레드(1160)는 데이터 흐름에서 다음으로 예상되는 패킷을 식별하는 엔드-투-엔드 ACK를 전송한다. 이 실시예에서, 엔드-투-엔드 ACK는 UDP 데이터 패킷(1300)의 포맷을 따르지만 빈 데이터 페이로드 필드를 가지며, 다중 홉 seq 필드(1324)에서의 흐름에 대한 다음 예상 데이터 패킷의 시퀀스 번호를 포함한다. 엔드-투-엔드 ACK는 상술한 바와 같이 하나 이상의 싱글-홉 전송을 통해 메시 네트워크(100)를 통해 전송되지만, 목적지 디바이스는 상술한 바와 같이 ACK 데이터 패킷에 대한 엔드-투-엔드 ACK 프로세스를 구현하지 않는다.

[0138] 소스 확인 처리

[0139] 도 11e를 참조하면, 신뢰할 수 있는 데이터 전송 프로세스는 또한 각각의 데이터 흐름에 대해 실행되고 엔드-투-엔드 ACK가 수신될 때 1222에서 시작하는 소스 확인응답 처리 스레드(1220)를 포함한다. 블록(1224)은 소스 디바이스(300)의 마이크로 프로세서(202)가 엔드-투-엔드 ACK에서 destination_uuid 필드(1312)를 판독하고 ACK가 소스 디바이스로 어드레싱되는지를 결정하도록 지시한다. 엔드-투-엔드 ACK가 메시 네트워크(100)의 다른 디바이스로 어드레싱되면, 블록(1226)은 마이크로 프로세서(202)가 엔드-투-엔드 ACK를 포워딩 버퍼(268)에 기록하도록 지시한다. 포워딩 프로세스 스레드(1200)는 메시 네트워크(100)를 따라 ACK를 포워딩할 것이다.

[0140] 블록(1224)에서, 엔드-투-엔드 ACK가 소스 디바이스로 어드레싱되면, 마이크로 프로세서(202)는 블록(1228)으로 지시된다. 블록(1228)은 마이크로 프로세서(202)가 ACK 데이터 패킷의 멀티-홉 seq 필드(1324)에서 ACK 시퀀스 번호가 이전에 수신되었는지를 결정하도록 지시한다. 소스 디바이스가 이전에 동일한 다음에 예상되는 데이터 패킷을 나타내는 ACK를 수신한 경우, 프로세스는 블록(1230)에서 계속되며, 상기 블록은 마이크로 프로세서(202)가 동일한 엔드-투-엔드 ACK가 수신된 횟수를 카운트하는데 사용되는 카운터 x_{ACC}를 증가시키도록 지시한다. 동일한 엔드-투-엔드 ACK 시퀀스 번호를 여러 번 수신하면 데이터 패킷이 목적지 디바이스에 도달하지 않고 있기 때문에 메시 네트워크를 통한 목적지로의 링크를 더 이상 사용할 수 없음을 나타낸다. 그런 후, 블록(1230)은 마이크로 프로세서(202)를 블록(1242)으로 지시한다.

[0141] 블록(1228)에서 소스 디바이스가 이전에 동일한 다음 예상 데이터 패킷을 나타내는 ACK를 수신하지 않은 경우, 엔드-투-엔드 ACK는 데이터 흐름의 모든 이전 데이터 패킷이 수신되었음을 확인하고 블록(1232)은 마이크로 프로세서(202)로 하여금 카운터 x_{ACC}를 0으로 재설정하도록 지시한다. 블록(1232)은 또한 마이크로 프로세서(202)가 데이터 흐름과 관련된 엔드-투-엔드 타임아웃 카운터를 중단시키도록 지시하고, 이는 상술한 바와 같이 프로세스 스레드(1110)의 블록(1130)에서 시작되었다.

[0142] 블록(1234)은 마이크로 프로세서(202)로 하여금 수신이 전송 큐(332)로부터 확인응답된 데이터 패킷을 제거하도록 지시한다. 도 11a를 다시 참조하면, 블록(1126)에서, 할당된 전송 큐(258)로부터의 데이터 패킷이 선택된 링크 큐(338, 340 또는 342)에 기록되나, 블록(1234)이 마이크로 프로세서(202)로 하여금 엔드-투-엔드 ACK를 처리한 후 데이터 패킷을 제거하도록 지시할 때까지 전송 큐로부터 제거되지 않는다. 따라서, 승인이 보류중인 전송 큐(332)의 헤드에서 순차적으로 유지되는 몇몇 데이터 패킷이 있을 수 있다. 엔드-투-엔드 ACK 프로세스 스레드(1160)와 관련하여 설명된 바와 같이, 엔드-투-엔드 ACK는 목적지 디바이스(702)에서 수신된 모든 데이터 패킷에 대해 전송되지 않을 수 있지만, x_{0max} 순서 패킷을 수신했거나 엔드-투-엔드 ACK 타이머가 만료될 때 전송된다. 소스에서 수신된 엔드-투-엔드 ACK는 전송 큐(332)의 헤드에서 유지되고 있는 데이터 패킷이거나 전송 큐의 헤드에 있는 몇몇 패킷에 대응하는 멀티 홉 multi-hop seq 필드(1324)의 시퀀스 번호를 가질 수 있다. 어느 경우이든, 소스 ACK 처리 스레드(1220)의 블록(1234)은 엔드-투-엔드 ACK의 시퀀스 번호보다 작은 시퀀스 번호를 갖는 전송 큐(332)로부터 모든 데이터 패킷을 제거하도록 마이크로 프로세서(202)에 지시할 것이다.

[0143] 블록(1234)의 실행 후에, 수신된 엔드-투-엔드 ACK 내의 시퀀스 번호에 대응하는 데이터 패킷(즉, 확인응답을 보류하는 다음 데이터 패킷)이 존재하면, 데이터 패킷은 적용 가능한 전송 큐(260)의 헤드에 뒤이어 (있다면) 남아있는 데이터 패킷으로 이동한다. 블록(1236)은 마이크로 프로세서(202)로 하여금 추가 패킷이 전송 큐(332)에 남아 있는지 여부를 결정함으로써 데이터 흐름에 추가 데이터 패킷이 아직 목적지 디바이스에 의해 인지되어 있는지 여부를 결정하도록 지시한다. 블록(1236)에서 확인응답을 대기하는 적어도 하나의 데이터 패킷이 전송 큐(332)에 유지되면, 블록(124)은 마이크로 프로세서(202)를 블록(1238)으로 지시한다. 블록(1238)은 마이크로 프로세서(202)로 하여금 데이터 패킷으로부터 timestamp 필드(1330)를 판독하고 상기 timestamp 값에 기초하여 엔드-투-엔드 타임아웃 타이머를 리셋하도록 지시한다. 따라서 엔드-투-엔드 타임아웃은 확인응답을 대기중인 다음 데이터 패킷의 실제 전송 시간을 기반으로 업데이트된다. 블록(1238)은 마이크로 프로세서(202)를 블록(1250)으로 지시한다. 블록(1236)에서, 전송될 적용 가능한 전송 큐(260)에 남아있는 데이터 패킷이 없다면, 마이크로 프로세서(202)는 또한 블록(1250)으로 지시된다.

[0144] 전송 혼잡

[0145] 엔드-투-엔드 ACK를 처리한 소스 ACK 처리 스레드(1220)의 블록(1230, 1236 및 1238)에서, 마이크로 프로세서(202)는 네트워크 혼잡 프로세스 스레드(1240)를 실행하도록 지시된다. 네트워크 혼잡 프로세스 스레드(1240)는 메시 네트워크(100)를 통해 데이터 흐름에 의해 경험된 혼잡을 모니터하고 이에 따라 소스 디바이스(300)로부터의 전송을 적응시킨다. 블록(1250)에서, 마이크로 프로세서(202)는 메시 네트워크(100)의 전송 상태를 결정하도록 지시된다. 이 실시예에서, 메시 네트워크(100)를 통한 데이터 흐름에 대한 3 가지 가능한 전송 상태가 소스 디바이스에서 구현된다. 슬로우 스타트 전송 상태는 소스 디바이스(300)에서 데이터 흐름을 개시할 때 구현된다. 슬로우 스타트 전송 상태 하에서, 신뢰할 수 있는 전송 프로세스는 혼잡 윈도우 크기 CW = 1(데이터 패킷)로 시작한다. 혼잡 회피 상태 및 빠른 복구 상태도 아래 설명된대로 구현된다. 블록(1250)은 현재 전송 상태가 슬로우 스타트로 설정되어 있는지 여부를 결정하도록 마이크로 프로세서(202)에 지시하고,이 경우에 마이크로 프로세서는 블록(2522)으로 지시된다. 블록(1252)은 마이크로 프로세서(202)로 하여금 혼잡 윈도우(CW)를 목적지 디바이스로부터 수신된 엔드-투-엔드 ACK에서 확인된 데이터 패킷의 수(즉, # ACK)만큼 증가하게 지시한다. 이 완만한 증가는 소스 디바이스(300)가 다수의 데이터 패킷을 전송하는 것을 방지한 다음 메시 네트워크(100)가 트래픽 혼잡인지의 여부를 판정할 기회가 있다. 혼잡 윈도우는 프로세스 스레드(1110)와 관련하여 상술한 바와 같이 전송 큐(332) 내에서 유지되고, 메시 네트워크(100)가 혼잡하지 않은 경우, 혼잡 윈도우의 크기가 증가될 수 있도록 메시 서비스 전송 큐 위치(260) 내의 충분한 저장이 할당된다.

[0146] 프로세스 스레드(1240)는 블록(1254)에서 계속되며, 여기서 마이크로 프로세서(202)는 혼잡 윈도우(CW)의 현재 크기가 혼잡 윈도우 임계 크기(CW_{TH})보다 큰지 여부를 결정하도록 지시된다. 혼잡 윈도우 임계 크기(CW_{TH})는 신뢰할 수 있는 데이터 전송 프로세스 동안 변경될 수 있지만 사전 정의된 최소 크기 이상으로 유지될 것이다. 블록(1254)에서, 혼잡 윈도우(CW)의 크기가 CW_{TH} 미만으로 유지되면, 마이크로 프로세서(202)는 블록(1222)으로 되돌아 간게 지시되어 다음 엔드-투-엔드 ACK를 대기한다.

[0147] 혼잡 윈도우 크기가 블록(1254)에서 CW_{TH}에 도달하면, 마이크로 프로세서(202)는 전송 상태가 혼잡 회피로 설정되는 블록(1256)으로 지시된다. 마이크로 프로세서(202)는 블록(1222)으로 다시 지시되어 다음 엔드-투-엔드 ACK를 대기한다. 따라서, 신뢰할 수 있는 전송 프로세스는 보수적인 전송 속도로 전송을 시작하고 혼잡 윈도우의 크기를 임계값 CW_{TH}까지 증가시킴으로써 전송 속도를 증가시킬 것이다.

[0148] 블록(1250)에서, 전송 상태가 슬로우 스타트로 설정되지 않으면, 마이크로 프로세서(202)는 블록(1258)으로 지시되고, 상기 블록에서 마이크로 프로세서는 전송 상태가 혼잡 회피로 설정되는지 여부를 결정하도록 지시된다. 전송 상태가 혼잡 회피로 설정되면, 블록(1258)은 마이크로 프로세서(202)를 블록(1260)으로 지시한다. 블록(1260)에서, 마이크로 프로세서(202)는 각 확인된 데이터 패킷에 대한 혼잡 윈도우(CW)의 크기를 분수 k/CW만큼 증가 시키도록 지시되고, 여기서 k는 1, 2, 3 등의 정수 값을 가질 수 있다. 용어 k/CW는 일반적으로 10진수 값과 같을 것이고 혼잡 윈도우(CW)의 크기가 훨씬 느리게 증가할 것이다. 혼잡 윈도우 크기는 정수의 데이터 패킷으로 표현되기 때문에, 혼잡 윈도우의 크기는 다른 데이터 패킷이 혼잡 윈도우 크기에 추가되도록 연속적인 분수 증분이 합산될 때만 증가할 것이다. 혼잡 회피 상태 하에서 혼잡 윈도우는 단지 느린 속도로 크기 증가한다. 마이크로 프로세서(202)는 블록(1222)으로 다시 지시되고 다음 엔드-투-엔드 ACK를 기다린다.

[0149] 블록(1258)에서, 전송 상태가 혼잡 회피로 설정되지 않으면, 현재 전송 상태는 빠른 복구이고, 마이크로 프로세서(202)는 혼잡 윈도우가 혼잡 윈도우 임계 크기 CW_{TH}로 설정되는 블록(1262)으로 지시된다. 마이크로 프로세서

(202)는 다음 엔드-투-엔드 ACK를 기다리기 위해 블록(1222)으로 다시 지시된다. 따라서, 블록들(1250-1256)은 목적지 디바이스(70)에서 수신되는 데이터 패킷들의 성공적인 확인응답이 있을 때 혼잡 윈도우의 크기를 증가시키며, 이는 메시 네트워크(100)가 엔드-투-엔드 전송 링크가 확립되었고 아직 트래픽이 혼잡되지 않은 것을 나타낸다.

[0150] 블록(1228)에서 동일한 시퀀스 번호를 갖는 하나 이상의 엔드-투-엔드 ACK가 소스 디바이스(300)에서 수신될 때, x_{ACK} 카운터는 상술한 바와 같이 증가되고 마이크로 프로세서(202)는 네트워크 혼잡 프로세스 스레드(1240)의 블록(1242)으로 지시된다. 블록(1242)은 현재 전송 상태가 슬로우 스타트인지 혼잡 회피인지를 결정하도록 마이크로 프로세서(202)에게 지시하고, 이 경우 프로세스는 블록(1244)에서 계속된다. 블록(1244)은 x_{ACK} 카운터가 (일 실시예에서, 3으로 설정될 수 있는) ACK_{MAX} 임계값에 도달했는지를 결정하도록 마이크로 프로세서에 지시한다. 그런 후 블록(1246)은 마이크로 프로세서(202)가 전송 상태를 빠른 복구로 설정하도록 지시한다. 슬로우 스타트 또는 혼잡 회피 전송 상태에 있을 때, 소스 디바이스(300)는 목적지 디바이스(702)에 의해 예상되는 데이터 패킷을 재전송하기 전에 대기한다. 블록(1242)에서 x_{ACK} 카운터가 아직 ACK_{MAX} 임계값에 도달하지 않은 경우, 마이크로 프로세서(202)는 블록(1222)으로 지시되어 다음 엔드-투-엔드 ACK를 대기한다.

[0151] 블록(1242)에서, 마이크로 프로세서(202)는 전송 상태가 이미 빠른 복구로 설정되었다고 결정하면, 마이크로 프로세서는 블록(1248)으로 지시되고, 상기 블록에서 혼잡 윈도우(CW)의 크기가 단일 데이터 패킷에 의해 증가된다. 이 경우, 전송이 실패한 일부 데이터 패킷이 존재하지만, 일반적으로 데이터 패킷은 여전히 목적지 디바이스에 도달하고 있으며 이는 혼잡 윈도우를 약간 증가시키는 것이 문제가되지 않아야 한다고 가정한다.

[0152] 소스 디바이스(300)는 또한 엔드-투-엔드 타이머를 모니터링하기 위해 재전송 프로세스 스레드(1270)를 실행하여 현재 전송 큐(332)의 헤드에 있는 데이터 패킷에 대한 타임아웃이 발생했는지를 결정한다. 블록(1272)에서 데이터 패킷에 있는 timestamp 필드(1330)가 미리 결정된 임계 시간을 초과하면, 마이크로 프로세서(202)는 블록(1272)으로 지시되고, 상기 블록에서 엔드-투-엔드 ACK 시퀀스 번호에 대응하는 데이터 패킷이 재전송된다. 엔드-투-엔드 타임아웃 타이머는 또한 재전송 시간에 기초하여 재시작되고, 데이터 패킷의 timestamp 필드(1330)는 그에 따라 업데이트된다.

