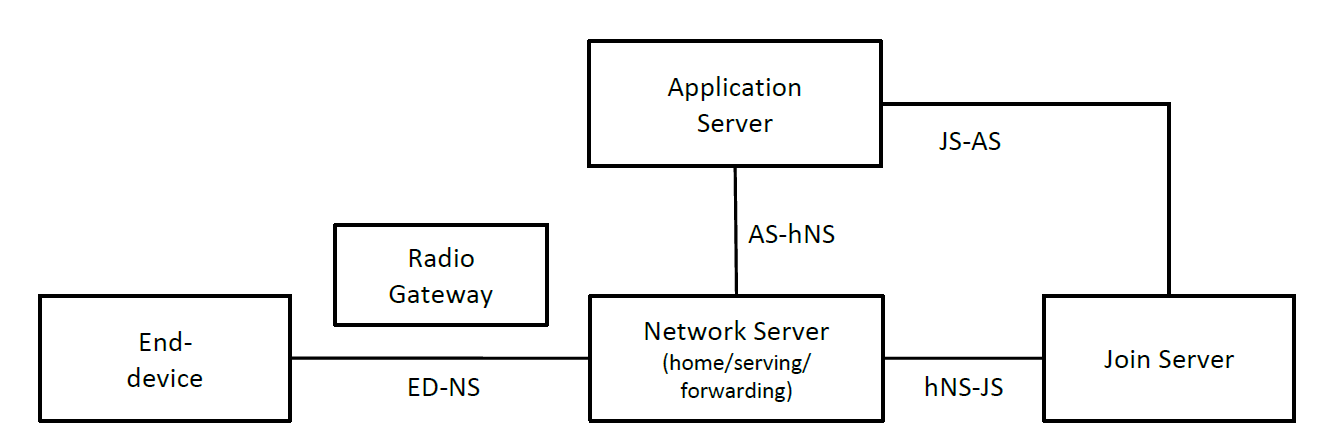
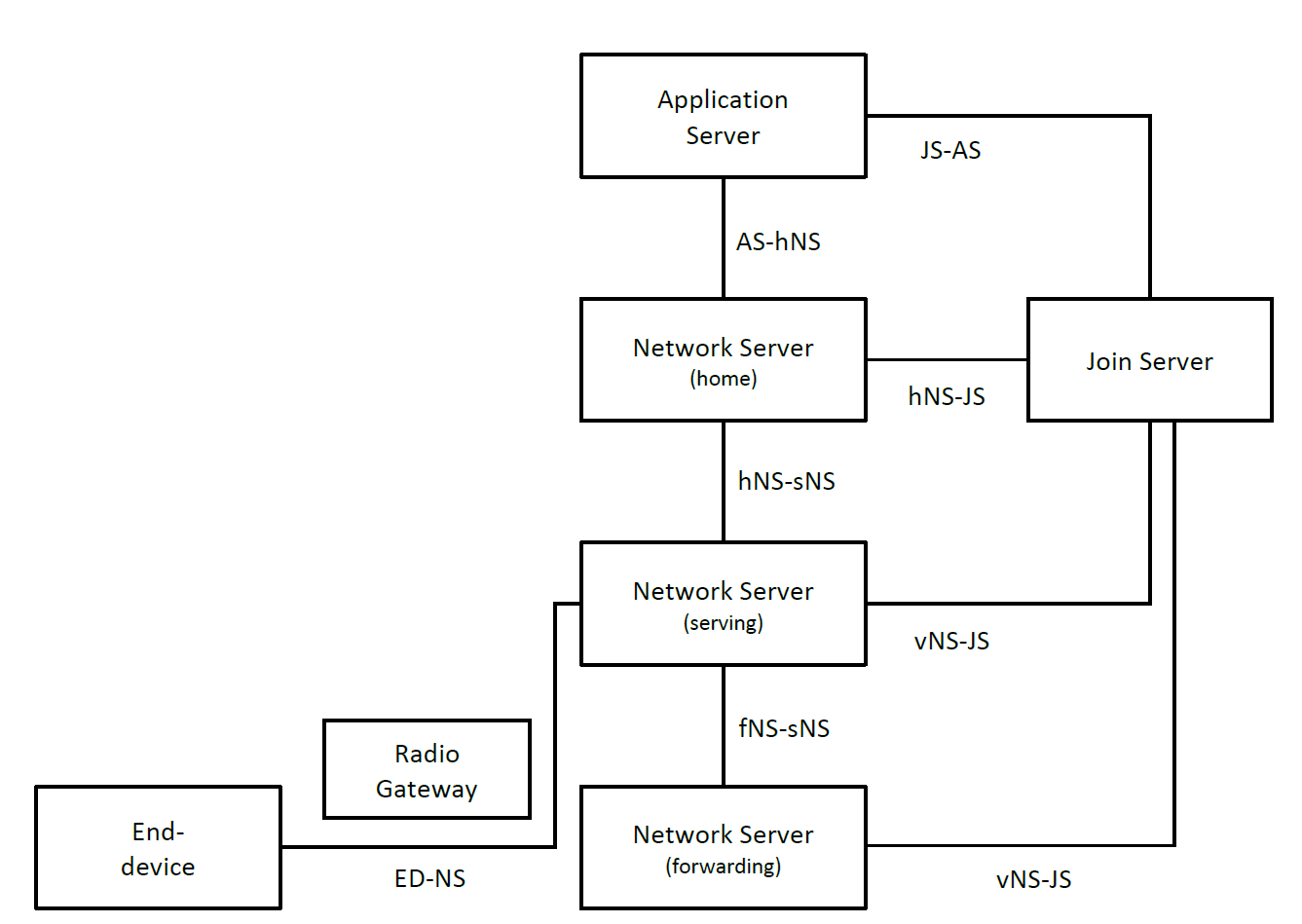
중요내용 정리

1. Network reference model

2가지의 NRM이 있다.



End device at home

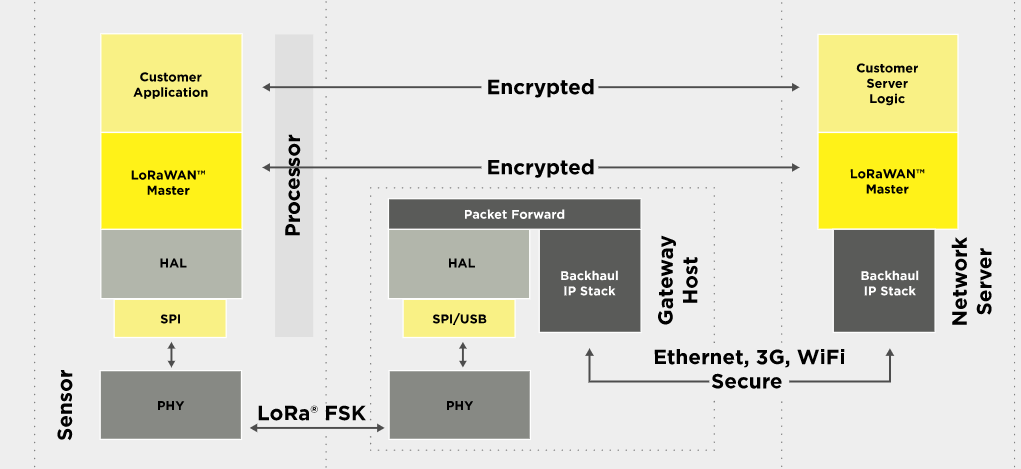


Roaming end device

이 2가지 모델로 구성되어있다.

**End 디바이스**

센서나 작동기 이고 무선으로 LoRa WAN 네트워크에 radio gateway를 통해 연결되어있다.

end 디바이스들의 application layer는 클라우드안의 특정한 application 서버와 연결되어 있다  


End 디바이스들의 모든 application layer의 payload는 그에 해당하는 application에 route되어있다.

**Radio gateway**

radio gateway는 모든 전송된 lora radio packet들을 네트워크 서버와 연결된 ip backbone망으로 전달한다.

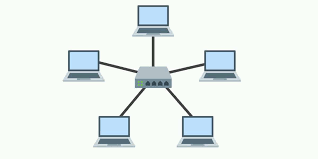
Radio gateway는 PHY층에서 작동한다.

목적: 1. uplink의 무선 packet을 해독하고 처리되지 않은 packet들을 네트워크 서버로 보내는 것이다.

2.downlink에서는 반대로 radio gateway는 간단히 network서버에서 오는 요청의 payload에 대해 다른 해석 없이 전송한다.