[0153] 그런 후, 블록(1274)은 혼잡 윈도우(CW)의 크기를 단일 데이터 패킷으로 재설정하도록 마이크로 프로세서(202)에 지시한다. 혼잡 윈도우 임계 크기 CW_{TH}는 또한 혼잡 윈도우의 현재 크기의 절반으로 설정되고 엔드-투-엔드 ACK의 예상 패킷이 재전송 되었기 때문에 x_{ACK} 카운터는 0으로 리셋된다.

[0154] 전송 우선순위

[0155] 도 11b를 다시 참조하면, 일 실시예에서 우선순위 전송이 싱글-홉 전송 프로세스 스레드(1140)에서 구현될 수 있다. 블록(1150)에서, 링크 큐(338)의 헤드로부터 데이터 패킷을 단순히 처리하기보다는 마이크로 프로세서(202)는 특정 유형의 데이터 패킷의 전송을 우선하도록 지시될 수 있다. 혼잡 메시 네트워크(100)를 통해 대량의 콘텐츠 데이터가 전송될 때, 정상 전송 시퀀스에서 처리된 콘텐츠 데이터는 엔드-투-엔드 ACK 및 싱글-홉 ACK 패킷이 지연되게 할 수 있다. 이러한 확인응답 데이터 패킷의 지연으로 패킷이 목적지에서 실제로 수신되고 엔드-투-엔드 ACK에서 확인될 때 소스 디바이스에 의해 데이터 패킷이 재전송되거나 싱글-홉 ACK가 지연될 때 무선 링크에 의한 재전송이 시도될 수 있다. 상기 참조된 미국 가출원 제 62/343,056 호에 기술된 다른 컨트롤 메시지(HELLO, JOIN, LEAVE 등)는 그 전체가 참조로 본 명세서에 포함되고, 메시 네트워크(100)의 설정과 관련되며 이러한 메시지의 지연은 네트워크의 효율적인 확장을 방지하고, 따라서 혼잡을 더욱 증가시킨다. 일 실시예에서, 블록(1150)은 마이크로 프로세서(202)로 하여금 각 무선 링크 큐(262)의 데이터 패킷을 우선순위대로 처리하도록 지시할 수 있다. 예를 들어, 컨트롤 패킷의 확인응답에 최고 우선순위가 할당되고 컨트롤 패킷 자체에 대한 다음 우선순위가 할당될 수 있다. 엔드-투-엔드 및 싱글-홉 ACK 데이터 패킷에 낮은 우선순위가 할당될 수 있다. 콘텐츠 데이터 패킷에 가장 낮은 우선순위가 할당될 수 있다. 이 대안적인 실시예에서, 블록(1150)은 할당된 우선순위에 따라 마이크로 프로세서(202)가 링크 전송 큐를 비우도록 지시한다. 다른 실시예에서, 상이한 유형의 콘텐츠 데이터 흐름에 상이한 우선순위가 할당될 수 있다.

[0156] 멀티 캐스트 전송

[0157] 멀티 캐스트 데이터 전송 프로세스 실시예가 도 13에 도시되어 있다. 멀티 캐스트 데이터 전송 프로세스는 소스 디바이스(예컨대, 도 7의 디바이스(300))로부터 다수의 목적지 디바이스로의 전송을 수반하고, 하나 이상의 라우팅 디바이스(예를 들어, 디바이스(700))를 더 포함할 수 있다.

[0158] 도 13a를 참조하면, 멀티 캐스트 그룹에 가입하기 위해 디바이스에 의해 실행된 가입 프로세스는 1340으로 도시되고, 디바이스의 사용자가 멀티 캐스트 그룹에 가입 요청을 시작할 때 블록(1342)에서 시작한다. 일 실시예에서, 멀티 캐스트 그룹은 애플리케이션들(302, 304, 704 및 710) 중 하나와 연관될 수 있고, 멀티 캐스트 그룹에 가입하기 위한 사용자로부터의 요청이 애플리케이션 내에서 수신되어, 가입 요청을 처리하는 API에 대한 호출로서 메시 서비스(310)에 전송될 수 있다. 각각의 그룹은 그룹 ID 블록(1344)에 의해 식별된 다음 소스 디바이스의 마이크로 프로세서(202)로 하여금 그룹이 이미 가입되었는지를 결정하도록 지시한다. 메시 네트워크(100)에서 디바이스가 발견될 때, 이들은 라우팅 테이블 위치(264)에 추가되고 이들 디바이스가 가입된 groupID는 라이팅 테이블에서 디바이스와 연관된다. 그룹이 이미 가입된 경우, 사용자가 가입하고자하는 그룹 ID와 일치하는 그룹 ID를 갖는 디바이스에 대한 라우팅 테이블 위치(264)에 적어도 하나의 엔트리가 있을 것이다. 라우팅 테이블 위치(264)에 이미 일치하는 그룹 ID가 존재한다면, 마이크로 프로세서는 블록(1346)으로 지시되고, 여기서 마이크로 프로세서는 그룹이 이전에 가입되었다는 경고를 사용자에게 발행하도록 지시된다.

[0159] 블록(1344)에서 라우팅 테이블 위치(264)에 아직 일치하지 않은 그룹 ID가 있으면, 마이크로 프로세서는 블록(1348)으로 지시된다. 블록(1348)은 마이크로 프로세서(202)가 그룹 ID를 디바이스 상에 가입된 멀티 캐스트 그룹의 리스트에 추가하도록 지시한다. 블록(1350)은 이어서 멀티 캐스트 그룹에 또한 가입된 메시 네트워크상의 디바이스들의 리스트를 수신하기 위해 요청이 전송될 메시 네트워크(100)상의 타겟 디바이스를 결정하도록 마이크로 프로세서(202)에 지시한다. 타겟 디바이스는 일반적으로 소스 디바이스가 연결된 마스터 모드의 액세스 포인트 디바이스와 같이 네트워크를 통한 싱글-홉에 의해 소스 디바이스로부터 분리된 메시 네트워크(100)상의 디바이스일 수 있다.

[0160] 블록(1352)은 마이크로 프로세서(202)로 하여금 적용 가능한 그룹 ID와 함께 디바이스 리스팅을 포함하는 요청을 생성하도록 지시한다. 리스팅은 소스 디바이스 및 가입된 그룹 ID, 및 소스 디바이스에 의해 메시 네트워크(100)를 통해 도달할 수 있지만 타겟 디바이스를 통해서는 도달할 수 없은 다른 디바이스의 리스팅을 포함해야 한다. 요청은 각 디바이스에 대해 가입된 그룹 ID를 추가하여 (일반적으로 위에서 설명한 것처럼) HELLO 또는 JOIN 요청의 형태를 취할 수 있다. 블록(1354)은 마이크로 프로세서(202)로 하여금 요청을 타겟 디바이스로 전송하도록 지시한다. 프로세스(1340)는 이어서 블록(1356)에서 계속되며, 상기 블록은 소스 디바이스(300)의 마이크로 프로세서(202)로 하여금 ACK가 타겟 디바이스로부터 수신되었는지를 결정하도록 지시한다. ACK가 수신되지 않은 경우, 마이크로 프로세서(202)는 블록(1356)으로 되돌아 가고 프로세스(1340)는 ACK를 기다리는 동안 유효하게 중단된다.

[0161] 타겟 디바이스에 의해 실행된 가입 응답 프로세스가 1370에서 도시되고, 메시 네트워크(100)상의 다른 디바이스로부터 타겟 디바이스에서 요청이 수신될 때 블록(1372)에서 시작된다. 블록(1374)은 타겟 디바이스의 마이크로 프로세서(202)로 하여금 요청 메시지에 리스팅을 판독하고 라우팅 테이블 위치(264)를 업데이트하여 요청 메시지에 포함된 소스 디바이스 및 다른 디바이스에 대한 그룹 ID를 추가하게 지시한다. 따라서, 타겟 디바이스는 요청 메시지가 수신될 때마다 라우팅 테이블 위치(264)에서 그의 라우팅 테이블을 업데이트한다.

[0162] 그런 후, 블록(1378)은 마이크로 프로세서(202)로 하여금 각각의 가입된 그룹 ID와 함께 타겟 디바이스로부터 도달 가능한 디바이스의 리스팅을 생성하도록 지시한다. 요청을 발생한 소스 디바이스는 리스팅에서 제외된다. 블록(1376)은 또한 마이크로 프로세서(202)로 하여금 도달 가능한 디바이스의 리스팅을 포함하는 확인응답(HELLO/JOIN ACK)을 생성하도록 지시한다. 그 후, 프로세스(1370)는 블록(1378)에서 계속되며, 상기 블록은 타겟 디바이스의 마이크로 프로세서(202)로 하여금 요청을 발생한 소스 디바이스로 확인응답을 다시 전송하도록 지시한다.

[0163] 프로세스(1340)는 ACK가 타겟 디바이스로부터 수신되고 소스 디바이스의 마이크로 프로세서(202)가 블록(1358)으로 지시될 때 블록(1356)에서 계속된다. 블록(1358)은 소스 디바이스의 라우팅 테이블 위치(264)에서 라우팅 테이블을 업데이트하도록 마이크로 프로세서(202)에게 지시하여 메시 네트워크(100)상의 디바이스에 의해 가입된 멀티 캐스트 그룹을 식별하는 그룹 ID를 포함하게 한다.

[0164] 소스 디바이스(300)와 같은 디바이스들에서 실행되는 가입 취소 프로세스도 또한 1380으로 도 13에 도시되어 있다. 가입 취소 프로세스(1380)는 소스 디바이스(300)의 사용자가 멀티 캐스트 그룹으로부터 가입 취소를 요청할 때 시작된다. 블록(1384)은 소스 디바이스의 마이크로 프로세서(202)로 하여금 그룹이 가입되는지 여부를 결정하도록 지시하고, 이 경우 블록(1386)은 마이크로 프로세서로 하여금 디바이스에 의해 가입된 그룹 ID의 리스팅으로부터 멀티 캐스트 그룹을 제거하도록 지시한다. 라우팅 테이블 위치(264)에 일치하는 그룹 ID가 없으면, 마이크로 프로세서(202)는 블록(1348)으로 지시되고, 여기서 마이크로 프로세서는 그룹이 현재 가입되어 있지 않

다는 경고를 사용자에게 발행하도록 지시된다.

[0165] 네트워크를 구축하는데 사용되는 서버 또는 라우터 인프라 구조에 의해 유지되는 종래의 멀티 캐스트 그룹과 대조적으로, 메시 네트워크(100)는 멀티 캐스트 그룹 정보를 위한 중앙 저장소를 반드시 가질 필요는 없다. 따라서, 멀티 캐스트 그룹 정보는 도 13에 도시된 멀티 캐스트 데이터 전송 프로세스에 따라 메시 네트워크(100)를 통해 전파되고 공유되어야 한다.

[0166] 멀티 캐스트 전송 프로세스는 1400으로 도 13b에 도시되어 있고, 애플리케이션(302, 304, 704 및 710) 중 하나가 특정 그룹 ID에 대응하는 멀티 캐스트 그룹에 콘텐츠 데이터의 전송을 요청할 때 블록(1402)에서 시작한다. 애플리케이션과 메시 서비스(310) 사이의 컨텐츠 데이터의 전송은 일반적으로 목적지가 그룹 ID에 의해 식별되는 uuid에 의해 컨텐츠 데이터의 목적지를 식별하는 것을 제외하고 도 4에 도시된 애플리케이션 이벤트 처리 프로세스(800)의 블록(816 내지 824)에 따라 진행된다. 대상은 그룹 ID로 식별된다. 그룹 ID는 멀티 캐스트 그룹 전송과 관련되어있는 것으로 메시 서비스(310)에 의해 식별 가능한 고유한 포맷을 갖는 숫자일 수 있다. 콘텐츠 데이터는 메시 서비스(310)에 의해 수신되고 도 10a에 도시된 메시 서비스 프로세스(1000)의 블록(1046 내지 1056)에 따라 처리된다.

[0167] 블록(1404)은 마이크로 프로세서(202)로 하여금 메모리(210)의 라우팅 테이블 위치(264)에 있는 디바이스들의 리스팅 및 관련된 그룹 ID를 판독하여 네트워크(100)상의 어느 디바이스가 그룹 ID에 대응하는 멀티 캐스트 그룹에 가입되는지를 결정하도록 지시한다. 이러한 디바이스를 멀티 캐스트 타켓 디바이스라 한다. 블록(1404)은 또한 마이크로 프로세서(202)로 하여금 리스트가 비어 있는지(즉, 멀티 캐스트 타겟 디바이스가 디바이스에 연결된 상태로 남아 있지 않은지)를 결정하도록 지시하는데, 이 경우에 마이크로 프로세서는 블록(1402)으로 되돌아가 멀티 캐스트 전송에 대한 다음 요청을 기다리도록 지시된다.

[0168] 블록(1406)은 마이크로 프로세서(202)로 하여금 그룹 ID에 대응하는 멀티 캐스트 메시 어드레스를 찾도록 지시한다. 이 실시예에서, 메시 네트워크(100)상의 어드레스 그룹은 네트워크상의 임의의 물리적 디바이스에 할당되지 않으며 멀티 캐스트 메시 전송에만 사용된다. 따라서, 멀티 캐스트 메시 어드레스는 특정 어드레스 패턴을 가질 수 있고 멀티 캐스트 그룹 전송을 위해 예약된 네트워크 어드레스 범위로부터 할당될 수 있다. 블록(1406)은 또한 마이크로 프로세서(202)로 하여금 메시 네트워크(100)를 통한 전송을 위해 데이터를 데이터 패킷에 기록하도록 지시한다. 데이터 패킷은 일반적으로 도 13에 도시된 UDP 데이터 패킷(1300)에 대응할 수 있으며, 여기서 전송 디바이스의 어드레스는 source_uuid 필드(1308)에 그리고 멀티 캐스트 메시 어드레스는 destination_uuid 필드(1312)에 기록된다.

[0169] 멀티 캐스트 전송 프로세스(1400)는 블록(1408)에서 계속되며, 여기서 마이크로 프로세서(202)는 블록(1404)에서 식별된 각각의 멀티 캐스트 목적지 디바이스에 대한 넥스트 홉을 결정하도록 지시된다. 각 멀티 캐스트 타겟 디바이스는 멀티 캐스트 목적지 디바이스에 도달할 수 있는 네트워크상의 포워딩 디바이스를 식별하거나 멀티 캐스트 타겟 디바이스가 넥스트 홉인 경우 디바이스 자체를 식별하는 라우팅 테이블 위치(264)에서 넥스트 홉 엔트리를 가져야 한다. 포워딩 디바이스는 멀티 캐스트 목적지 디바이스가 아닐 수 있으므로 멀티 캐스트 패킷을 멀티 캐스트 타겟 디바이스로 포워딩하는 데만 관여할 수 있다. 몇몇 경우에, 멀티 캐스트 타겟 디바이스로 포워딩 디바이스로 식별되는 여러 디바이스가 있을 수 있다. 따라서, 블록(1408)은 마이크로 프로세서(202)로 하여금 멀티 캐스트 타켓 디바이스로의 멀티 캐스트 전송을 위한 멀티 캐스트 포워딩 디바이스로서 작용할 수 있는 한 세트의 넥스트 홉 디바이스를 생성하도록 지시한다.

[0170] 블록(1410)은 마이크로 프로세서(202)로 하여금 인터넷 프로토콜(IP) 멀티 캐스트 포워딩에 의해 도달될 수 있는 멀티 캐스트 포워딩 디바이스 세트에 2이상의 포워딩 디바이스가 있는지를 결정하도록 지시한다. 각 멀티 캐스트 타겟 디바이스에 대한 넥스트 홉 포워딩 디바이스는 소스 디바이스에 인접한 디바이스이기 때문에, 라우팅 테이블은 적용 가능한 무선 링크 또는 전송에 이용 가능한 링크를 식별하는 엔트리를 포함할 것이다. 전송 디바이스가 액세스 포인트 또는 마스터 모드의 Wi-Fi 또는 Wi-Fi 다이렉트 디바이스인 경우 및 블록(1410)에서 마이크로 프로세서(202)가 2이상의 Wi-Fi 또는 Wi-Fi 다이렉트 클라이언트 모드 포워딩 디바이스가 메시 네트워크(100)를 통해 액세스 가능한 것으로 결정한다면, 프로세스는 블록(1412)에서 계속된다. 이들 조건 하에서, 2이상의 클라이언트 모드 포워딩 디바이스는 데이터 패킷의 IP 멀티 캐스트 포워딩을 통해 도달 가능할 것이다.

[0171] 블록(1412)은 마이크로 프로세서(202)로 하여금 데이터 패킷을 IP 멀티 캐스트 패킷으로 캡슐화하도록 지시한다. 이어서, IP 멀티 캐스트 패킷은 무선 링크 큐 위치(262)에서 적용 가능한 큐(338 또는 340)에 기록되고, 데이터 패킷이 IP 멀티 캐스트 프로토콜 패킷으로 전송되는 것을 제외하고는 도 11b에 도시된 싱글-홉 전송 프로세스 스레드(1140)의 블록(1142 내지 1154)에 기술된 바와 같이 전송이 일반적으로 계속된다. 그러나, 멀티

캐스트 전송은 메시 네트워크(100)에서 추가적인 혼잡을 초래할 것이기 때문에, 임의의 형태의 엔드-투-엔드 확인응답을 구현하지 않을 수 있다. 오히려 멀티 캐스트 전송은 베스트 에포트(best effort) 전송을 따를 수 있다.

[0172] 이들 디바이스로의 전송이 완료된 것으로 간주되기 때문에, 블록(1414)은 마이크로 프로세서(202)로 하여금 멀티 캐스트 포워딩 디바이스 세트로부터 인터넷 프로토콜(IP) 멀티 캐스트 포워딩에 의해 도달 가능한 디바이스를 제거하도록 지시한다. 블록(1414)은 마이크로 프로세서(202)를 다시 블록(1416)으로 지시하며, 이는 마이크로 프로세서(202)로 하여금 임의의 멀티 캐스트 포워딩 디바이스가 세트에 남아 있는지 여부를 결정하도록 지시한다. 멀티 캐스트 포워딩 디바이스가 세트에 남아 있지 않으면, 블록(1416)은 마이크로 프로세서(202)를 블록(1402)으로 되돌려 다음 멀티 캐스트 전송을 대기하게 지시한다.

[0173] 블록(1416)에서, 추가 디바이스가 멀티 캐스트 포워딩 디바이스 세트에 남아 있으면, 이들은 유니 캐스트 전송에 의해서만 액세스될 수 있고, 마이크로 프로세서는 블록(1418)으로 지시된다. 블록(1418)은 마이크로 프로세서(202)로 하여금 세트에 남아있는 멀티 캐스트 포워딩 디바이스에 대해 데이터 패킷의 유니 캐스트 전송을 수행하도록 지시한다. 따라서, 데이터 패킷은 무선 링크 큐 위치(262)에서 링크 큐(즉, 블루투스 링크 큐(342))에 기록되고, 도 11b에 도시된 프로세스 싱글-홉 전송 프로세스 스레드(1140)에 따라 전송이 진행된다. 블록(1418)은 마이크로 프로세서(202)를 블록(1402)으로 되돌려 다음 멀티 캐스트 전송을 기다리게 지시한다.

[0174] 블록(1410)에서 인터넷 프로토콜(IP) 멀티 캐스트 포워딩에 의해 도달 가능한 멀티 캐스트 포워딩 디바이스 세트 내의 단일 또는 포워딩 디바이스가 있거나 포워딩 디바이스가 없다면, 마이크로 프로세서는 블록(1418)으로 지시되고, 여기서 마이크로 프로세서(202)는 나머지 멀티 캐스트 포워딩 디바이스에 대한 데이터 패킷의 유니 캐스트 전송을 수행하게 지시된다.

[0175] 메시 네트워크(100)상의 멀티 캐스트 그룹으로의 데이터 패킷의 전송은 소스 디바이스에서 이용 가능한 가장 효율적인 프로토콜에 따라 수행된다. IP 멀티 캐스트 프로토콜을 사용하면 이 프로토콜을 지원하는 무선 링크 처리가 간단해진다.

[0176] 멀티 캐스트 포워딩 프로세스는 1440으로 도 13c에 도시되어 있으며, 도 11c에 도시된 엔드-투-엔드 ACK 프로세스 스레드(1160)의 블록(1164 및 1166) 사이에 삽입될 수 있다. 따라서, 프로세스는 데이터 패킷이 메시 네트워크(100)상의 임의의 디바이스에서 수신될 때 블록(1162)(도 11c)에서 시작한다. 블록(1164)은 수신 디바이스의 마이크로 프로세서(202)로 하여금 상술한 바와 같이, 데이터 패킷이 수신되었다는 것을 확인하는 데이터 패킷을 전송한 싱글-홉 ACK를 다시 전송하도록 지시한다. 블록(1164)은 또한 마이크로 프로세서(202)가 데이터 패킷의 destination_uuid 필드(1312)를 읽도록 지시한다.

[0177] 멀티 캐스트 포워딩 프로세스(1440)의 블록(1442)은 마이크로 프로세서(202)로 하여금 데이터 패킷의 destination_uuid 필드(1312)의 어드레스가 메시 네트워크(100)를 통한 멀티 캐스트 전송을 위해 예약된 어드레스의 범위에 대응하는지 여부를 결정하도록 지시한다. 데이터 패킷의 어드레스가 멀티 캐스트 어드레스가 아니면, 블록(1142)은 마이크로 프로세서(202)를 엔드-투-엔드 ACK 프로세스 스레드(1160)의 블록(1166)으로 다시 지시하고, 상술한 바와 같이, 데이터 패킷의 유니 캐스트 처리를 진행한다. 데이터 패킷의 어드레스가 멀티 캐스트 어드레스인 경우, 블록(1142)은 마이크로 프로세서(202)를 블록(1144)으로 지시하며, 상기 블록은 마이크로 프로세서가 특정 멀티 캐스트 데이터 패킷이 이전에 수신되었는지를 결정하도록 지시한다. 멀티 캐스트 전송의 경우, 동일한 데이터 패킷이 2개의 상이한 멀티 캐스트 포워딩 디바이스로부터 수신될 가능성이 있으며, 이 경우에 블록(1144)은 마이크로 프로세서(202)를 엔드-투-엔드 확인 프로세스 스레드(1160)의 블록(1162)으로 다시 지시해 다음 데이터 패킷의 수신을 대기하게 한다.

[0178] 블록(1444)에서, 멀티 캐스트 데이터 패킷이 멀티 캐스트 포워딩 디바이스에서 이전에 수신되지 않았다면, 프로세스는 블록(1446)에서 계속된다. 블록(1446)은 마이크로 프로세서(202)로 하여금 멀티 캐스트 메시 어드레스를 사용하여 라우팅 테이블 위치(264)에서 대응하는 그룹 ID를 조회하도록 지시한다. 블록(1448)은 그 후 디바이스가 블록(1446)에서 결정된 그룹 ID에 대응하는 멀티 캐스트 그룹에 가입되어 있는지 여부를 결정하도록 마이크로 프로세서(202)에 지시한다. 디바이스가 멀티 캐스트 그룹에 가입되면, 블록(1148)은 마이크로 프로세서(202)를 블록(1448)으로 지시해 도 10a에 도시된 메시 서비스 프로세스(1000)의 블록(1006 내지 1012)에 따라 컨텐츠 데이터를 처리하고 데이터를 적용 가능한 애플리케이션에 전달하게 한다.

[0179] 블록(1448)에서, 디바이스가 멀티 캐스트 그룹에 가입되지 않으면, 마이크로 프로세서(202)는 블록(1452)으로 지시된다. 블록(1452-1464)은 도 13b에 도시된 멀티 캐스트 전송 프로세스(1400)의 블록(1404 및 1408-1418)과

동일하며, 멀티 캐스트 데이터 패킷은 메시 네트워크(100)를 통해 멀티 캐스트 타겟 디바이스로 더 전파될 수 있다. 블록(1464)의 실행 후에, 마이크로 프로세서(202)는 엔드-투-엔드 확인 프로세스 스레드(1160)의 블록(1162)으로 되돌아가 다음 데이터 패킷의 수신을 대기하도록 지시된다.

[0180] 멀티 캐스트 포워딩 프로세스(1440)는 멀티 캐스트 메시 어드레스와 연관된 멀티 캐스트 그룹에 가입된 경우 멀티 캐스트 데이터 패킷이 디바이스로 전달되게한다. 멀티 캐스트 전달 프로세스(1440)는 또한 블록(1452)에서 가입된 디바이스의리스팅에서 발견된 임의의 엔트리가 있는 경우 멀티 캐스트 데이터 패킷이 넥스트 홉 디바이스로 전달되게 한다. 디바이스가 메시 네트워크(100)의 특정 분기상의 마지막 멀티 캐스트 타켓 디바이스인 경우 포워딩이 필요하지 않다. 따라서, 도 13에 도시된 멀티 캐스트 프로세스는 데이터 패킷의 불필요한 전송을 피하면서 데이터 컨텐츠를 다수의 디바이스로 전파하는 것을 용이하게 한다. 이웃 디바이스들 사이의 무선 링크 전송을 위한 싱글-홉 전송 프로세스 스레드(1140)를 구현하는 것 외에도, 프로세스는 신뢰할 수 없은 데이터 전송 기반으로 수행된다.

[0181] 방송 전송

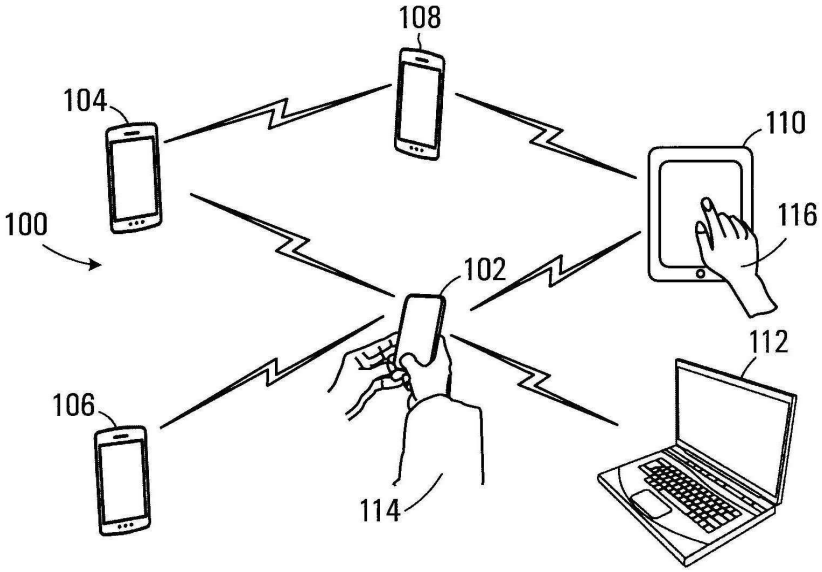
[0182] 브로드 캐스트 전송은 메시 네트워크(100)상의 모든 디바이스가 브로드 캐스트 타겟 디바이스인 것으로 간주된다는 점을 제외하고는 멀티 캐스트 전송과 유사하다. 브로드 캐스트 전송 데이터 패킷은 또한 예약된 멀티 캐스트 어드레스 범위 내의 하나가 아닌 단일 브로드 캐스트 메시 어드레스를 타겟으로 한다. 일 실시예에서, 그 어드레스는 어드레스의 모든 비트가 "1"로 설정됨으로써 설정될 수 있다. 순환 브로드 캐스트 데이터 패킷으로 메시 네트워크(100)를 플러딩하는 것을 피하기 위해, 도 12에 도시된 패킷 데이터 패킷(1300)은 TTL(time-to-live) 필드를 포함하도록 구성될 수 있다. TTL 시간에 도달하면, 디바이스에 의해 데이터 패킷이 더 이상 포워딩되지 않는다. 메시 네트워크(100)상의 모든 디바이스가 브로드 캐스트에 "가입"되므로, 도 13a에 도시된 프로세스(1340 및 1370)는 관련이 없으므로 브로드 캐스트 트래픽에 대해 생략된다. 프로세스들(1400 및 1440)은 멀티 캐스트 전송의 경우에서와 같이 그룹 ID에 기초하여 포워딩 디바이스를 결정하는 대신에, 모든 알려진 디바이스가 포워딩 디바이스로 간주되도록 수정된다. 따라서 소스 디바이스는 모든 알려진 디바이스를 찾고 라우팅 테이블 위치(264)에서 이러한 모든 디바이스로 이어지는 넥스트 홉을 찾는다. 이들 포워딩 디바이스로의 전송은 도 13b에 도시된 프로세스(1400)의 블록(1410-1418)과 유사한 방식으로 진행된다.

[0183] 각각의 브로드 캐스트 포워딩 디바이스는 브로드 캐스트 데이터 패킷이 이전에 수신되었는지를 결정하고, 그렇지 않은 경우, 디바이스상의 관련 애플리케이션으로 전달하기 위해 그리고 메시 네트워크(100)를 통해 다른 브로드 캐스트 타겟 디바이스로 포워딩하기 위해 데이터 컨텐츠를 처리한다. 브로드 캐스트 전송은 소스 어드레스 및 메시 포트 ID를 포함하는 브로드 캐스트 흐름 ID에 의해 식별될 수 있다. 도 11에 도시된 신뢰할 수 있는 데이터 전송 프로세스의 경우와 같이, 고유 한 시퀀스 번호는 브로드 캐스트 전송에서 데이터 패킷의 순서를 식별하는데 사용될 수 있다. 디바이스는 데이터 패킷이 이중으로 수신되는지 여부를 결정하고 동일한 데이터 패킷의 중복 포워딩을 피하기 위해 각각의 브로드 캐스트 전송에 대한 최종 수신 시퀀스 번호를 추적할 수 있다.