**Network 서버**

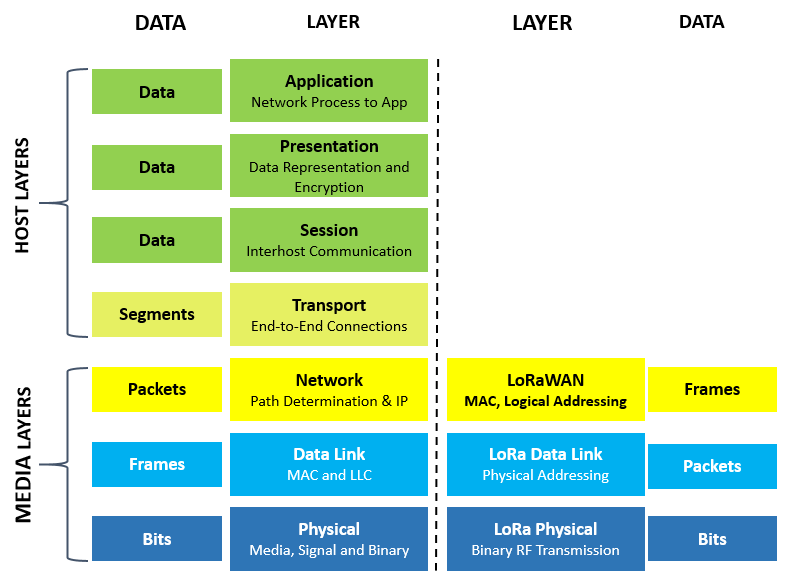
Lora WAN MAC layer를 end 디바이스가 network에 연결 되어있는 동안에 종료한다.

이것은 star topology의 중심이다. 

각 network 서버는 고유한 NSID를 통해 구별되고 하나 또는 다수의 NetID들로 구성될 수 있다.

Network서버들의 일반적인 특징들: 1 end 디바이스의 주소를 체크한다.

2 frame인증 그리고 frame 카운터 체크(OSI layer비교 사진)



3 ACK

4 data rate에 적응-> ADR이라고 Lora 신호에 따라 data rate를 변화시킴.

5 end 디바이스들에게 오는 MAC layer의 요구들을 응답하는 것

6 uplink application payload들을 적절한 application 서버로 전송하는 것

7 모든 application 서버에서 network로 연결된 모든 end 디바이스들로 전송되는 down link payload를 대기열에 집어넣는다.

8 join request 와 join accept 메시지를 end 디바이스와 join 서버에 전달.

로밍 아키텍쳐에서 네트워크 서버는 end 디바이스가 roaming 상태인지, 수반한 roaming의 형태에 따라 3가지 역할을 할지도 모른다.

**Serving network 서버는 end device의 MAC layer를 제어한다.**

**Home network 서버는 디바이스, 서비스, routing profile과 end 디바이스의 DevEUI를 저장하는 곳이다.(home 네트워크 서버는 join 과정에 사용될 join서버와 직접적으로 연관이 있다, application 서버와 연결 되어있다.)**

Home과 serving network가 분리되어있을 때 그들은 roaming 합의하에 있다.

Uplink와 downlink의 packet들은 serving과 home network 서버사이에서 전달 된다.

**Forwarding network 서버는 radio gateway를 관리하는 network 서버이다.**

(serving과 forwarding network가 분리되어있을 때 그들은 roaming 합의하에 있다.)

End 디바이스를 지원하는 하나이상의 forwarding network 서버가 있을 수 있다.

Uplink와 Downlink의 packet들은 forwarding 서버와 serving 서버사이에 전달 된다.

**Join 서버:**

**Join 서버는 OTA end 디바이스들의 활성 프로세스를 관리한다.**

(2 type)몇몇 join 서버들은 하나의 network 서버와 연결되거나 join 서버가 몇몇 network 서버들과 연결 되어있을지 모른다.

End 디바이스들은 join 서버가 join request메시지의 joinEUI를 통해 정보를 얻을 지도 모르는 신호를 보낸다.

**각각의 join 서버는 구별된다.** 유니크한 joinEUI에 의해서

Lora WAN 1.0/1.0.2[LW10, LW102] Join request의 메시지의 APPEUI는 변경되었다. LoraWAN 1.1[LW11]의 필드의 JoinEUI

Join 서버는 end 디바이스의 home 네트워크 서버 식별자를 알고 다른 network 서버의 정보를 로밍 과정에 의해 요구될 때 제공한다

join 서버는 uplink의 join request frame을 처리하고 downlink의 join accept frame을 생성하는 정보를 포함한다.

그리고 network와 application 세션키의 파생을 수행한다.

그리고 end 디바이스의 network 세션키를 network 서버로 그리고 application 세션키를 해당하는 application 서버로 전송한다.

Join 서버가 각각의 end 디바이스들을 컨트롤 하기위해서 가져야할 정보들은 다음을 포함할 것이다.

1. DevEUI
2. APPkey
3. NwkKey(오직 LoraWAN 1.1 end 디바이스에 적용가능하다)
4. Home network 서버 identifier
5. Application 서버 identifier
6. 여러 네트워크가 end 디바이스를 지원할 수 있는 경우 기본 네트워크가 선택할 수 있는 방법
7. End 디바이스의 Lora WAN network의 버전(LoraWAN 1.0, 1.0.2 or 1.1)

Root key인 Nwkkey와 appkey는 오직 join서버와 end 디바이스에서 사용이 가능하다.그리고 그들은 절대 보내지 않는다. network서버나 application 서버에

Root 키들인 Nwkkey그리고 appkey를 end 디바이스와 backend에 안전히 제공하고 저장하고 사용하는 것은 보안 해결책의 전반적으로 본질적이다

그러나 이 솔루션의 요소에는 SE(보안 요소) 및 HSM(하드웨어 보안 모듈)이 포함될 수 있습니다.

이러한 정보가 실제로 join 서버에 프로그래밍되는 방법은 본 문서의 범위를 벗어나고 **Join 서버마다 서로 다를 수 있습니다.**(예를 들어 웹 포털을 통하거나 API 집합을 통할 수 있습니다.)

Join 서버와 network서버는 End point 인증, 무결성 그리고 replay 보호 그리고 기밀성을 제공하면서 안전한 통신을 설정 할 수 있어야한다.

또한 Join 서버는 응용프로그램 세션 키를 응용프로그램 서버에 안전하게 제공할 수 있어야 합니다.

Join 서버는 여러 개의 application 서버에 연결되고 application 서버는 여러 개의 join 서버에 연결될 수 있습니다.

Join 서버와 application 서버는 엔드포인트 인증, 무결성, 재생 보호 및 기밀성을 제공하는 보안 통신을 설정할 수 있어야 합니다.

**Application 서버:.**

Application 서버는 연결된 End 디바이스의 모든 application 계층 payload를 처리하고 최종 사용자에게 application 수준 서비스를 제공합니다

또한 연결된 end 디바이스로 향하는 모든 application 계층 downlink payload도 생성합니다.

하나의 network서버에 여러 개의 application 서버가 연결되어 있을 수 있으며, 하나의 Application 서버가 여러 개의 Network 서버에 연결되어 있을 수 있습니다(예: 여러 개의 네트워크를 통해 최종 장치를 작동).

Application 서버를 여러 Join 서버에 연결할 수도 있습니다.

Home Network 서버는 DevEUI를 기반으로 uplink를 적절한 Application 서버로 라우팅합니다.  
앞서 언급한 네트워크 요소 외에도, LoRaWAN 아키텍처는 이러한 엔티티 사이에서 다음과 같은 네트워크 인터페이스를 정의합니다.

**Home network 서버-Join 서버**: 이 인터페이스는 Join 서버와 Network 서버 간의 join or 참가?(활성화) 프로시저를 지원하는 데 사용됩니다.

**v Network 서버-Join 서버**: 이 인터페이스는 로밍 활성화 절차에 사용됩니다. End-Device와 연결된 home Network 서버의 NSID 및 NetID를 검색하는 데 사용됩니다.

**End Device-Network 서버**: 이 인터페이스는 LoRa를 지원하는 데 사용됩니다.End-Device와 Network 서버 간에 WAN MAC 계층 신호 전달 및 페이로드 전달입니다.

**Application 서버 – home Network 서버**: 이 인터페이스는 애플리케이션 페이로드 및 Application 서버와 Network 서버 간의 관련 metadata 전송을 지원하는 데 사용됩니다.

**Home Network 서버 - serving Network 서버**: 이 인터페이스는 home Network 서버와 serving Network 서버 간의 로밍 신호 및 payload 전달을 지원하는 데 사용됩니다.