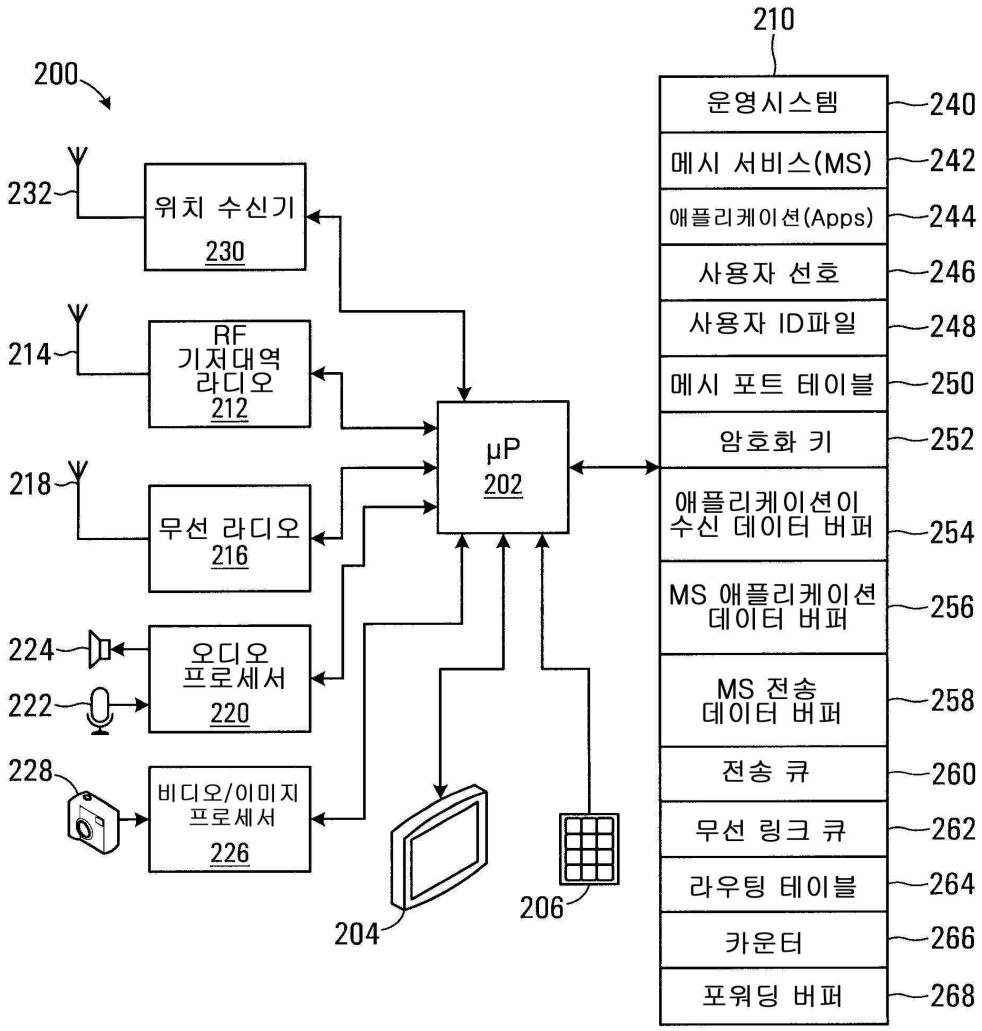
[0184] 특정 실시예들이 설명되고 도시되었지만, 이러한 실시예들은 본 발명을 예시하는 것으로 간주되어야 하며 첨부된 청구범위에 따라 해석되는 본 발명을 제한하는 것이 아니다.

도면

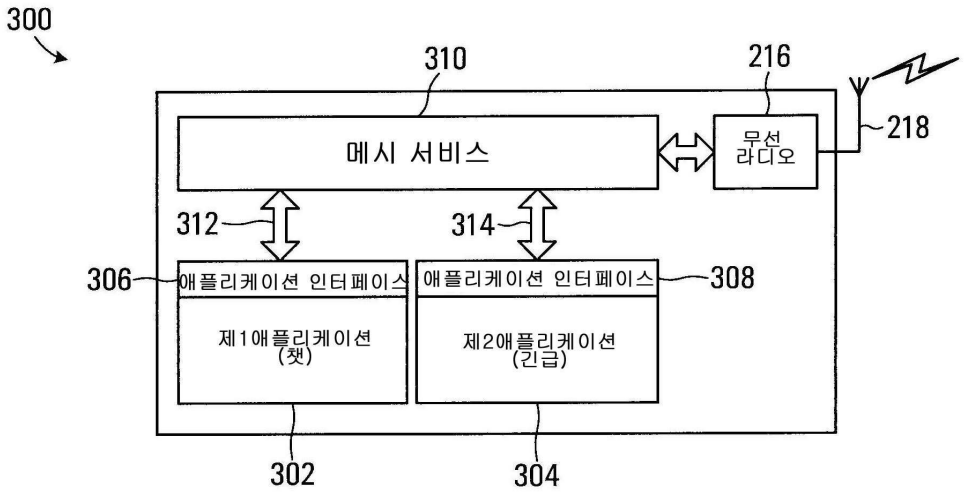
도면1



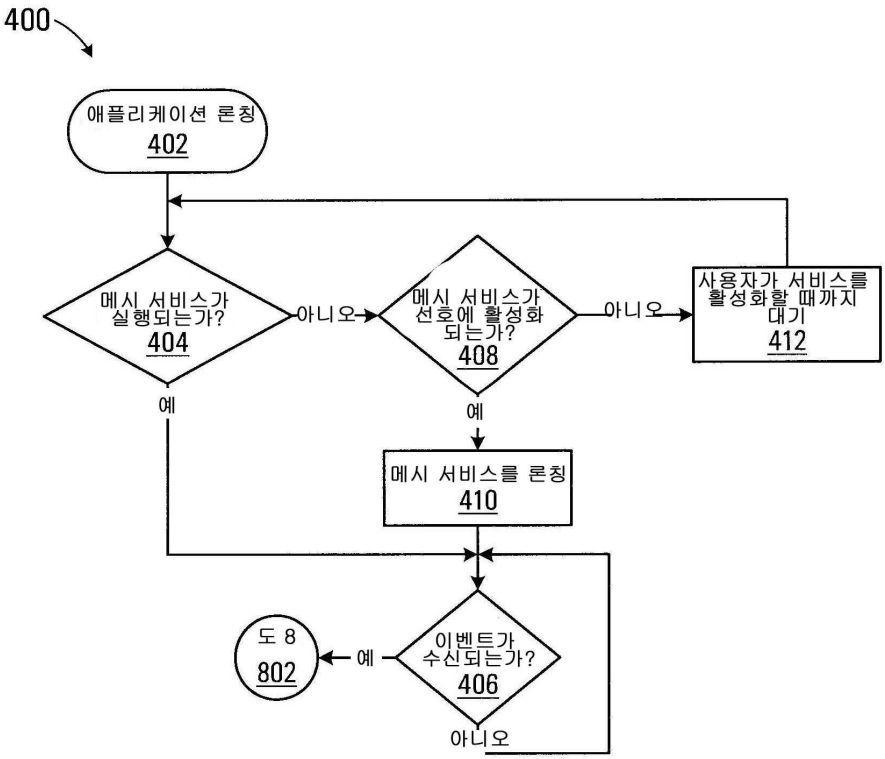
도면2


200
210
운영시스템 240
메시 서비스(MS) 242
애플리케이션(Apps) 244
사용자 선호 246
사용자 ID파일 248
메시 포트 테이블 250
암호화 키 252
애플리케이션이 수신 데이터 버퍼 254
MS 애플리케이션 데이터 버퍼 256
MS 전송 데이터 버퍼 258
전송 큐 260
무선 링크 큐 262
라우팅 테이블 264
카운터 266
포워딩 버퍼 268
232
위치 수신기 230
214
RF 기저대역 라디오 212
218
무선 라디오 216
224
222
오디오 프로세서 220
228
비디오/이미지 프로세서 226
μP 202
204
206

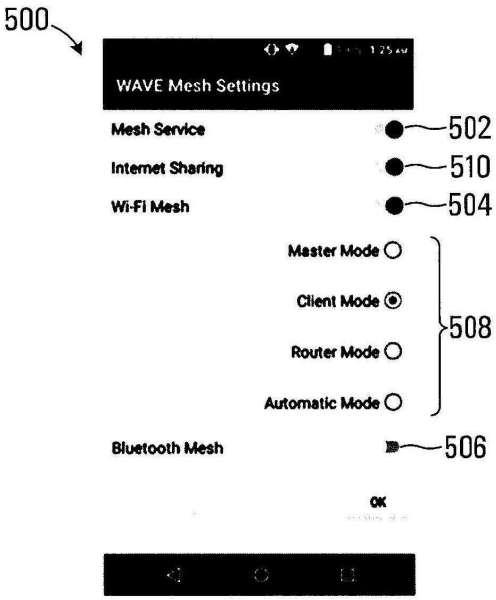
도면3


300
310
216
메시 서비스
무선 라디오
218
312
314
306
애플리케이션 인터페이스
애플리케이션 인터페이스
308
제1애플리케이션 (챗)
제2애플리케이션 (긴급)
302
304

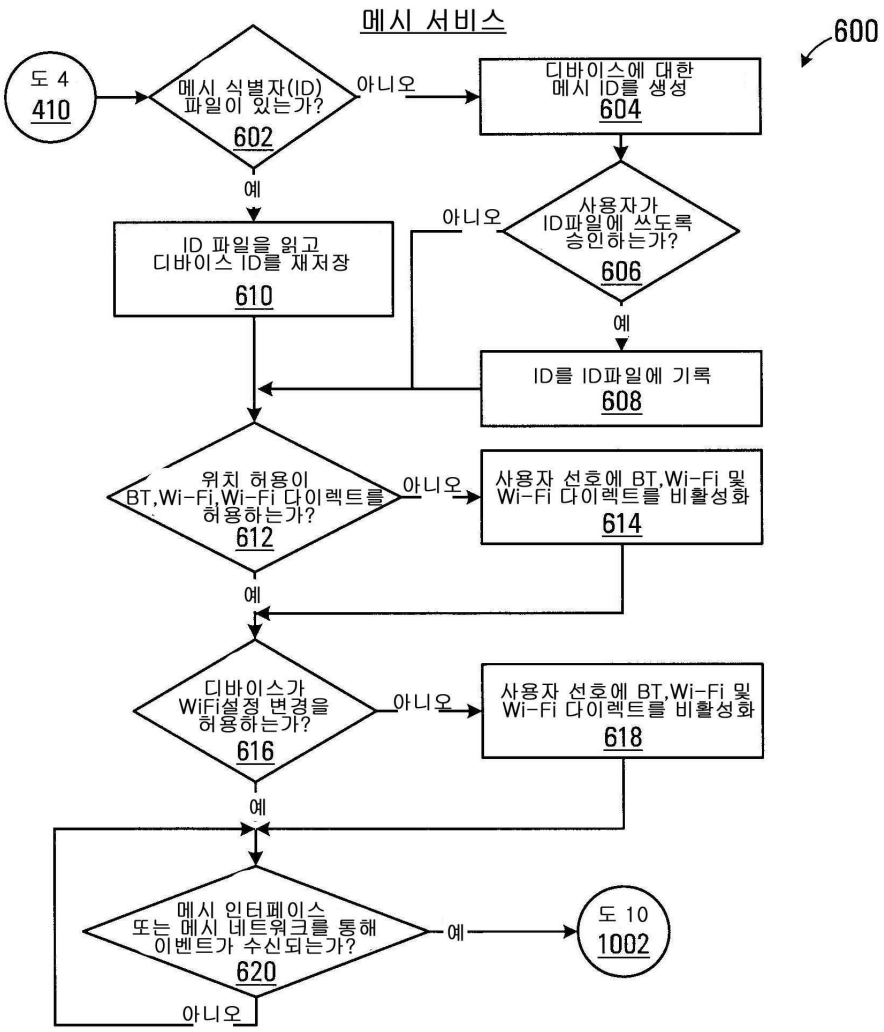
도면4


400
애플리케이션 론칭
402
메시 서비스가 실행되는가?
404
아니오
메시 서비스가 선호에 활성화 되는가?
408
아니오
사용자가 서비스를 활성화할 때까지 대기
412
예
예
메시 서비스를 론칭
410
이벤트가 수신되는가?
406
예
도 8
802
아니오

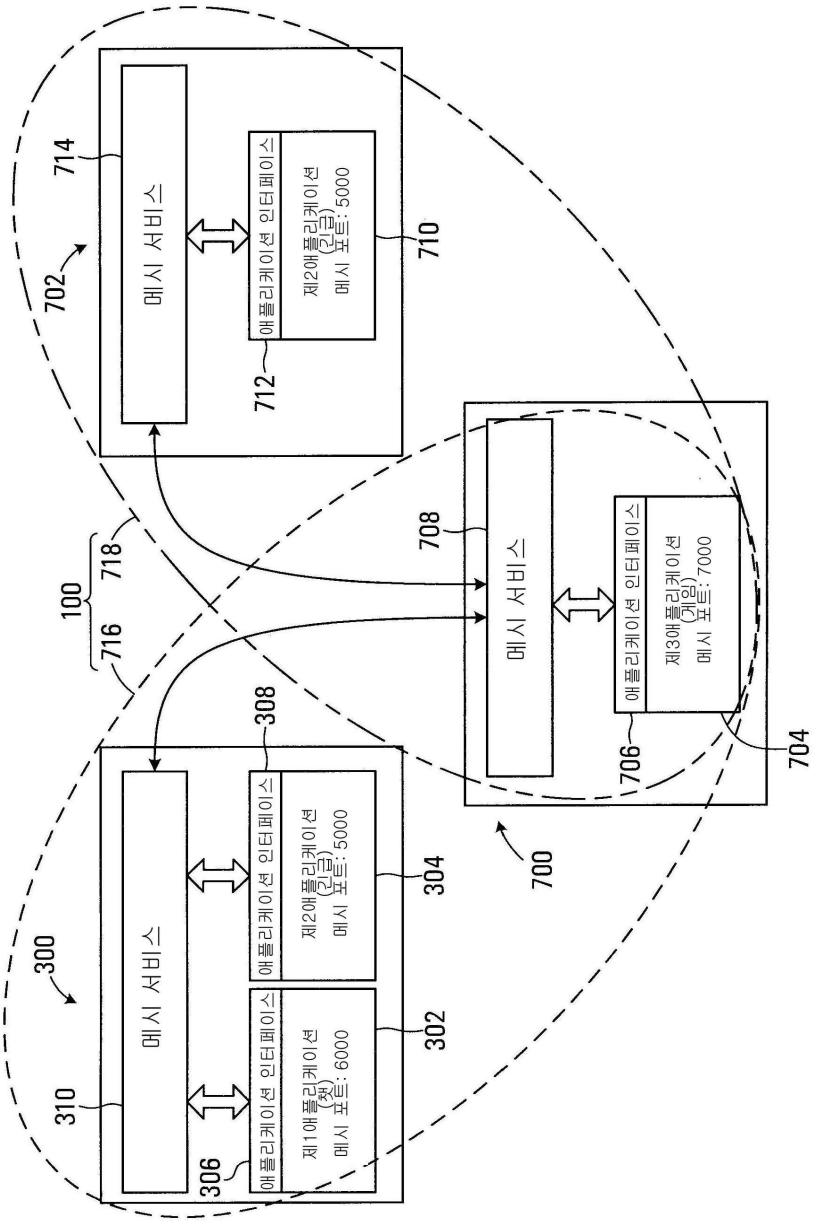
도면5


500
WAVE Mesh Settings
Mesh Service
502
Internet Sharing
510
Wi-Fi Mesh
504
Master Mode
Client Mode
Router Mode
Automatic Mode
508
Bluetooth Mesh
506
OK

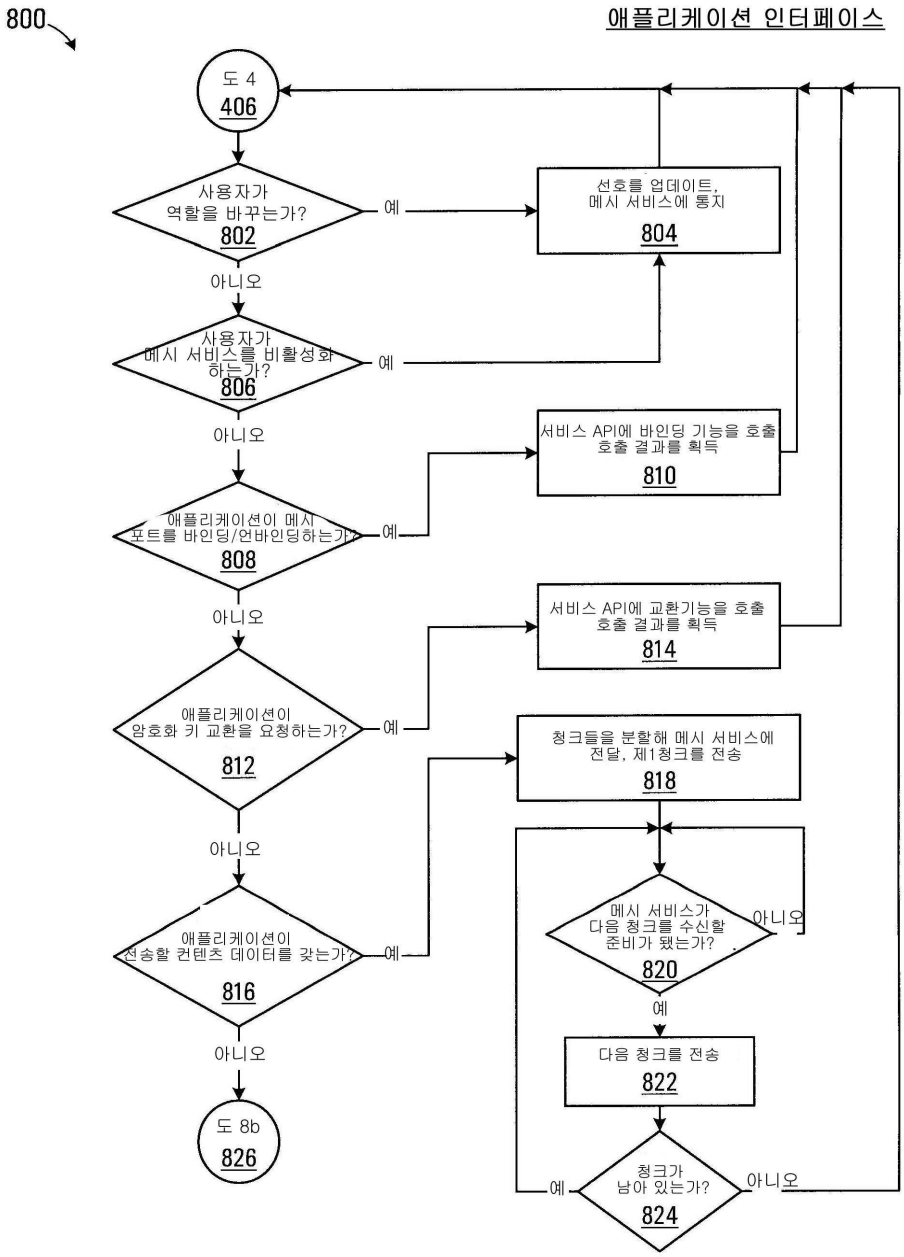
도면6


메시 서비스
600
도 4
410
메시 식별자(ID) 파일이 있는가?
602
아니오
디바이스에 대한 메시 ID를 생성
604
예
ID 파일을 읽고 디바이스 ID를 재저장
610
아니오
사용자가 ID파일에 쓰도록 승인하는가?
606
예
ID를 ID파일에 기록
608
위치 허용이 BT,Wi-Fi,Wi-Fi 다이렉트를 허용하는가?
612
아니오
사용자 선호에 BT,Wi-Fi 및 Wi-Fi 다이렉트를 비활성화
614
예
디바이스가 WiFi설정 변경을 허용하는가?
616
아니오
사용자 선호에 BT,Wi-Fi 및 Wi-Fi 다이렉트를 비활성화
618
예
메시 인터페이스 또는 메시 네트워크를 통해 이벤트가 수신되는가?
620
예
도 10
1002
아니오

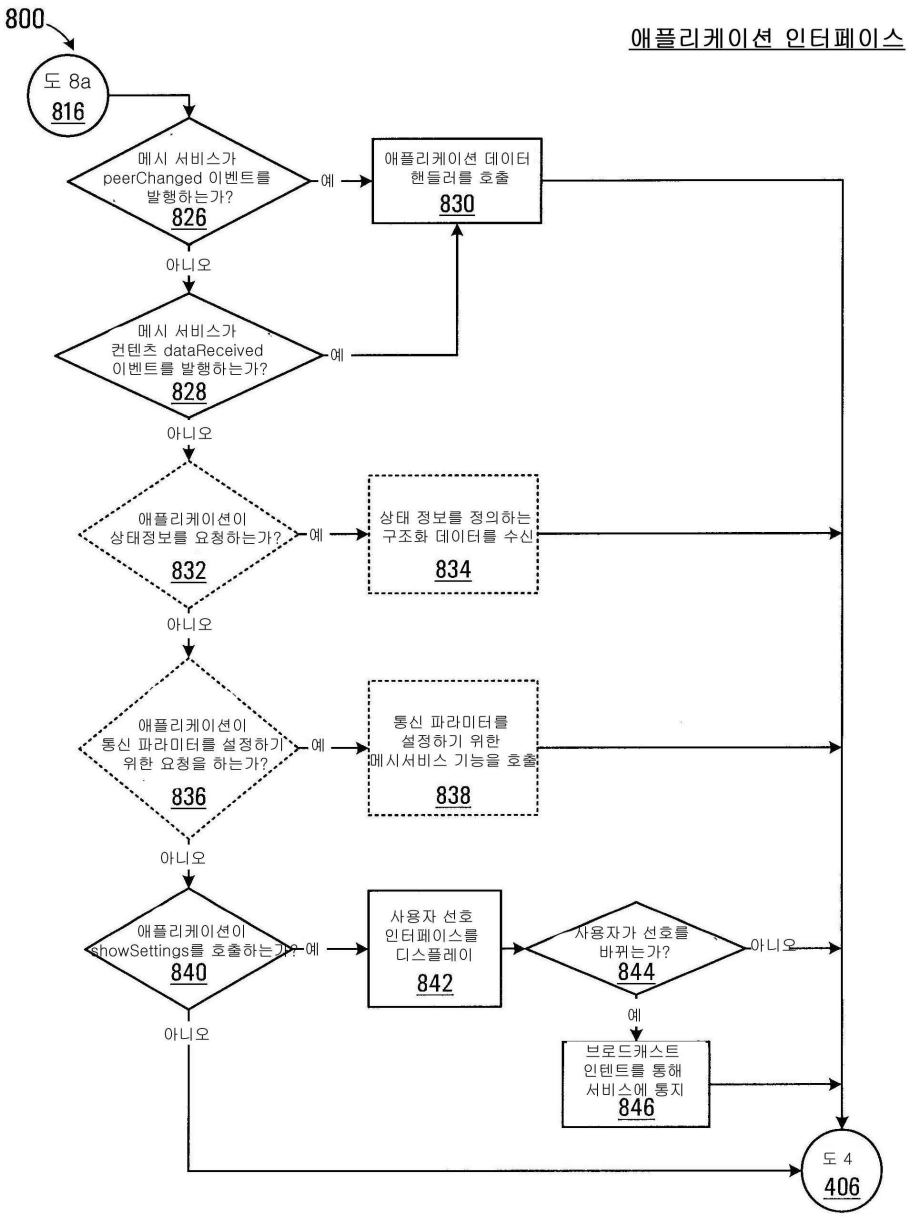
도면7


메시 서비스
애플리케이션 인터페이스
제2애플리케이션 (리그) 메시 포트: 5000
710
712
714
702
메시 서비스
애플리케이션 인터페이스
제3애플리케이션 (게임) 메시 포트: 7000
704
706
708
700
100
716
718
메시 서비스
애플리케이션 인터페이스
제2애플리케이션 (리그) 메시 포트: 5000
304
308
애플리케이션 인터페이스
제1애플리케이션 (챗) 메시 포트: 6000
302
306
310
300

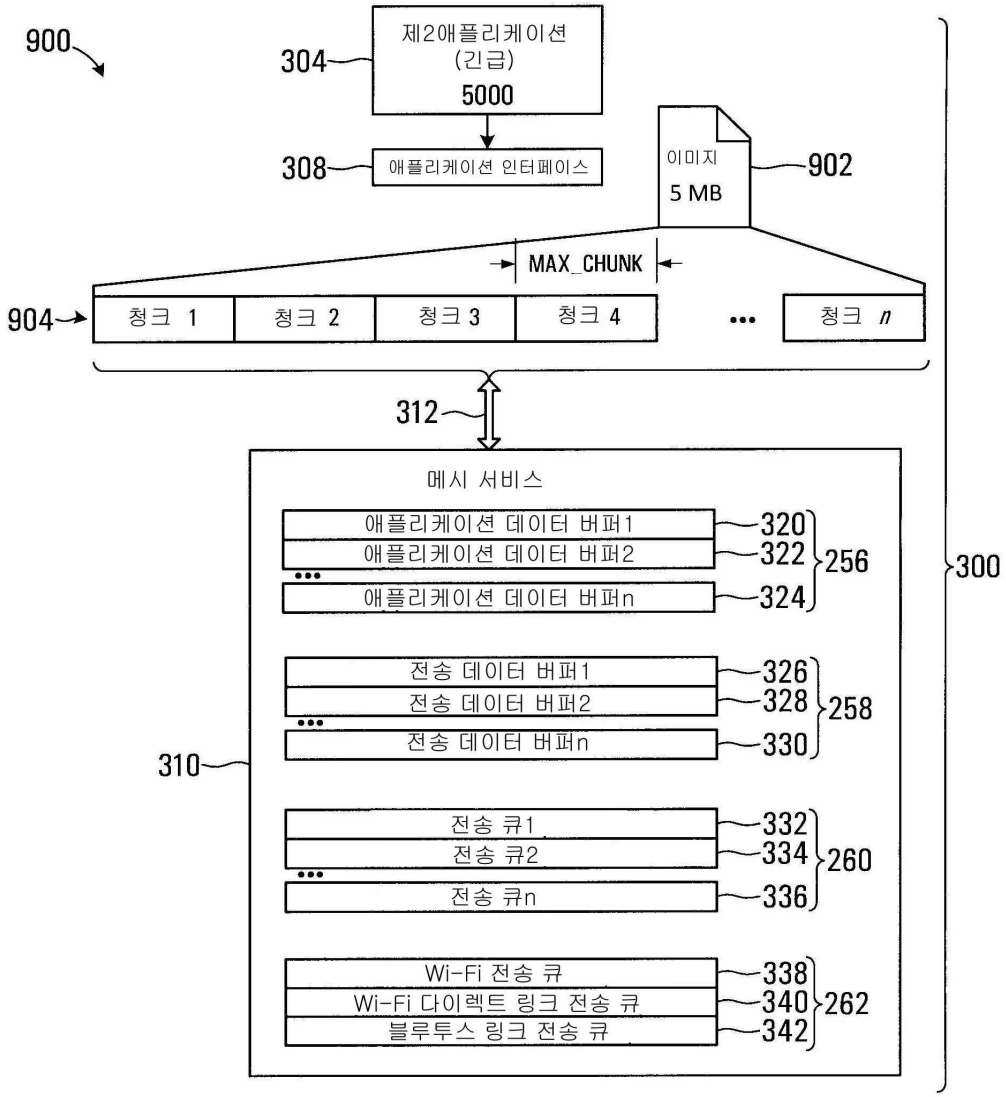
도면8a


800
애플리케이션 인터페이스
도 4
406
사용자가
역할을 바꾸는가?
802
예
선호를 업데이트,
메시 서비스에 통지
804
아니오
사용자가
메시 서비스를 비활성화
하는가?
806
예
아니오
서비스 API에 바인딩 기능을 호출
호출 결과를 획득
810
애플리케이션이 메시
포트를 바인딩/언바인딩하는가?
808
예
아니오
서비스 API에 교환기능을 호출
호출 결과를 획득
814
애플리케이션이
암호화 키 교환을 요청하는가?
812
예
청크들을 분할해 메시 서비스에
전달, 제1청크를 전송
818
아니오
메시 서비스가
다음 청크를 수신할
준비가 됐는가?
820
아니오
애플리케이션이
전송할 컨텐츠 데이터를 갖는가?
816
예
예
다음 청크를 전송
822
아니오
도 8b
826
청크가
남아 있는가?
824
예
아니오

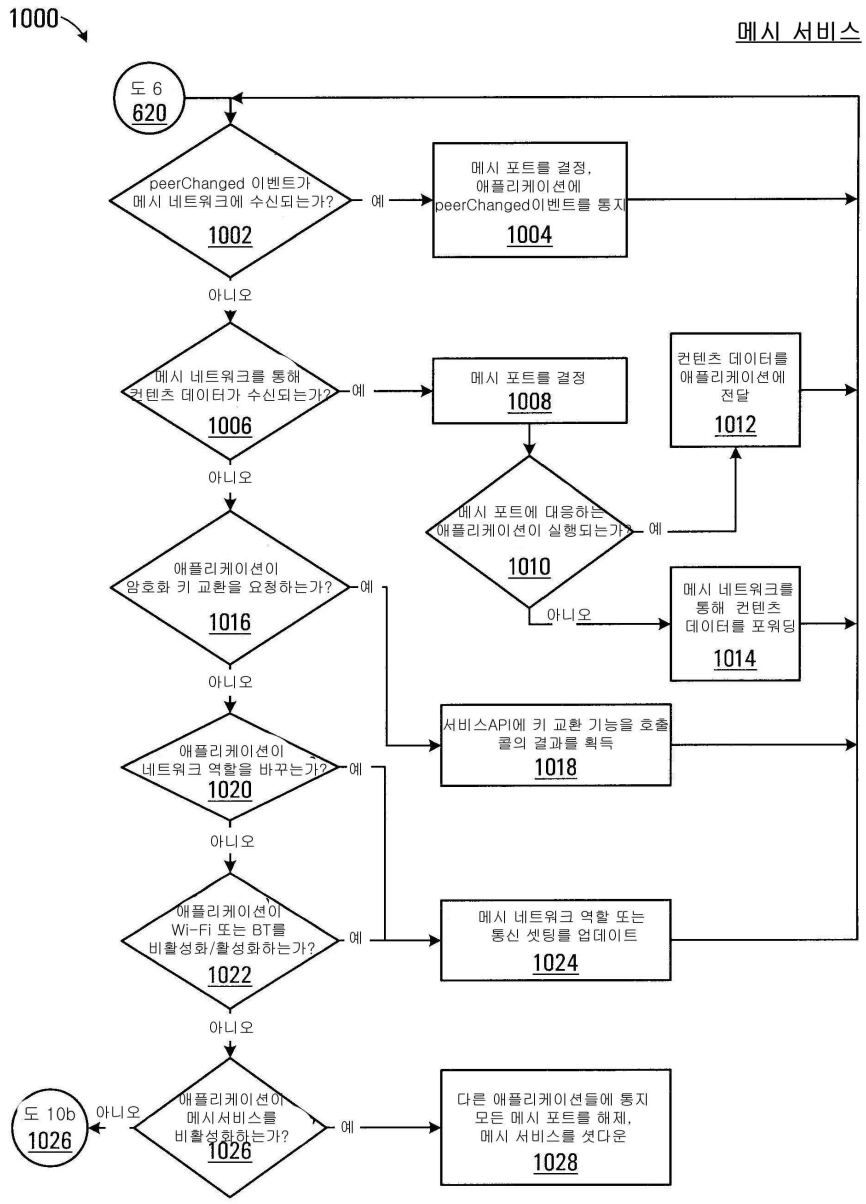
도면8b


800
애플리케이션 인터페이스
도 8a
816
메시 서비스가 peerChanged 이벤트를 발행하는가?
826
예
애플리케이션 데이터 핸들러를 호출
830
아니오
메시 서비스가 컨텐츠 dataReceived 이벤트를 발행하는가?
828
예
아니오
애플리케이션이 상태정보를 요청하는가?
832
예
상태 정보를 정의하는 구조화 데이터를 수신
834
아니오
애플리케이션이 통신 파라미터를 설정하기 위한 요청을 하는가?
836
예
통신 파라미터를 설정하기 위한 메시서비스 기능을 호출
838
아니오
애플리케이션이 showSettings를 호출하는가?
840
예
사용자 선호 인터페이스를 디스플레이
842
사용자가 선호를 바꾸는가?
844
아니오
예
브로드캐스트 인텐트를 통해 서비스에 통지
846
아니오
도 4
406