**Serving Network 서버 - forwarding Networking 서버**: 이 인터페이스는 serving network서버와 forwarding Network 서버 간의 로밍 신호 및 payload 전달을 지원하는 데 사용됩니다.

**Application 서버 - Join 서버**: 이 인터페이스는 응용프로그램 세션 키를 Join 서버에서 Application 서버로 전달하는 데 사용됩니다.

1. End 디바이스의 Type과 상태들.

Lora WAN end 디바이스에 2가지의 상태들이 있다.

**ABP activated end** 디바이스들, 그리고 **OTA activated end** 디바이스들이.

**ABP end 디바이스들**은 join 과정을 무시함(skip)으로써 직접적으로 특정한 네트워크에 묶여 있다.

**OTA end 디바이스**는 활성화하기 위해 선택한 네트워크에 Join 절차를 수행한다.

**ABP end 디바이스들**은 그것이 제조업체를 떠났을 때나 그 이후의 환경설정 정보들을 가질 것이다.   
:DevAddr, AppSkey, network 세션 key.

Network 세션키들은 SNWKSIntKey, FNwksIntKey 그리고 NwkSenckey이다. 이는 R1.1

그리고 NwkSkey는 R1.0/1.0.2의 end 디바이스에서 사용한다.

이러한 end 디바이스는 네트워크를 쉽게 사용할 수 있도록 이들의 home network 서버는DevAddr, network 세션키들을, end 디바이스의 application 서버의 정보를 가져야하고 Application 서버는 end 디바이스의 DevAddr, Appskey를 가져야한다.

**OTA end 디바이스들**은 그것이 제조업체를 떠났을 때나 그 이후의 환경설정 정보들 (DevEUI, NwkKey(R1.1에서만), AppKey, Join EUI)을 가져야한다.

이점에서 generic end 디바이스라고 불린다.

어떤 Network 서버 또는 Application 서버도 generic 디바이스가 commissioning 전까지는 generic 디바이스에 대한 정보를 가지고 있지 않을 수 있습니다.

하나의 end 디바이스에 대한 재구성은 그것의 라이프 사이클 동안 가능할지도 모른다.

Commissioning 과정은 End 디바이스와 그것의 home network서버와 특정한 application 서버와 함께 연관 되어있다.

OTA end 디바이스의 commission된 Join 서버는 end 디바이스에 대한 Home 네트워크 서버의 정보를 가질 것이다.

End 디바이스와 연관된 Application 서버는 End 디바이스의 DevEUI를 가질 것이다.

Home network 서버는 End 디바이스와 그것의 서비스의 광범위한 프로파일 정보를 가질 것이다.

commissioned OTA end 디바이스가 수행한 join과정이 성공적일 때 DevAddr, 네트워크 세션키들 그리고 AppsKey를 안다.

Join 서버는 DevEUI, DevAddr, 네트워크 세션키들, AppsKey, 그리고 Devnonce를 안다.

Join 서버는 DevEUI와 AppSkey를 Application 서버로 배달한다..

Join 서버는 Network 세션키들과 부가적으로 암호화된 AppsKey를 Network 서버에 배달한다.

1. Commissioning 프로세스

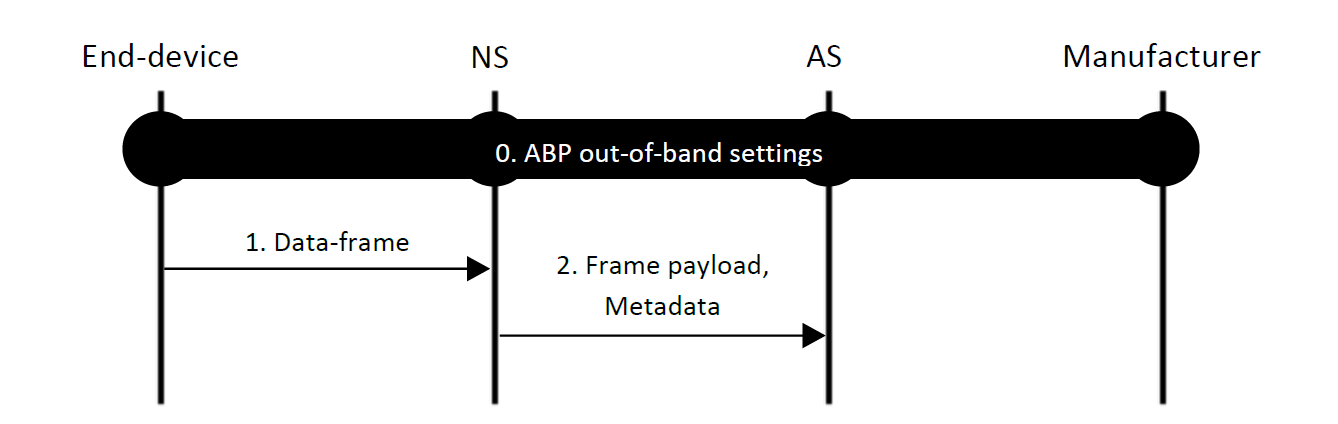
Commissioning 과정은 주어진 end 디바이스에 대해 Application 서버와 join 서버 (오직 OTA에 해당한다.) 그리고 Network 서버에서 실행되어졌다.

그것은 Join 서버가 end 디바이스와 home 네트워크를(오직 OTA에 해당한다.) 연관 시키는 것을, home network 서버와 application 서버가 end 디바이스와 end 디바이스에 해당 서비스 가입에 관한 프로파일 정보를 수신하는 것을 수반한다.

Decommissioning과정은 end 디바이스와 home 네트워크 서버, Application 서버 간의 관계를 부순다.

이 과정은 Application 서버와 Network 서버가 commissioning 할 때 만들어진 상태를 새로 설정하는 것을, join 서버의 end 디바이스와 home network 서버 간의 묶음을 푸는 것을(OTA에 적용가능) 수반한다.

1. ABP End 디바이스들의 Activation



이 과정은 R1.0[LW10,LW102[ 그리고 R1.1[LW11] 과정에서 적용 가능하다.

**단계 0**: End 디바이스, network 서버, 그리고 application 서버들은 필요한 정보들로 구성되므로, 그 end 디바이스가 전원이 켜지자마자 패킷을 보낼 수 있다.

**단계 1**: end 디바이스가 application payload를 보낼 때, 네트워크를 통한 어떠한 준비신호 없이 할 수 있다. Packet은 appSKey를 통한 암호화된 application payload, network 세션키 무결성 키들(SNwkSIntKey and FNwkSIntKey = R1.1에서의 end 디바이스, NwkSKey는 나머지)을 이용해서 MIC를 포함한다.

Network 서버가 packet을 받았을 때, 수신된 packet의 DevAddr에 기반하여 네트워크 세션 무결성 키를 조회를 수행해야 한다.

Network 서버는 회수된 키를 사용하여 MIC를 검증해야 합니다. 키를 찾을 수 없거나 MIC 검증에 실패하면 Network 서버는 패킷을 삭제해야 합니다.

**단계 2**: Network 서버는 승인된 패킷의 암호화된 payload를 End-Device와 연결된 Application 서버로 보내야 합니다. Application payload에는 DevAddr, FPort, timestemp등과 같은 metadata가 함께 제공될 수 있습니다. Network 서버는 End-Device에서 첫 번째 패킷을 수신하는 것을 End-Device에 대한 LoRa 세션 activation로 간주해야 합니다.

1. OTA end 디바이스들의 Activation

OTA Activation 과정은 network와 end 디바이스가 서로 인증하기위해 이용거나 uplink로 패킷을 보내거나 downlink의 패킷들을 받는 권한을 얻기위해 사용된다.

Network 서버들은 end 디바이스와 관련해 2가지의 방식으로 분류된다.

Home network 서버는 end 디바이스, 서비스, 그리고 end 디바이스의 routing profile을 붙잡고 activation이 끝난 후 Application 서버와 Join 서버의 상호작용을 하는 network 서버이다.

반면에 Visited network 서버는 end 디바이스가 제공받을 수 있도록 비즈니스 및 기술적인 동의(계약) home network 서버를 가지는 다른 network 서버이다.

Activation 과정에는 2가지의 변형이 있다, 다시 말해 Activation at home 과, Roaming Activation이

Activation at home: end 디바이스는 home network 서버의 무선 범위안에서 Activation 과정을 수행한다.