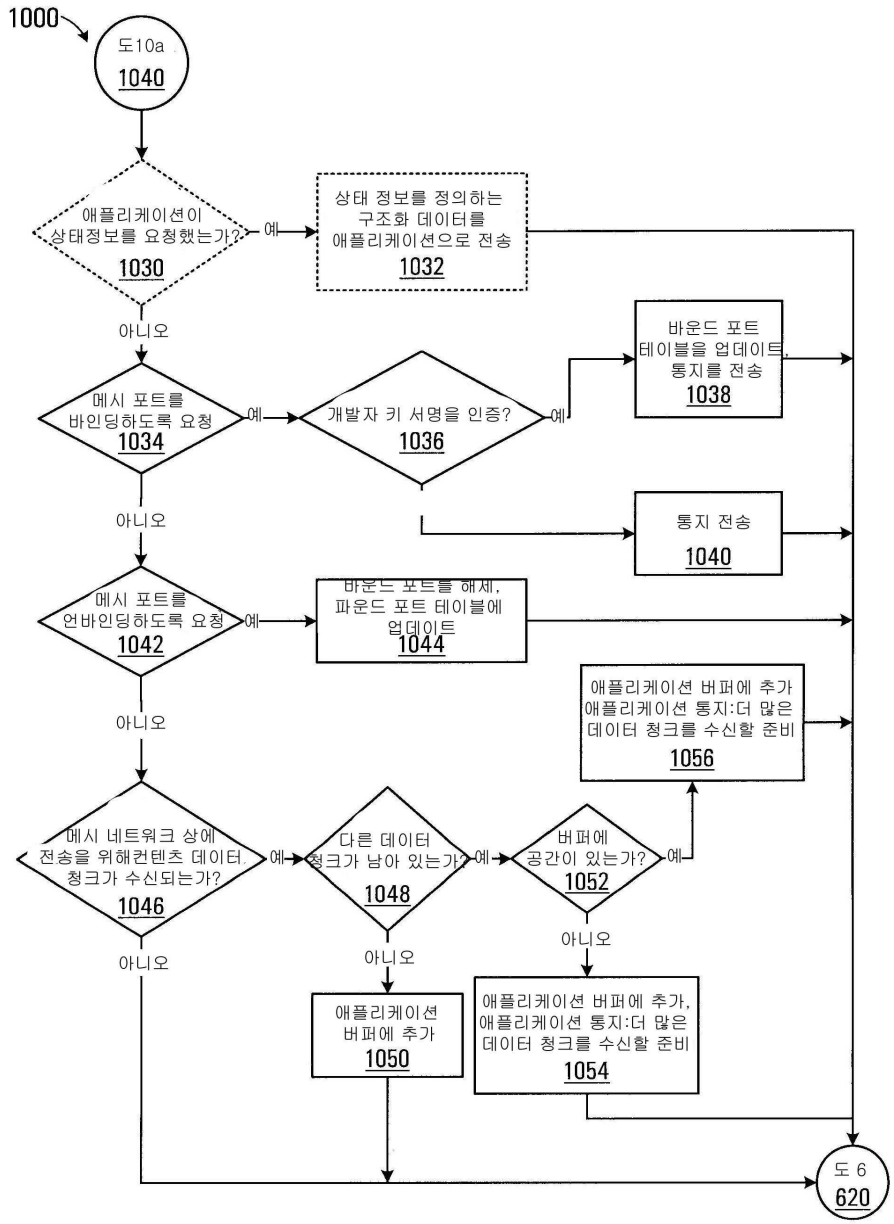
도면9


900
304
제2애플리케이션
(긴급)
5000
308
애플리케이션 인터페이스
이미지
5 MB
902
MAX_CHUNK
904
청크 1
청크 2
청크 3
청크 4
청크 n
312
메시 서비스
애플리케이션 데이터 버퍼1
애플리케이션 데이터 버퍼2
애플리케이션 데이터 버퍼n
320
322
324
256
전송 데이터 버퍼1
전송 데이터 버퍼2
전송 데이터 버퍼n
326
328
330
258
310
전송 큐1
전송 큐2
전송 큐n
332
334
336
260
Wi-Fi 전송 큐
Wi-Fi 다이렉트 링크 전송 큐
블루투스 링크 전송 큐
338
340
342
262
300

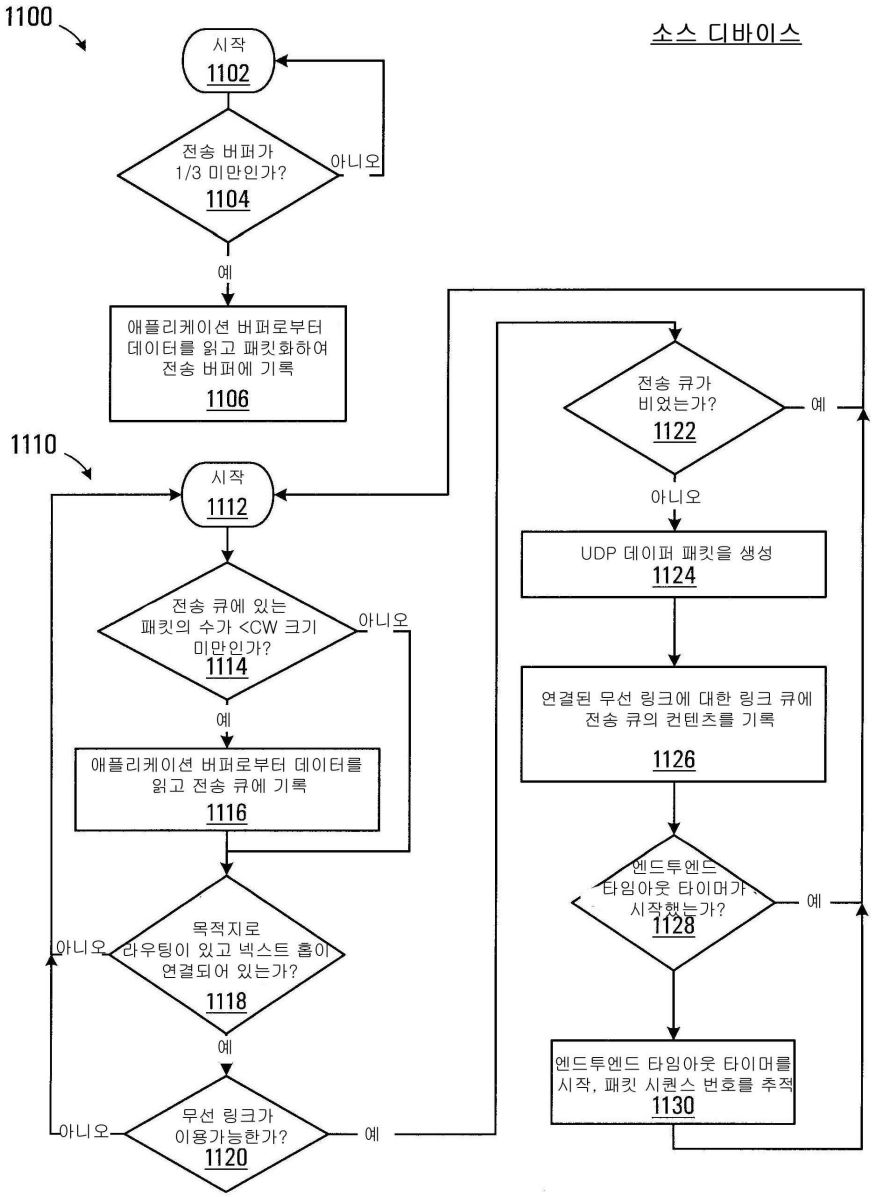
도면10a


1000
메시 서비스
도 6
620
peerChanged 이벤트가 메시 네트워크에 수신되는가?
1002
예
메시 포트를 결정, 애플리케이션에 peerChanged이벤트를 통지
1004
아니오
메시 네트워크를 통해 컨텐츠 데이터가 수신되는가?
1006
예
메시 포트를 결정
1008
컨텐츠 데이터를 애플리케이션에 전달
1012
아니오
메시 포트에 대응하는 애플리케이션이 실행되는가?
1010
예
아니오
메시 네트워크를 통해 컨텐츠 데이터를 포워딩
1014
애플리케이션이 암호화 키 교환을 요청하는가?
1016
예
아니오
서비스API에 키 교환 기능을 호출 콜의 결과를 획득
1018
애플리케이션이 네트워크 역할을 바꾸는가?
1020
예
아니오
애플리케이션이 Wi-Fi 또는 BT를 비활성화/활성화하는가?
1022
예
메시 네트워크 역할 또는 통신 셋팅를 업데이트
1024
아니오
애플리케이션이 메시서비스를 비활성화하는가?
1026
예
다른 애플리케이션들에 통지 모든 메시 포트를 해제, 메시 서비스를 셧다운
1028
아니오
도 10b
1026

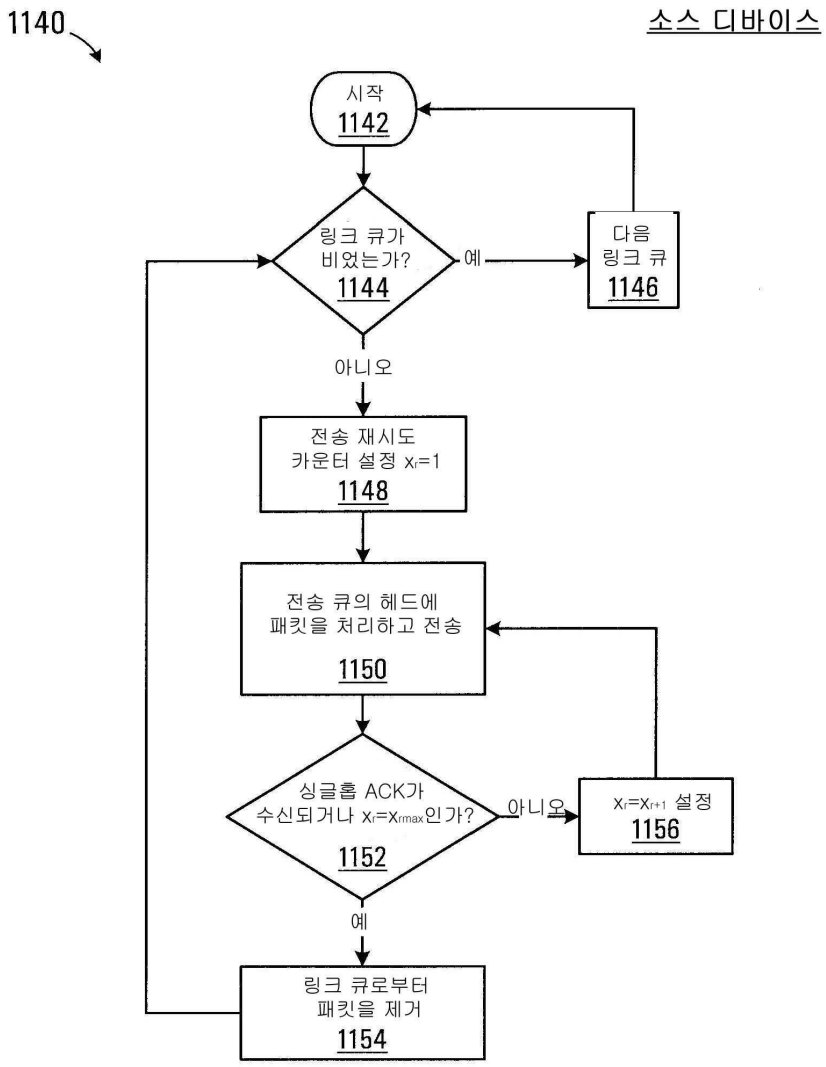
도면10b


1000
도10a
1040
애플리케이션이 상태정보를 요청했는가?
1030
예
상태 정보를 정의하는 구조화 데이터를 애플리케이션으로 전송
1032
아니오
메시 포트를 바인딩하도록 요청
1034
예
개발자 키 서명을 인증?
1036
예
바운드 포트 테이블을 업데이트, 통지를 전송
1038
아니오
통지 전송
1040
메시 포트를 언바인딩하도록 요청
1042
예
바운드 포트를 해제, 파운드 포트 테이블에 업데이트
1044
아니오
애플리케이션 버퍼에 추가 애플리케이션 통지:더 많은 데이터 청크를 수신할 준비
1056
메시 네트워크 상에 전송을 위해컨텐츠 데이터 청크가 수신되는가?
1046
예
다른 데이터 청크가 남아 있는가?
1048
예
버퍼에 공간이 있는가?
1052
예
아니오
아니오
애플리케이션 버퍼에 추가
1050
애플리케이션 버퍼에 추가, 애플리케이션 통지:더 많은 데이터 청크를 수신할 준비
1054
아니오
도 6
620

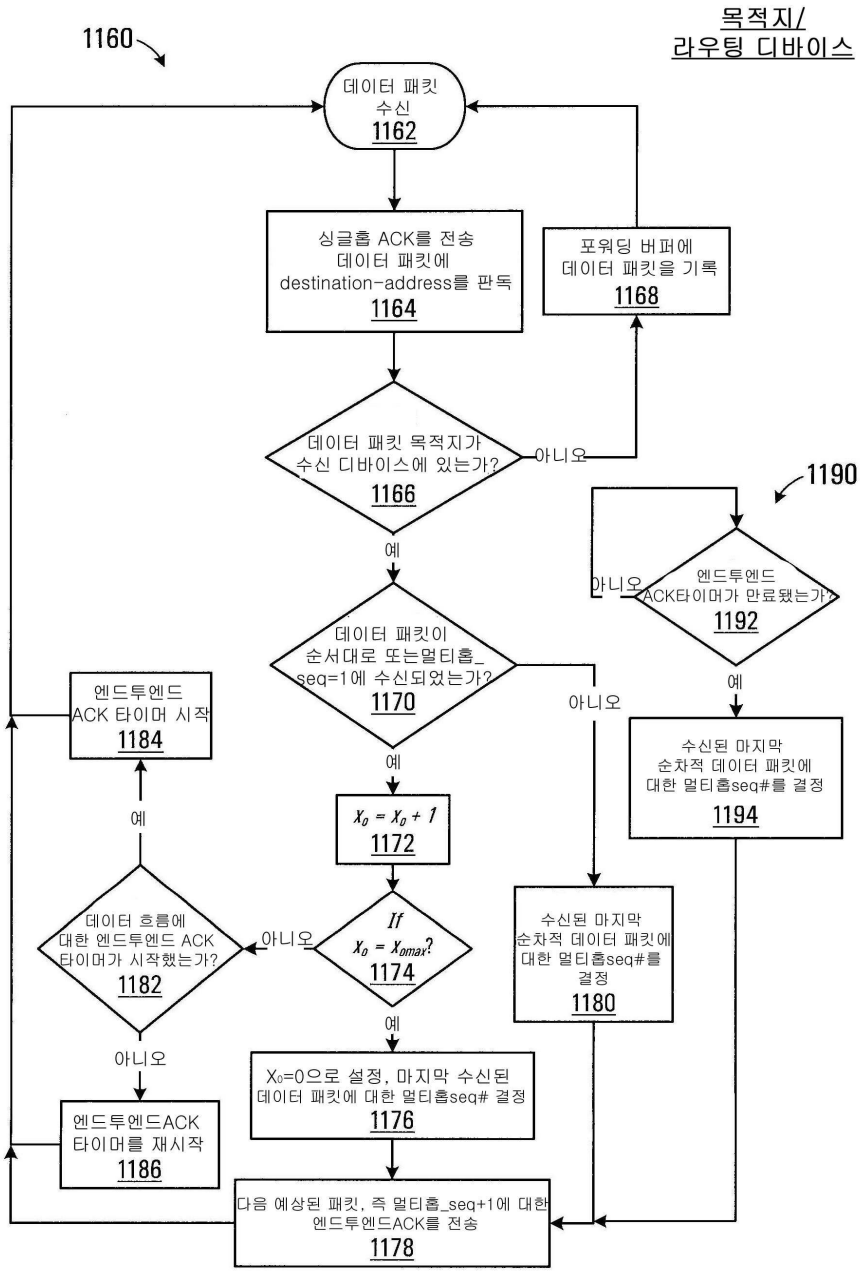
도면11a


1100
소스 디바이스
시작
1102
전송 버퍼가 1/3 미만인가?
1104
아니오
예
애플리케이션 버퍼로부터 데이터를 읽고 패킷화하여 전송 버퍼에 기록
1106
1110
시작
1112
전송 큐에 있는 패킷의 수가 <CW 크기 미만인가?
1114
아니오
예
애플리케이션 버퍼로부터 데이터를 읽고 전송 큐에 기록
1116
목적지로 라우팅이 있고 넥스트 홉이 연결되어 있는가?
1118
아니오
예
무선 링크가 이용가능한가?
1120
아니오
예
전송 큐가 비었는가?
1122
예
아니오
UDP 데이퍼 패킷을 생성
1124
연결된 무선 링크에 대한 링크 큐에 전송 큐의 컨텐츠를 기록
1126
엔드투엔드 타임아웃 타이머가 시작했는가?
1128
예
엔드투엔드 타임아웃 타이머를 시작, 패킷 시퀀스 번호를 추적
1130

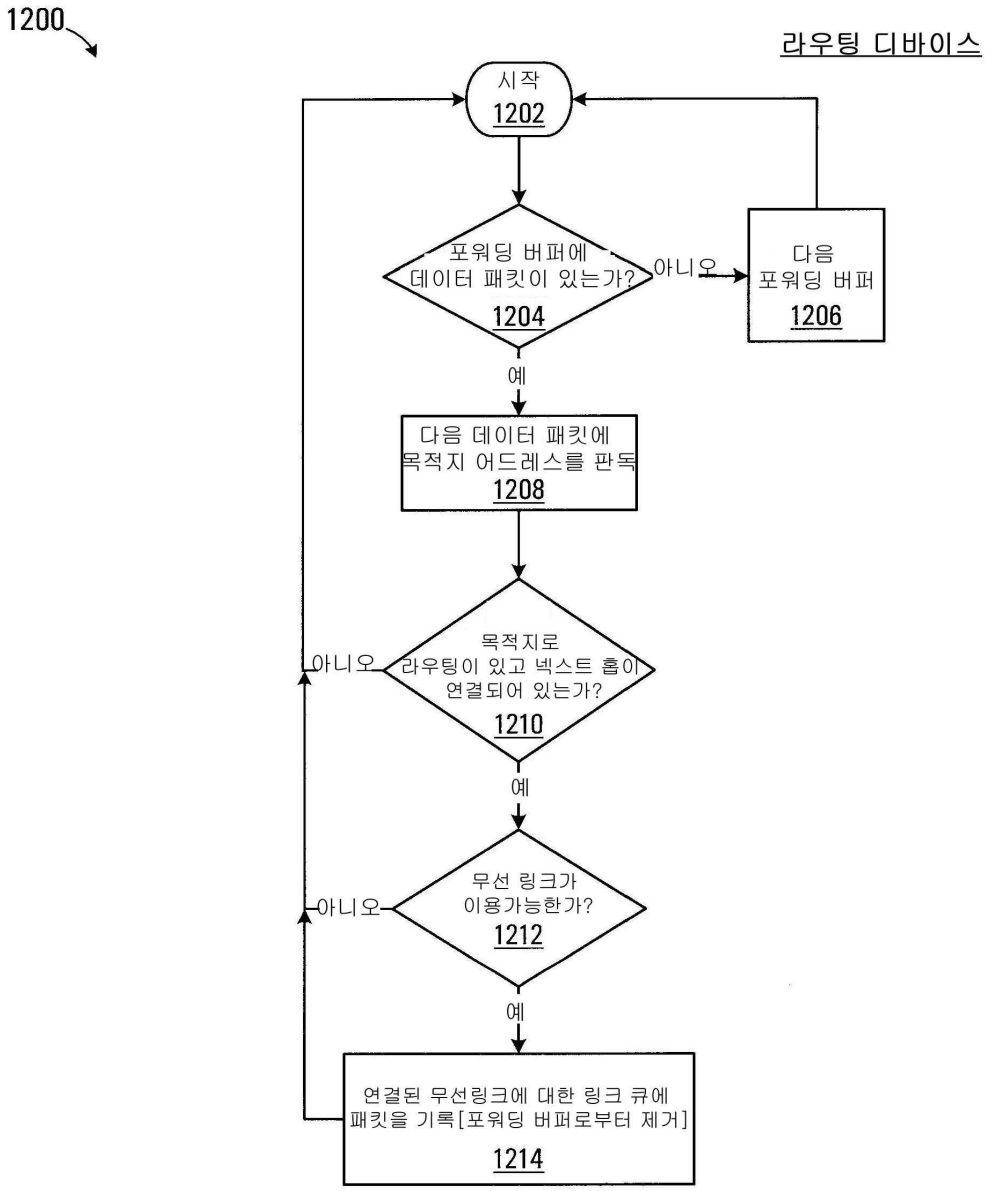
도면11b


1140
소스 디바이스
시작
1142
링크 큐가
비었는가?
1144
예
다음
링크 큐
1146
아니오
전송 재시도
카운터 설정 x_r=1
1148
전송 큐의 헤드에
패킷을 처리하고 전송
1150
싱글홉 ACK가
수신되거나 x_r=x_{rmax}인가?
1152
아니오
x_r=x_{r+1} 설정
1156
예
링크 큐로부터
패킷을 제거
1154

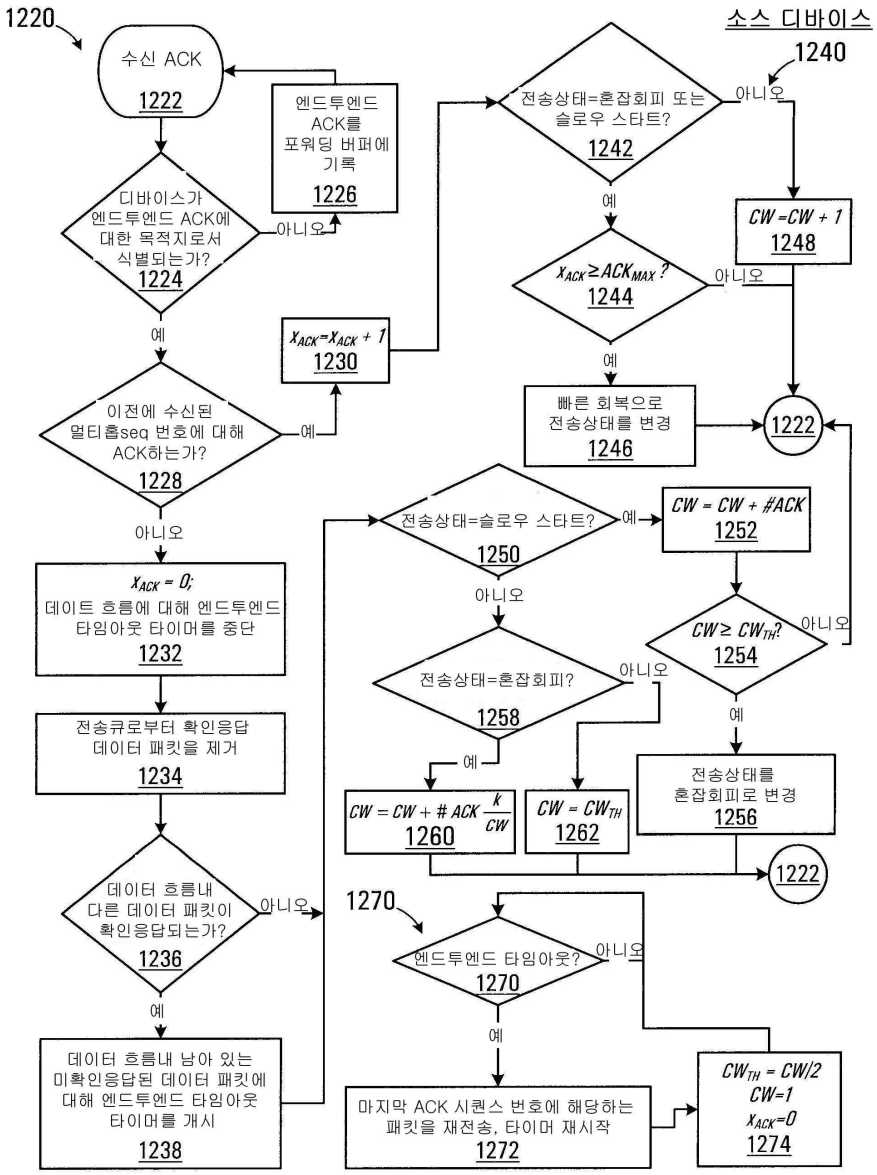
도면11c


1160
목적지/
라우팅 디바이스
데이터 패킷
수신
1162
싱글홉 ACK를 전송
데이터 패킷에
destination-address를 판독
1164
포워딩 버퍼에
데이터 패킷을 기록
1168
데이터 패킷 목적지가
수신 디바이스에 있는가?
1166
아니오
예
1190
엔드투엔드
ACK타이머가 만료됐는가?
1192
아니오
예
데이터 패킷이
순서대로 또는멀티홉_
seq=1에 수신되었는가?
1170
아니오
예
수신된 마지막
순차적 데이터 패킷에
대한 멀티홉seq#를 결정
1194
엔드투엔드
ACK 타이머 시작
1184
$X_0 = X_0 + 1$
1172
예
데이터 흐름에
대한 엔드투엔드 ACK
타이머가 시작했는가?
1182
아니오
If
$X_0 = X_{0max}$?
1174
아니오
수신된 마지막
순차적 데이터 패킷에
대한 멀티홉seq#를
결정
1180
예
아니오
X_0=0으로 설정, 마지막 수신된
데이터 패킷에 대한 멀티홉seq# 결정
1176
엔드투엔드ACK
타이머를 재시작
1186
다음 예상된 패킷, 즉 멀티홉_seq+1에 대한
엔드투엔드ACK를 전송
1178

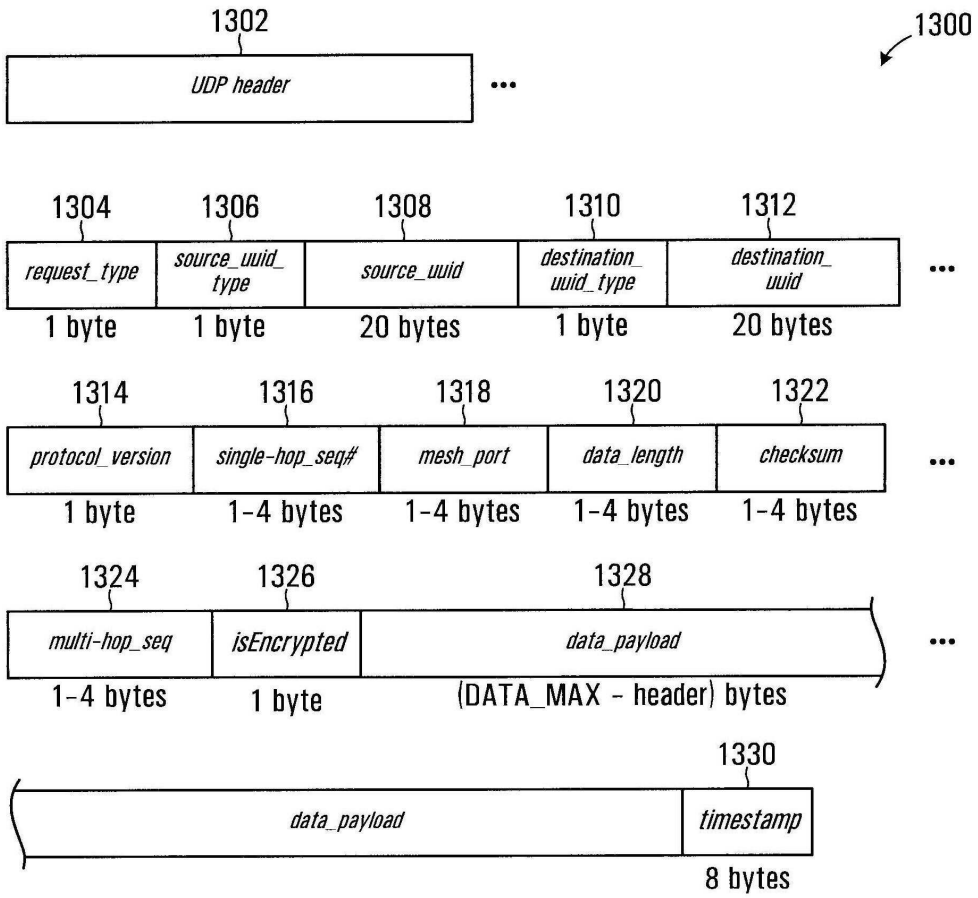
도면11d


1200
라우팅 디바이스
시작
1202
포워딩 버퍼에 데이터 패킷이 있는가?
1204
아니오
다음 포워딩 버퍼
1206
예
다음 데이터 패킷에 목적지 어드레스를 판독
1208
목적지로 라우팅이 있고 넥스트 홉이 연결되어 있는가?
1210
아니오
예
무선 링크가 이용가능한가?
1212
아니오
예
연결된 무선링크에 대한 링크 큐에 패킷을 기록[포워딩 버퍼로부터 제거]
1214