과정의 끝에서, home network 서버는 application 서버 및 join 서버에 연결할 수 있는 end 디바이스를 지원하는 유일한 서버이다.

Roaming Activation: end 디바이는 home network 서버의 무선 범위밖에서 있으나 visited network 서버안에서 Activation 과정을 수행합니다.

이 과정에서, visited network 서버는 join 서버의 도움으로 home network의 identify를 배우고 home network가 요구하는 end 디바이스와 서비스 profile을 얻는다.

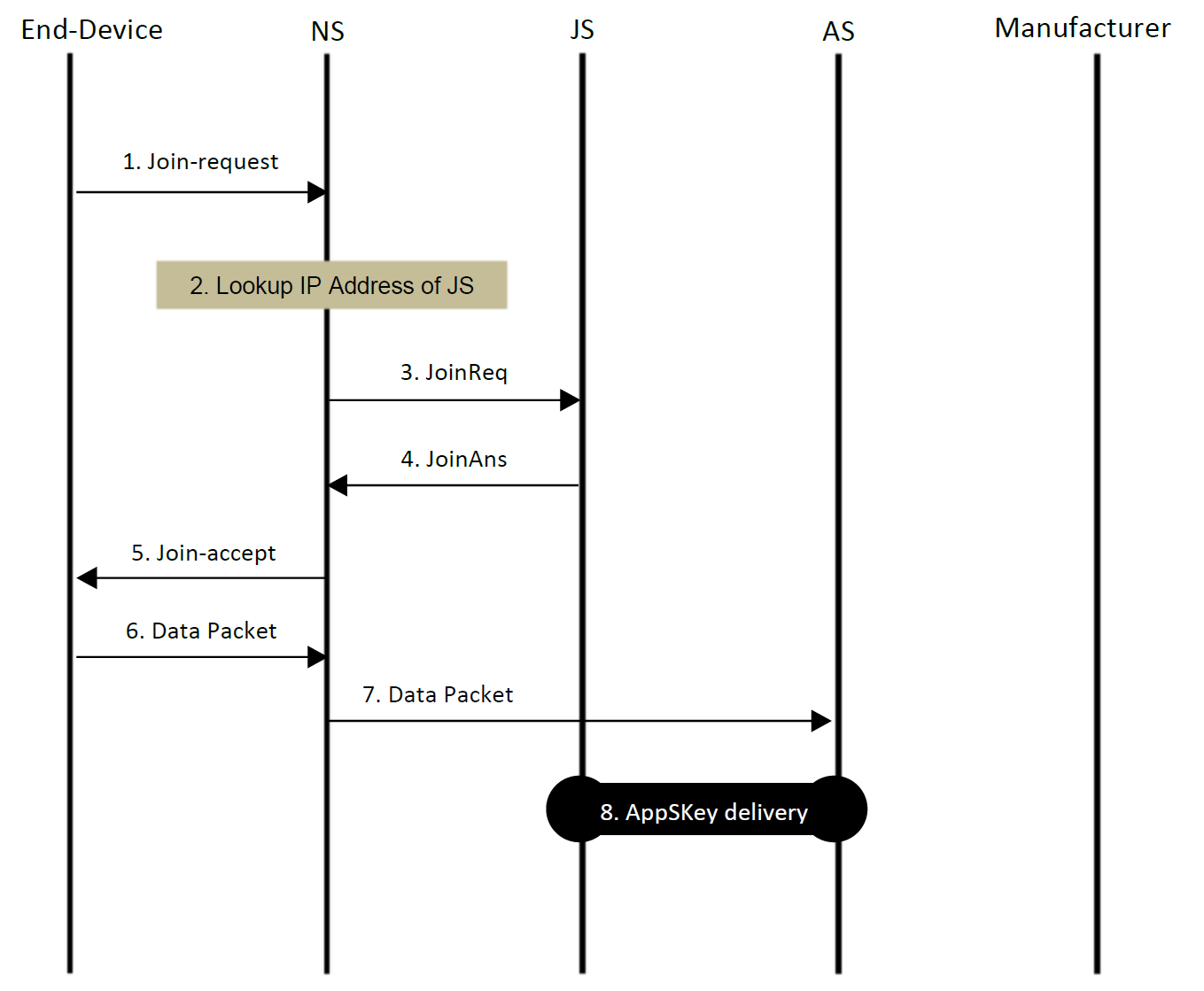
이 과정의 끝에서, end 디바이스는 서버는 application 서버 및 join 서버에 연결할 수 있는 Visited network 서버와 home network 서버에 의해 제공받는다.

End 디바이스가 성공적으로 join 이나 rejoin과정을 수행했을 때, end 디바이스는 backend와 Lora 세션을 가졌다고 여겨진다.

각각의 Lora 세션은 end 디바이스, 그리고 network 서버, join 서버 그리고 application 서버 에 의해 관리되는 문맥의 parameter들의 집합과 연관되어있다.( session keys, DevAddr, network 서버 ID, etc)

Lora 세션은 end 디바이스가 DeActiovation(Exit)과정 또는 다른 성공적인 join/rejoin과정을 수행할 때 종료한다.

1. OTA Activation at home 절차



R1.0[LW10,LW102]와 R1.1[LW11]의 end 디바이스에서 적용가능하다.

**단계 1**: end 디바이스들은 join request 메시지를 전송할 것입니다.

**단계 2**: network 서버가 join request메시지를 받았을 때, network 서버는 DevEUI로 식별된 end 디바이스를 위한 home network 서버인지 아닌지를 결정할 것입니다.

이 흐름에서 network서버가 end 디바이스의 home network 서버인지 추측 가능하다.

network 서버가 end 디바이스의 home network 서버가 아니지만 network 서버가 roaming activation 절차에 join 서버를 사용하도록 구성한 경우에는 Section 12를 보시오.

만약 network 서버가 end 디바이스의 home network 또는 join 서버로 구성되지 않으면, network 서버는 아무 말 없이 join request를 무시하고 여기서 절차를 종료할 것이다.

만약 network 서버가 대역 외의 메커니즘에 의해 ip 주소/호스트이름으로 구성되지 않은 경우에는 Network 서버가 join-request 메시지에 기반하여 join 서버의 ip 주소를 조회하기위해 DNS를 사용할 것이다.(section 20에 자세한 내용).

만약 DNS 조회가 실패하면, network 서버는 여기서 절차를 종료할 것이다.

R1.0[LW10]의 AppEUI로 구성된 end 디바이스는 join 서버를 식별하지 않으면, network 서버는 대역 외의 메커니즘에 의해 join 서버의 ip 주소/호스트 이름을 가지고 구별해야한다.

**단계 3:** network 서버는 join request메시지, MAC 버전, DevEUI, DevAddr, DLSettings, RXDelay, 그리고 부가적인 CFList의 PHYpayload를 나르면서 join 서버로 join request 메시지를 보낸다.

Network 서버는 network 서버와 end 디바이스 사이의 가장 높은 흔한 버전으로 MAC 버전을 맞출 것이다.

**단계 4:** join 서버는 MAC 버전에 따라 join request 메시지를 처리하고 result = Success 인 경우에 join ans를 network 서버에 보내야 한다. join-accept메시지, network 세션키들{(SNwkSIntKey, FNwkSIntkey 그리고 NwkSEncKey //R1.1에서는), (NwkSKey //R1.0/R.1.0.2의 end 디바이스에서)}이 포한된 PHYpayload는 AppsKey 또는 SessionKeyID 또는 둘 가지로 암호화 되었다.

그리고 Result = UnknownDevEUI인 경우는 end 디바이스는 join 서버에 의해 인식되지않았다,

Result = MICFailed 인 경우에는 join request의 MIC가 검증에 실패했다.

Result = FrameReplayed 인 경우는 DevNonce가 이전에 쓰인 것 이고, Result = JoinReqFailed 인 경우는 다른 에러 케이스이다.

Join 서버는 만들어진 session key들과 관련된 SessionKeyID를 만들 수 있다.

SNwkSIntKey, FNwkSIntKey, NwkSEncKey, 그리고 AppSKey는 end 디바이스가 R1.1일 때 LoraWAN 1.1 설명서[LW11]에 기반하여 생성될 수 있다.

NwkSKey는 end 디바이스 R1.0/R1.0.2일 때 LoraWAN 1.0 설명서[LW10]에 기반하여 생성될 수 있다.

**join 서버에서 Network 서버로 전달 될 때 AppSkey는 join 서버와 Application 서버 간에 공유 키로 사용되는 동안 암호화 되어졌었다.**

R1.0[LW10] end 디바이스들에서, AppEUI가 join 서버에 알려지지 않을 때 역시 join 서버는 join request의 메시지들의 처리할 것이다.

**단계 5:** network 서버는 PHYpayload를 전달한다. 만약 수신된 join Ans 메시지가 성공을 가르킨다면 end 디바이스의 join accept 메시지와 함께

end 디바이스는 join accept 메시지를 받았을 때 LoraWAN spec[LW10, LW102, LW11]에 기반해서 network 세션키들과 Appskey를 생성할 것이다.

**단계 6:** network 서버가 end 디바이스로부터 uplink packet을 받았을 때 network 서버는 DevEUI, 그리고 암호화된 AppSkey 또는 SeessionKeyID 또는 둘 다 Application 서버의 payload와 함께 보낼 것이다.

**단계 7:**  Application 서버가 application payload와 함께 임호화된 AppSkey를 받았을 때, application 서버는 join 서버와 Appication 서버와 공유된 비밀키를 사용해서 AppSkey를 복호화하고 AppSkey를 전달 받은 payload의 복호화에 사용할 것이다.

만약 복호화된 AppsKey가 network 서버에 의해 사용할 수 없다면 Application 서버는 다음 단계를 진행할 것이다.

**단계 8:** 이 단계는 Application 서버가 join 서버로부터 직접적으로 AppSkey를 받기를 원하는 경우에 일어난다.

AppSKeyrequest메시지를 보냄으로써 application 서버는 end 디바이스의 DevEUI에 의해 명확해진 AppSKey, 그리고 join 서버에 SessionKetID를 요구할 것이다.

AppSkey은 join 서버와 application 서버사이에 공유된 암호를 통해 암호화 된다.

Join 서버는 join request메시지를 통해 암호화 된 AppSKey, DevEUI 그리고 SessionKeyID를 Application 서버로 전달한다.

그러고나서 application 서버는 암호화된 AppSKey를 join 서버와 Application서버와 공유된 비밀키를 통해 복호화 할 것이다.

그러고 나서 Application 서버는 AppSKey를 사용해서 application payload를 암호화와 복호화를 시작할 것이다.

End 디바이스의 commissioned OTA Activation은 network 서버와 join 서버가 동일한 관리자도메인에 속해있거나 2개의 다른 분리된 관리자 도메인에 속하거나 두가지 모두 일어날 수 있다.

1. Deactivation of OTA end 디바이스

OTA-activation end 디바이스의 Lora 세션은 다양한 이유들 사용자가 contract 종료에 도달하거나 악의적인 end 디바이스의 동작 등으로 종료될 수 있다.

세션을 비활성화 하는 데 사용되는 과정은 join과정의 반대 부분인 종료절차이다.

종료 과정을 수행하기 위한 Lora WAN 신호는 명시적이거나 전용적이지 않다.

End 디바이스와 벡엔드는 이 과정을 수행하기 위해 application layer의 신호에 의존한다고 추측된다.

Application layer의 신호의 trigger와 세부사항들은 이 범위를 벗어난다.

application 서버가 Home network 서버에게 종료과정에 대해 알리고 분리된 serving network가 있을 때

home network 서버는 LoraWAN 세션의 종료를 serving network 서버로 전달하기위해 Handover Roaming Stop Procrdure을 수행할 것이다.

성공적으로. 새로운 join/rejoin 과정을 수행하는 end 디바이스들은 또한 현재의 LoraWAN 세션의 종료하며 어떤 면에서 이는 그 세션과 관련해서 비활성화로 여겨진다.

1. Security Associations

테이블이(가) 표시된 사진

자동 생성된 설명

1. Roaming 절자

**Roaming의 type:**

Lora roaming 2가지 형태가 있다, 즉(namely) Passive Roaming 과 Handover Roaming.

Passive Roaming은 end 디바이스가 network 서버의 LoraWAN 서비스로부터 이득을 얻는 것을 가능하게 한다. 심지어 해당 end 디바이스에 대해 두 network의 RF 능력(채널들)의 중복 제한안에서 end 디바이스가 다른 network 서버의 제어하에 gateway(들)를 사용하고 있을 때조차도

end 디바이스의 Lora 세션과 MAC layer의 제어는 serving network 서버라고 불리는 이전의 network 서버에 의해 유지된다. 반면에 무선으로 오거나 가는 frame은 forwarding network 서버라고 불리는 나중의 network 서버에 의해 다뤄진다.

주어진 Lora 세션에 대해서 오직 하나의 serving network 서버만 있을 수 있으며 반면에 0개 또는 다수의 forwarding network 서버들은 같은 세션에 참가 할 수 있다.

2가지 형태의 forwarding network 서버들이 있다: stateful과 stateless.

Stateful forwarding network 서버는 end 디바이스의 Passive roaming의 시작(onset)에 대한맥락(절차?)을 만들고 같은 end 디바이스의 uplink/downlink의 packet들의 차후 절차를 위해 해당 맥락(절차?)을 활용한다.

Stateless forwarding network 서버는 어떠한 그런 맥락을 만들지 않는다. 그러므로 결국 어떠한 uplink/downlink의 packet들을 서로 독립적으로 처리해야한다.

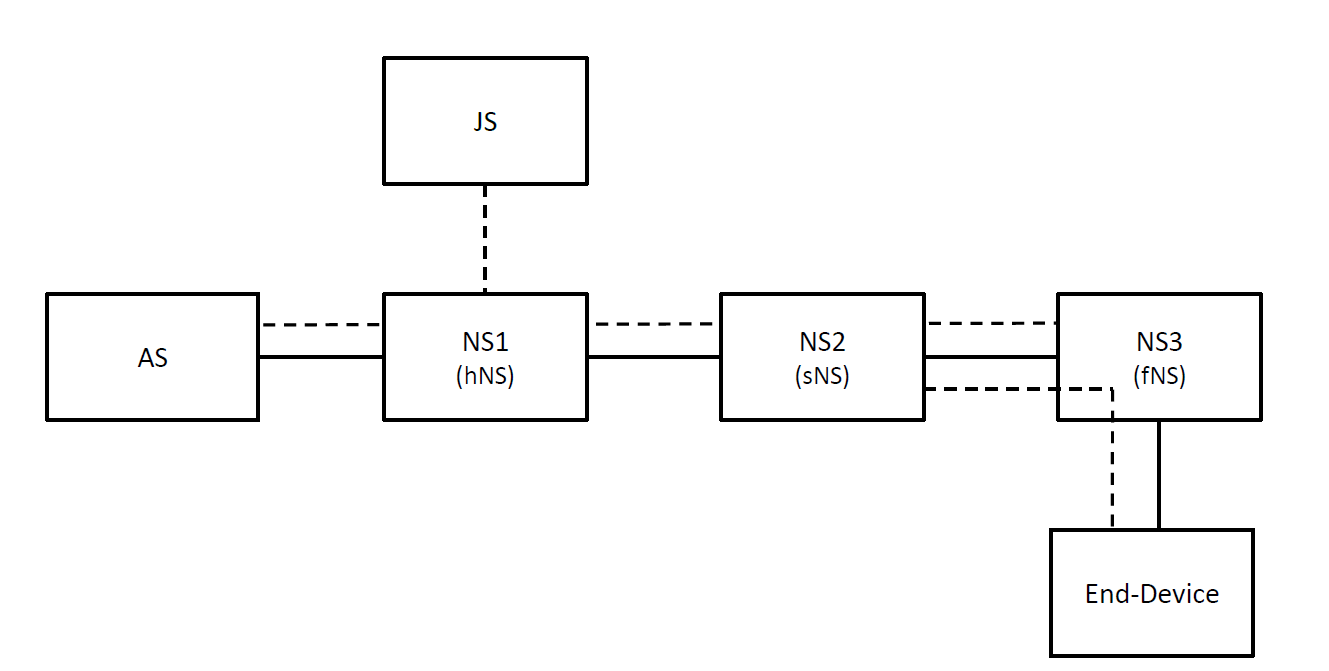
그것은 주어진 forwarding network 서버가 stateless인지 몇몇의 범위 밖의 메커니즘에 의해 roaming 파트너에게 stateful이라고 알려졌는지 추측 가능하다.

Handover Roaming은 하나의 network서버로부터 다른 서버로 MAC layer의 제어의 전달이 가능하다.