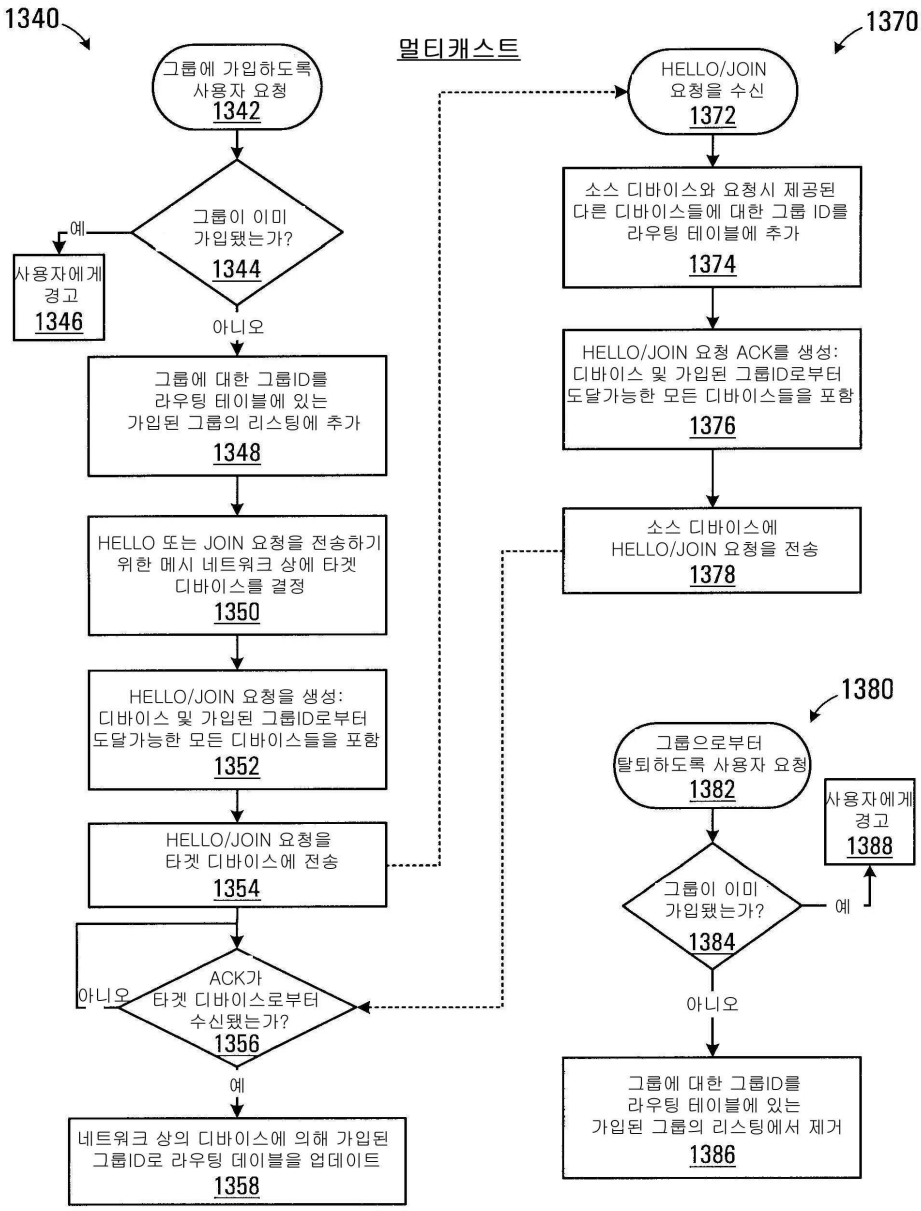
도면11e


1220
수신 ACK
1222
엔드투엔드 ACK를 포워딩 버퍼에 기록
1226
디바이스가 엔드투엔드 ACK에 대한 목적지로서 식별되는가?
1224
아니오
예
$x_{ACK}=x_{ACK}+1$
1230
이전에 수신된 멀티홉seq 번호에 대해 ACK하는가?
1228
예
아니오
$x_{ACK}=0;$
데이트 흐름에 대해 엔드투엔드 타임아웃 타이머를 중단
1232
전송큐로부터 확인응답 데이터 패킷을 제거
1234
데이터 흐름내 다른 데이터 패킷이 확인응답되는가?
1236
아니오
예
데이터 흐름내 남아 있는 미확인응답된 데이터 패킷에 대해 엔드투엔드 타임아웃 타이머를 개시
1238
소스 디바이스
1240
전송상태=혼잡회피 또는 슬로우 스타트?
1242
아니오
예
$x_{ACK} \geq ACK_{MAX}$?
1244
아니오
예
$CW=CW+1$
1248
빠른 회복으로 전송상태를 변경
1246
1222
전송상태=슬로우 스타트?
1250
예
아니오
$CW=CW+\#ACK$
1252
$CW \geq CW_{TH}$?
1254
아니오
예
전송상태=혼잡회피?
1258
아니오
예
$CW=CW+\#ACK\frac{k}{CW}$
1260
$CW=CW_{TH}$
1262
전송상태를 혼잡회피로 변경
1256
1222
1270
엔드투엔드 타임아웃?
1270
아니오
예
마지막 ACK 시퀀스 번호에 해당하는 패킷을 재전송, 타이머 재시작
1272
$CW_{TH}=CW/2$
$CW=1$
$x_{ACK}=0$
1274

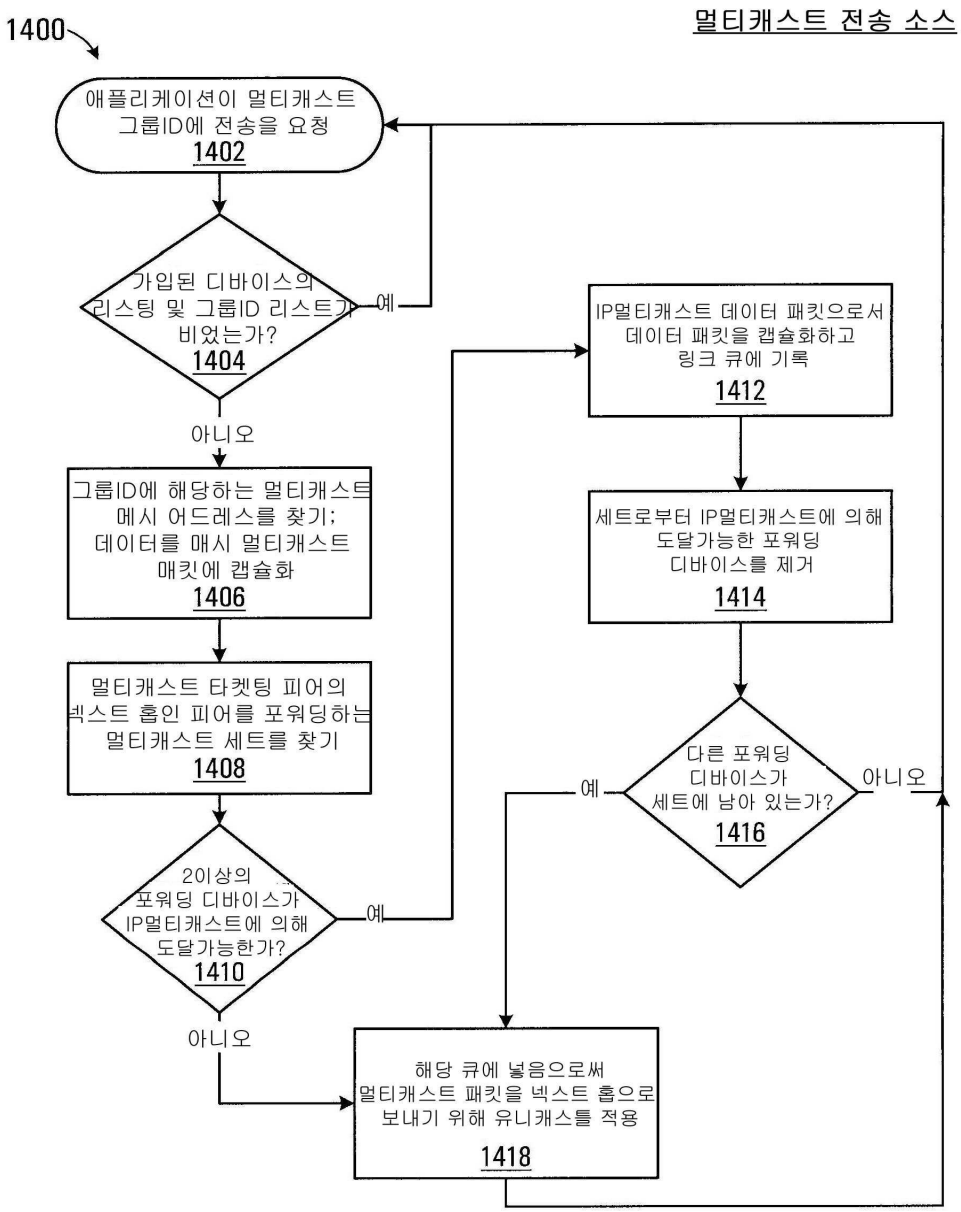
도면12


1300
1302
UDP header
1304
request_type
1 byte
1306
source_uuid_type
1 byte
1308
source_uuid
20 bytes
1310
destination_uuid_type
1 byte
1312
destination_uuid
20 bytes
1314
protocol_version
1 byte
1316
single-hop_seq#
1-4 bytes
1318
mesh_port
1-4 bytes
1320
data_length
1-4 bytes
1322
checksum
1-4 bytes
1324
multi-hop_seq
1-4 bytes
1326
isEncrypted
1 byte
1328
data_payload
(DATA_MAX - header) bytes
1330
data_payload
timestamp
8 bytes

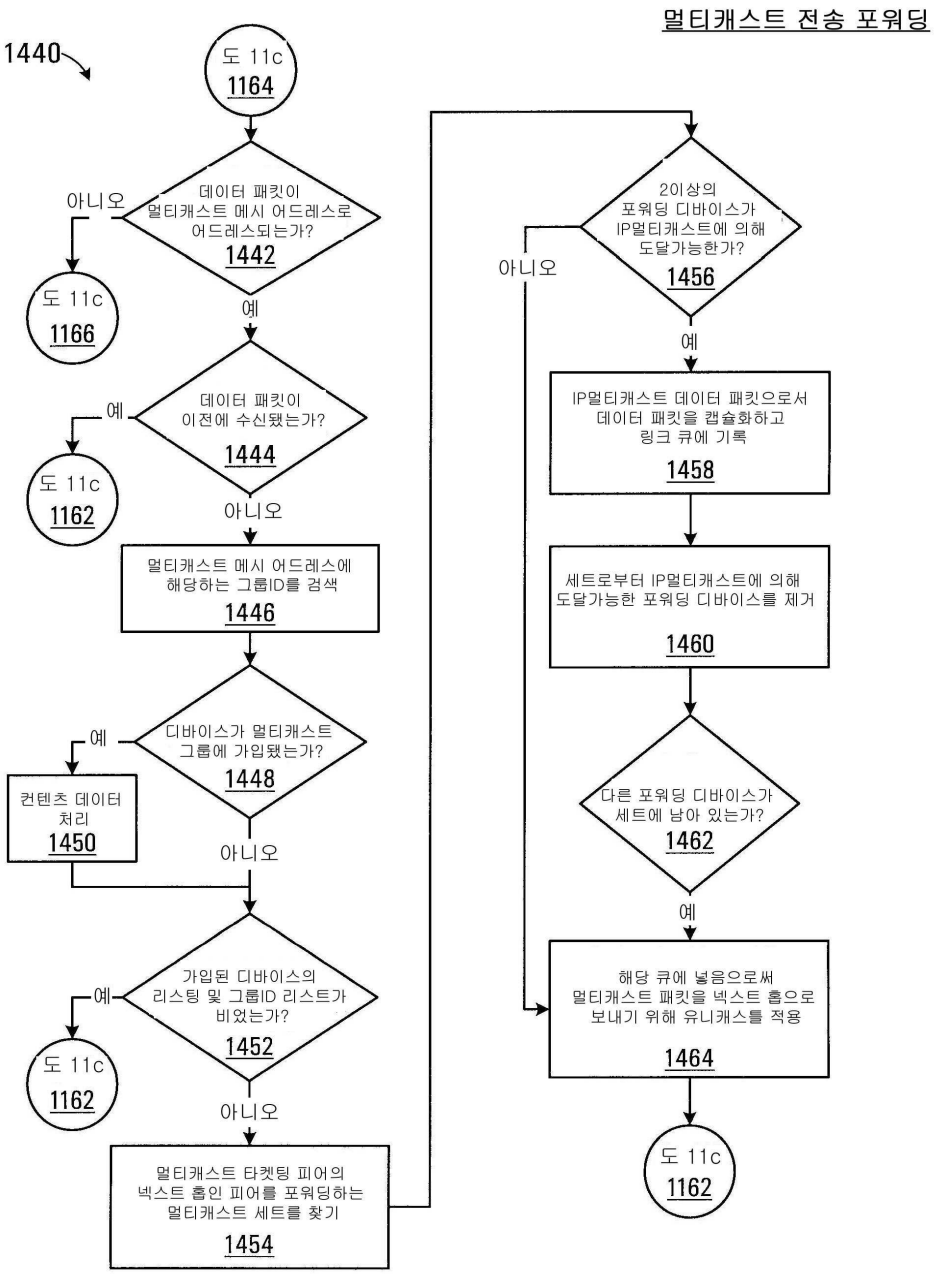
도면13a


1340
그룹에 가입하도록 사용자 요청
1342
그룹이 이미 가입됐는가?
1344
예
사용자에게 경고
1346
아니오
그룹에 대한 그룹ID를 라우팅 테이블에 있는 가입된 그룹의 리스팅에 추가
1348
HELLO 또는 JOIN 요청을 전송하기 위한 메시 네트워크 상에 타겟 디바이스를 결정
1350
HELLO/JOIN 요청을 생성: 디바이스 및 가입된 그룹ID로부터 도달가능한 모든 디바이스들을 포함
1352
HELLO/JOIN 요청을 타겟 디바이스에 전송
1354
ACK가 타겟 디바이스로부터 수신됐는가?
1356
아니오
예
네트워크 상의 디바이스에 의해 가입된 그룹ID로 라우팅 테이블을 업데이트
1358
멀티캐스트
1370
HELLO/JOIN 요청을 수신
1372
소스 디바이스와 요청시 제공된 다른 디바이스들에 대한 그룹 ID를 라우팅 테이블에 추가
1374
HELLO/JOIN 요청 ACK를 생성: 디바이스 및 가입된 그룹ID로부터 도달가능한 모든 디바이스들을 포함
1376
소스 디바이스에 HELLO/JOIN 요청을 전송
1378
1380
그룹으로부터 탈퇴하도록 사용자 요청
1382
사용자에게 경고
1388
그룹이 이미 가입됐는가?
1384
예
아니오
그룹에 대한 그룹ID를 라우팅 테이블에 있는 가입된 그룹의 리스팅에서 제거
1386

도면13b


멀티캐스트 전송 소스
1400
애플리케이션이 멀티캐스트 그룹ID에 전송을 요청
1402
가입된 디바이스의 리스팅 및 그룹ID 리스트가 비었는가?
1404
예
아니오
그룹ID에 해당하는 멀티캐스트 메시 어드레스를 찾기; 데이터를 매시 멀티캐스트 매킷에 캡슐화
1406
멀티캐스트 타켓팅 피어의 넥스트 홉인 피어를 포워딩하는 멀티캐스트 세트를 찾기
1408
2이상의 포워딩 디바이스가 IP멀티캐스트에 의해 도달가능한가?
1410
예
아니오
IP멀티캐스트 데이터 패킷으로서 데이터 패킷을 캡슐화하고 링크 큐에 기록
1412
세트로부터 IP멀티캐스트에 의해 도달가능한 포워딩 디바이스를 제거
1414
다른 포워딩 디바이스가 세트에 남아 있는가?
1416
예
아니오
해당 큐에 넣음으로써 멀티캐스트 패킷을 넥스트 홉으로 보내기 위해 유니캐스틀 적용
1418

도면13c


멀티캐스트 전송 포워딩
1440
도 11c
1164
데이터 패킷이 멀티캐스트 메시 어드레스로 어드레스되는가?
1442
아니오
예
도 11c
1166
데이터 패킷이 이전에 수신됐는가?
1444
예
아니오
도 11c
1162
멀티캐스트 메시 어드레스에 해당하는 그룹ID를 검색
1446
디바이스가 멀티캐스트 그룹에 가입됐는가?
1448
예
컨텐츠 데이터 처리
1450
아니오
가입된 디바이스의 리스팅 및 그룹ID 리스트가 비었는가?
1452
예
도 11c
1162
아니오
멀티캐스트 타켓팅 피어의 넥스트 홉인 피어를 포워딩하는 멀티캐스트 세트를 찾기
1454
2이상의 포워딩 디바이스가 IP멀티캐스트에 의해 도달가능한가?
1456
아니오
예
IP멀티캐스트 데이터 패킷으로서 데이터 패킷을 캡슐화하고 링크 큐에 기록
1458
세트로부터 IP멀티캐스트에 의해 도달가능한 포워딩 디바이스를 제거
1460
다른 포워딩 디바이스가 세트에 남아 있는가?
1462
예
해당 큐에 넣음으로써 멀티캐스트 패킷을 넥스트 홉으로 보내기 위해 유니캐스틀 적용
1464
도 11c
1162