Home network 서버는 join 서버와 application서버와 함께 control plane과 data plane 유지하고 심지어 end 디바이스가 하나에 network 서버에서 다른 서버로 가는 Handover Roaming을 진행한 이후에도 유지한다.

Home network 서버는 end 디바이스가 다음 join 절차를 수행할때까지 주어진 lora 세션을 동일하게 유지한다.

Forwarding network 서버와 달리, serving network 서버는 end 디바이스이 RF 설정들을 제어할 수 있는 능력을 가진다. 이는 조금 더 유연한 roaming 시나리오를 허락한다.



이 그림은 handover와 passive roaming 둘다 보여준다.

이 예제에서 end 디바이스는 home network 서버로 일하는 network 서버 1을 통해 활성화 된다. 이어서 end 디바이스는 network 서버 2가 serving network 서버가 될 때 network 서버1로부터 network 서버 2로 handover Roaming을 수행하고 또한 network 서버3이 end 디바이스의 forwarding network 서버가 되었을 때 network 서버 2로부터 network 서버 3로 passive roaming을 수행한다.

Roaming 활성화는 visited network 서버의 커버리지에서 end 디바이스를 활성화 하기위한 능력이다.

1. LoRa 세션으로 가는 동안의 Passive roaming
2. LoRa 세션으로 가는 동안의 Hanover roaming
3. Home network 서버와 visited network 서버 간의 handover romaing에 기반한 새로운 lora 세션의 roaming 활성화
4. Home network 서버와 visited network 서버 간의 passive roaming에 기반한 새로운 lora 세션의 roaming 활성화

Home network 서버와 visited network 서버가 어떠한 roaming에 합의가 없을 때의 새로운 lora 세션의 활성화는 문서의 범위에 벗어난다.

이는 2개의 network 서버들이 세번째 network서버와 roaming 합의를 가질지도 모르는 경우를 포함한다.

(예를 들어 home network 서버 그리고 3rd network 서버는 handover roaming 동의를 가지고, 그리고 3rd network 서버 그리고 visited network 서버는 passive roaming 동의를 가진다.

**Roaming policy:**

각각의 network operator는 그들의 NetIDs에 의해 구분되는 다른 network operator들과 passive roaming, handover roaming, 활성화에 기반한 passive roaming, 활성화에 기반한 handover roaming을 각각 allow/disallow 할 수 있는 roaming 정책으로 설정될 것이다

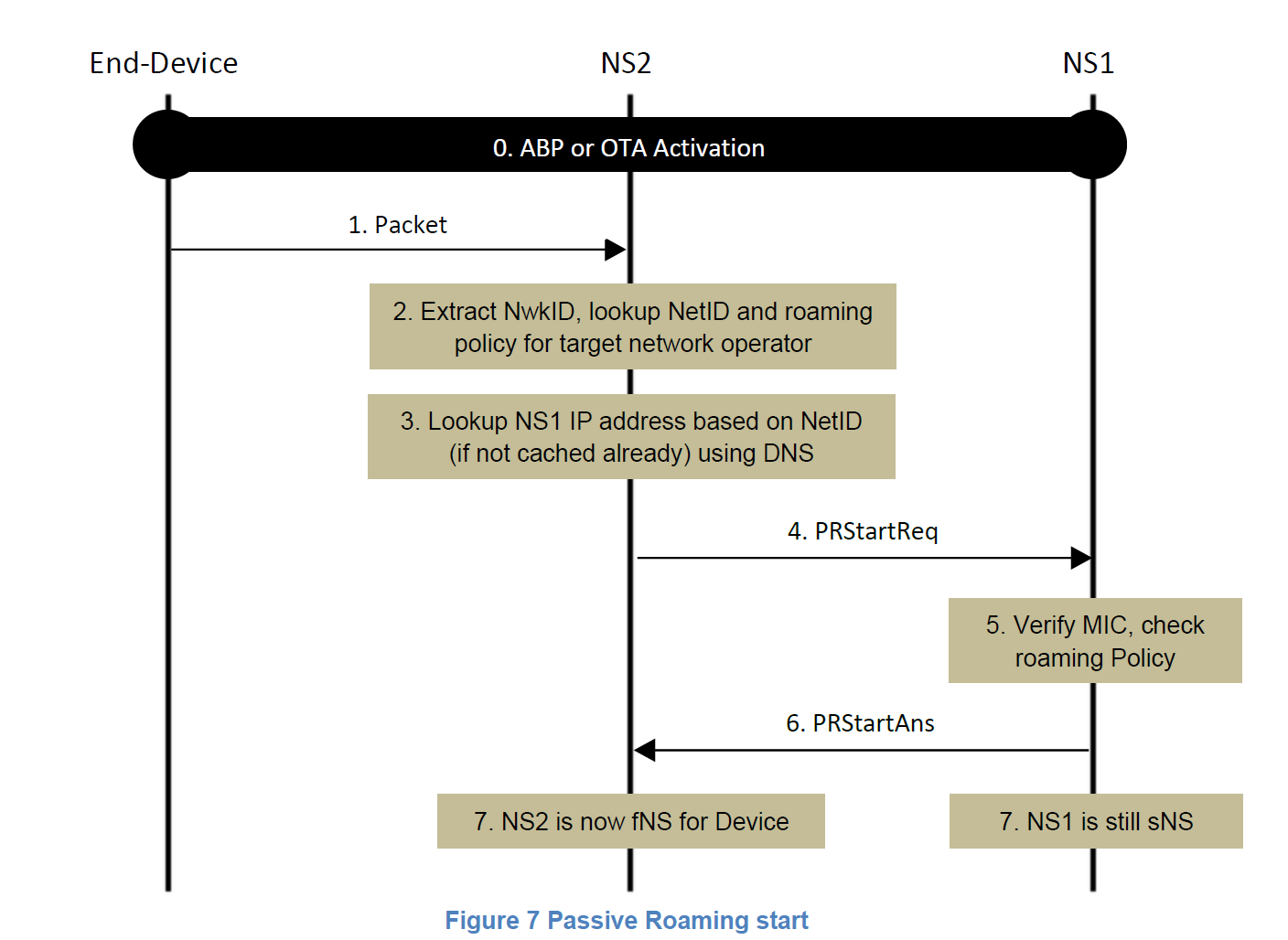
Passive roaming 에서, 정책은 forward network 서버에 의해 uplink MIC가 체크되든지 아니던지 할 것이다.

각각의 network operator는 DevEUI에 의해 구별되는 개별 end 디바이스들의 passive roaming, handover roaming, 활성화에 기반한 passive roaming, 활성화에 기반을 한 handover allow/disallow할 수 있는 roaming 정책으로 설정할 것이다.

**Passive roaming**

R1.0과 R1.1의 end 디바이스들과 네트워크들에 적용된다.

**Passive Roaming Start**



이 그림은 두 network 서버사이의 end 디바이스의 Lora 세션으로 가는 Passive Roaming 절차의 흐름을 설명한다.

**Passive Roaming based Activation of new LoRa Session**은 구역 12.2를 참조해라

**단계 0:** end 디바이스는 network 서버 1을 통해 활성화가 된다.

**단계 1:** end 디바이스가 packet을 보냈을 때, 그것은 end 디바이스에 대해 연관된 문맥을 가지고 있지 않은 network 서버2에 의해 수신된다.

**단계 2:** 만약 network 서버 2가 다른 network 서버 operator들과 passive roaming을 가능하게 구성했다면, network 서버 2는 passive roaming에 동의했던 operator들의 NetID(S)과 받은 packet으로 NwkID를 매핑을 시도할 것이다.

**(여기서 packet을 network 서버에 주는 이유는 gateway 단에서 packet으로 전송을 하기 때문이다.)**

만약 찾은 게 없다면, network 서버2는 packet을 버리고 여기서 절차를 멈출 것이다.

**단계 3:** 만약 하나 또는 다수의 NetIDs이 발견 됐다면, network server2는 각각의 NetID와 연결되어진 network 서버의 ip 주소를 보기위해 DNS를 사용할 것이다/ 만약 Network 서버2가 외부의 메커니즘에 의해 network 서버의 IP address/hostname을 미리 구성하지 않았다면

만약 하나이상 맞는다면, step 4-6은 실행된다. 각각의 일치하는 network 서버에서

**단계 4:** 만약 network 서버2가 이 packet에 대해 중복제거를 수행했다면 incoming packet과 연관된 ULMetadata 그리고 DedupWindowSize 와 함께 PHYPayload를 가지는 PRStartReq 메시지를 network 서버1로 보낸다.

Meta data의 세부내용들은 section 17에 설명된다.

중복제거는 하나의 backend 전송에서 같은 uplink packet의 다수의 복사본을 묶는(baching) 과정이다.

Forwarding network 서버가 중복제거를 하든 하지 않든 재량(description)이다.

Forwarding network 서버가 주어진 Uplink packet 다수의 복사본을 받았을 때 중복 제거하는 데 기다리는 시간은 중복제거 window라고 불린다.

**단계 5:** Network 서버1은 만약 제공받은 NetID로 식별된 network operator와 passive roaming을 수락했다면 살펴볼 것이고 만약 동의가 없다면 result = NoRoamingAgreement를 가지는 PRStartAns를 돌려주기로 결정할 것이다.

Network 서버1은 PHYPayload로부터 end 디바이스의 Devaddr를 얻어낼 것이고, 대응하는 network session integrity key(SNwkSIntKey 그리고 FNwkSIntKey R1.1에서, NwkSkey R1.0/1.0.2에서)를 식별하고 PHYPayload에서 MIC를 검증할 것이다.

만약 Devaddr이 발견되지 않는다면 Network 서버1은 Result = unknownDevaddr를 가지는PRSrartAns를 돌려보낼 것이다.

만약 FCntUp이 이미 사용되었다면 network 서버1은 Result = FrameReplayed를 가지는 PRStartAns를 돌려보낼 것이다.

만약 MIC 검증이 실패했다면 network 서버 1은 Result = MICFailed를 가지는 PRStartAns를 돌려보낼 것이다.

**단계 6:** 만약 식별된 end 디바이스가 설정되었다면 Passive Roaming을 사용하고 network 서버1이 network 서버2로 가는 passive routing 활성화 하거나 확장하기로 결정했다면 network 서버1은 network 서버2로 Result = Success, passive roaming과 연관된 DevEUI, ServiceProfile, 그리고 Lifetime을 가지는 PRStartAns 보낼 것이다.

역시 Network 서버1은 FCntUp과 FNwkSIntKey(R1.1) 또는 NwkSkey(R.1.0/1.0.2)를 PRStartAns 메시지안에 포함한다, 만약 network 서버1과 network 서버2간에 Passive roaming 동의가 필요하다면 network 서버2는 network 서버2는 uplink packet에 대해서 MIC 검사를 수행할 것이다.

만약 network 서버1가 이미 network 서버2로부터의 같은 uplink packet의 다른 복사에 대해 반응했다면 network 서버1은 오직 result = Success 그리고 DupUL을 가진 PRStartAns를 network 서버2로 보낼 것이다.

만약 network 서버1가 network 서버2를 통해 passive roaming을 바라지 않는다면 result = Deferred그리고 lifetime을 가지는 PRStartAns를 network 서버2로 보낼 것이다.

Network 서버2가 이 메시지에 대해 lifetime동안 같은 end 디바이스에 대해 더 이상 PRStartReq를 network 서버1로 보내지않는다.

만약 network 서버1가 이미 network 서버2로부터의 같은 uplink packet의 다른 복사본에 대해 응답했다면 network 서버1은 result = deferred와 DupUL를 가진 PRStartAns를 network 서버2로 보낼 것이다.

만약 실패가 발생했다면 단계 5에서 network 서버1은 식별된 결과와 PRStartAns를 network 서버2로 보낸다.

만약 network 서버1가 network 서버2를 통해 passive roaming을 바라지 않는다면 오직 같은 결과와 DupUL을 가진 PRStartAns을 network 서버2로 보낼 것이다.

Network 서버1이 같은 시간에 만약 다수의 network 서버들로부터 PRStartReq를 받았다면 그들 중 0~ 다수와 passive roaming을 하기로 선택할 것이다.

Network 서버1과 network 서버2는 연결의 lifetime의 만기 이후에 passive roaming을 그들 스스로 종료할 것이다.(추가적인 신호없이), 만료전에 passive roaming이 새로운 PRStartReq/PRStartAns의 새로운 단계와 함께 확장되지 않는한.

Stateless forwarding network 서버의 동작에서, network 서버1은 Passive Roaming에 관한 lifetime값을 0으로 설정할 것이다.

**단계 7:** network 서버2는 성공적인 PRStartAns를 받자마자 end 디바이스의 lora 세션에서 forwarding 서버가 될 것이다.

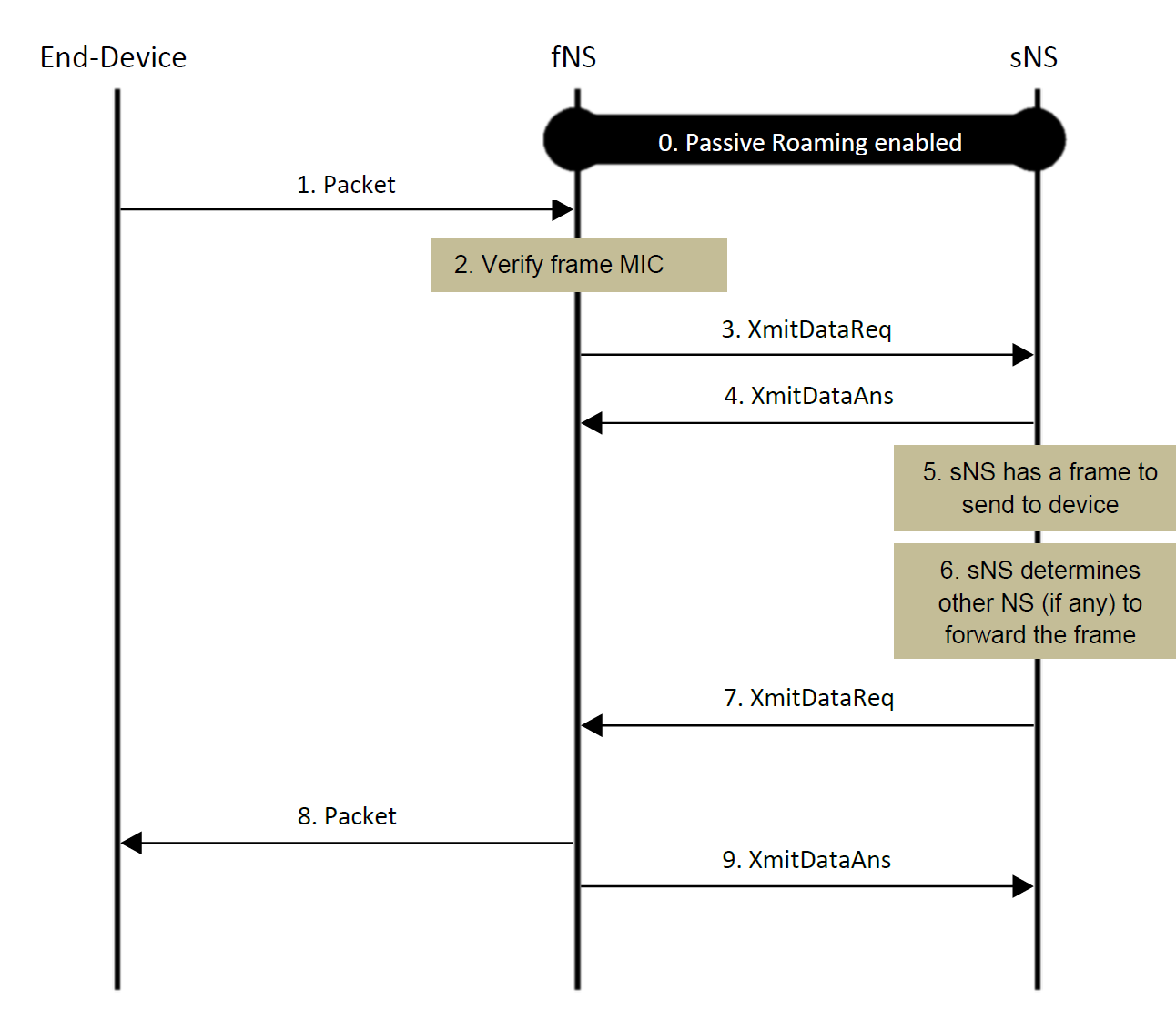
Network 서버1은 serving network 서버로서 계속 제공될 것이다.

이 지점이후에, network 서버2는 end 디바이스로부터 받은 packet을 network 서버1로 전달할 것이고 network 서버1은 network 서버2로부터 그러한 packet을 받을 것이다.

또한 network 서버1은 end 디바이스에게 packet을 보내기위해 network 서버2를 대체 forwarding network 서버로서 기록할 것이다.

Network 서버2는 network 서버1가 보낸 packet을 GW들의 중 하나로써 end 디바이스로 보내기위해 수용한다.

**Packet Transmission**



이 그림에서 passive roaming을 사용하는 end 디바이스에서 보내고 받는 packet들의 메시지의 흐름을 설명한다.

심지어 이 흐름은 downlink에 의해 즉시 잇따라오는 Uplink packet일지라도 흐름의 uplink와 downlink의 부분들은 end 디바이스의 class가 허용하는 독립적으로 어떠한 순서로 실행될 수 있음을 보여준다.

Stateless의 forwarding network 서버의 절차의 경우에는, 각각의 uplink packet들은 section 11.3에 따라(단계 1-4가 아닌 이는 stateful serving network서버로 추측한다.) 처리될 것이다.

그럼에도 불구하고, 이부분의 단계 5-11은 downlink packet의 처리에서 적용 가능하다. 심지어 stateless forwarding network서버의 처리과정 역시 적용 가능하다.

이부분의 모든 단계들은 stateful serving network 서버의 절차 uplink와 downlink packet 처리과정에서 적용 가능하다.

**단계 0:** stateful Passive roaming은 end 디바이스에 대해 forwarding network 서버와 serving network 서버 사이에서 가능하다.

**단계 1:** end 디바이스는forwarding network 서버가 전달받는 packet을 전송한다.

**단계 2:** 만약 forwarding network 서버가 serving network 서버에서 forwarding network서버간의 passive roaming 협의에 기반한 uplink packets들에 대해 MIC 검증이 필요하다면, forwarding network 서버는 packet으로부터 end 디바이스의 DevAddr을 얻고 FNwkSIntKey/NwkSKey를 확인하고 packet의 MIC를 검증할 것이다.

만약 FNwkSIntKey/NwkSKey가 발견되지 않거나 MIC 검증이 실패했다면 forwarding network 서버는 packet을 거부할 것이다.

**단계 3:** 만약 end 디바이스가 확인되었다면, forwarding network 서버는 받았던 packet의 PHYPayload와 관련된 ULMetada를 가지는 XmitDataReq 메시지를 확인된 end 디바이스의 serving network 서버로 보낼 것이다.

**단계 4:** serving network 서버는 forwarding network 서버로 받은 XmitDataReq에 대한 결과를 가지는 XmitDataAns 메시지를 보낼 것이다.

만약 DevAddr이 발견 되지않았다면 serving network 서버는 Result = UnKnownDevAddr을 가지는 XmitDataAns를 돌려줄 것이다.

만약 받았던 packet의 FCntUp이 end 디바이스가 주었던 이전의 수용했던 packet의 FCntUP보다 적다면, serving network 서버는 Result = FrameReplayed를 가지는 XmitDataAns를 돌려줄 것이다.

만약 MIC 검증이 실패했다면, serving network 서버는 Result = MICFailed를 가지는 XmitDataAns를 돌려줄 것이다.

반대는 serving network 서버는 Result = Success인 XmitDataAns를 돌려줄 것이다.

만약 serving network 서버가 forwarding network 서버로부터의 같은 uplink packet의 다른 복사본에 대해 응답했다면, serving network 서버는 이전의 XmitDataAns와 같은 결고와 DupUL을 가지는 XmitDataAns를 보낼 것이다.

Serving network 서버가 end 디바이스에게 packet을 보낼 때 이전의 단계를 따르거나 따르지 않을지도 모르는 후속단계들은 실행된다.

**단계 5:** serving network 서버는 end 디바이스에게 packet을 보낸다.

**단계 6:** serving network 서버는 자신의 아래의 GWs들중 하나를 통해 packet을 전달하던지 forwarding network 서버의 제어 아래의 GW를 통해 보내던지 결정할 것이다.

**단계 7:** 만약 serving network 서버가 forwarding network를 통해(via) packet을 보내기로 결정했다면 serving network 서버는 packet의 PHYPayload와 DLMetadata를 가지는 XmitDataReq 메시지를 forwarding network 서버로 보낼 것이다.

**단계 8:** 만약 받은 XmitDataReq에 오류 조건이 있다면, forwarding network 서버는 Resultㅇ를 오류 값으로 맞춰진 XmitDataAns를 serving network 서버로 보내고 packet을 전송할 시도를 하지 않을 것이다.

반면에 forwarding network 서버는 serving network 서버로부터 제공받은 metadata 정보에 기반해 end 디바이스에게 전송할 시도를 할 것이다.

만약 metadata가 GWinfo, ULToken 정보를 포함한다면, forwarding network 서버는 downlink 전송 GW를 선택하기 위해 이를 사용할지도 모른다.

Forwarding network 서버가 시간제한(constraint)과 network 상태 때문에 packet을 전송하는 것을 실패할지도 모른다..

이 경우에는 forwarding network 서버는 재전송을 하지 말아야 할 것이다.

**단계 9:** packet을 전송하기 시도하는 이후에, forwarding network 서버는 하나 또는 두개의 DLFreq1과 DLFreq2(packet이 RX1 또는 RX2 또는 둘다 보냈는지에 따라)와 성공적인 전송에 대해 Result = Success을 가지고 그렇지않으면 Result = XmitFailed 값인 XmitDataAns 메시지를 serving network 서버로 보낼 것이다.

**Passive Roaming Stop**

테이블이(가) 표시된 사진

자동 생성된 설명 테이블이(가) 표시된 사진

자동 생성된 설명

그림 1. 그림2

이 두 그림들은 Passive roaming 전송에 대한 메시지를 설명한다.

이 과정은 오직 stateful forwarding 서버에서만 적용 가능하다.

**그림 1 단계 0:** passive roaming은 end 디바이스에 대해 forwarding network 서버와 serving network 서버 사이에서 가능하다.

**그림 1 단계 1** serving network 서버가 passive roaming lifetime의 종료 이전에 end 디바이스에 대해 passive roaming을 종료하기로 결정했을 때, serving network 서버는 DevAddr 과 DevEUI 그리고 선택적으로 lifetime을 가지는 PRStopReq 메시지를 forwarding network 서버로 보낸다.

머무르는 기간동안 만약 forwarding network 서버가 stateful하고 serving network서버가 forwarding network 서버로부터 end 디바이스에 대해 다른 PRStartReq를받기를 원하지 않는다면 Serving network 서버는 lifetime을 포함한다.

**그림1 단계 2:** forwarding network 서버는 이미 passive roaming안에 있고 serving network 서버와 연관 되어있는 end 디바이스를 DevEUI로 검증할 것이다.

만약 두 조건이 만족한다면 forwarding network 서버는 Result = success 인 PRStopAns 메시지를 serving network 서버로 보낼 것이다.

반대로 forwarding network 서버는 Result = UnknownDevEUI인 PRStopAns 메시지를 serving network 서버에 보낼 것이다.

만약 lifetime을 포함하는 PRStopReq 메시지를 수신했다면 forwarding network 서버는 serving network 서버로 end 디바이스에 대한 다른 PRStartReq를 lifetime이 만료될 때 까지 보내지 않을 것이다.

End 디바이스에 대한 Passive roaming 이 PRStopReq 메시지 또는 PRStartAns/Result = deferred로 이전에 만료되었을 경우에, 새로운 lifetime 값이 0인 PRStopReq 메시지는 network 2가 end 디바이스로부터 packet을 받자마자 network 2는 end 디바이스에 대해 PRStartReq를 다시 보내는 것이 가능하다.

이는 stateful 한 forwarding network 서버에서 가능하다.

**그림 2 단계 0:** passive roaming은 end 디바이스에 대해 forwarding network 서버와 serving network 서버 사이에서 가능하다

**그림 2 단계 1:** forwarding network 서버가 passive roaming lifetime의 종료 이전에 end 디바이스에 대해 passive roaming을 종료하기로 결정했을 때, forwarding network 서버는 end 디바이스의 DevEUI를 가지는 PRStopReq 메시지를 serving network 서버로 보낸다.

**그림 2 단계 2:** serving network 서버는 자기 자신에 의해 제공되었고 이미 forwarding network 서버와의 passive roaming안에 있는 DevEUI를 가지고 end 디바이스를 검증한다.

만약 2조건이 만족한다면 serving network 서버는 Result = Success인 PRStopAns메시지를 forwarding network 서버로 보낼 것이다.

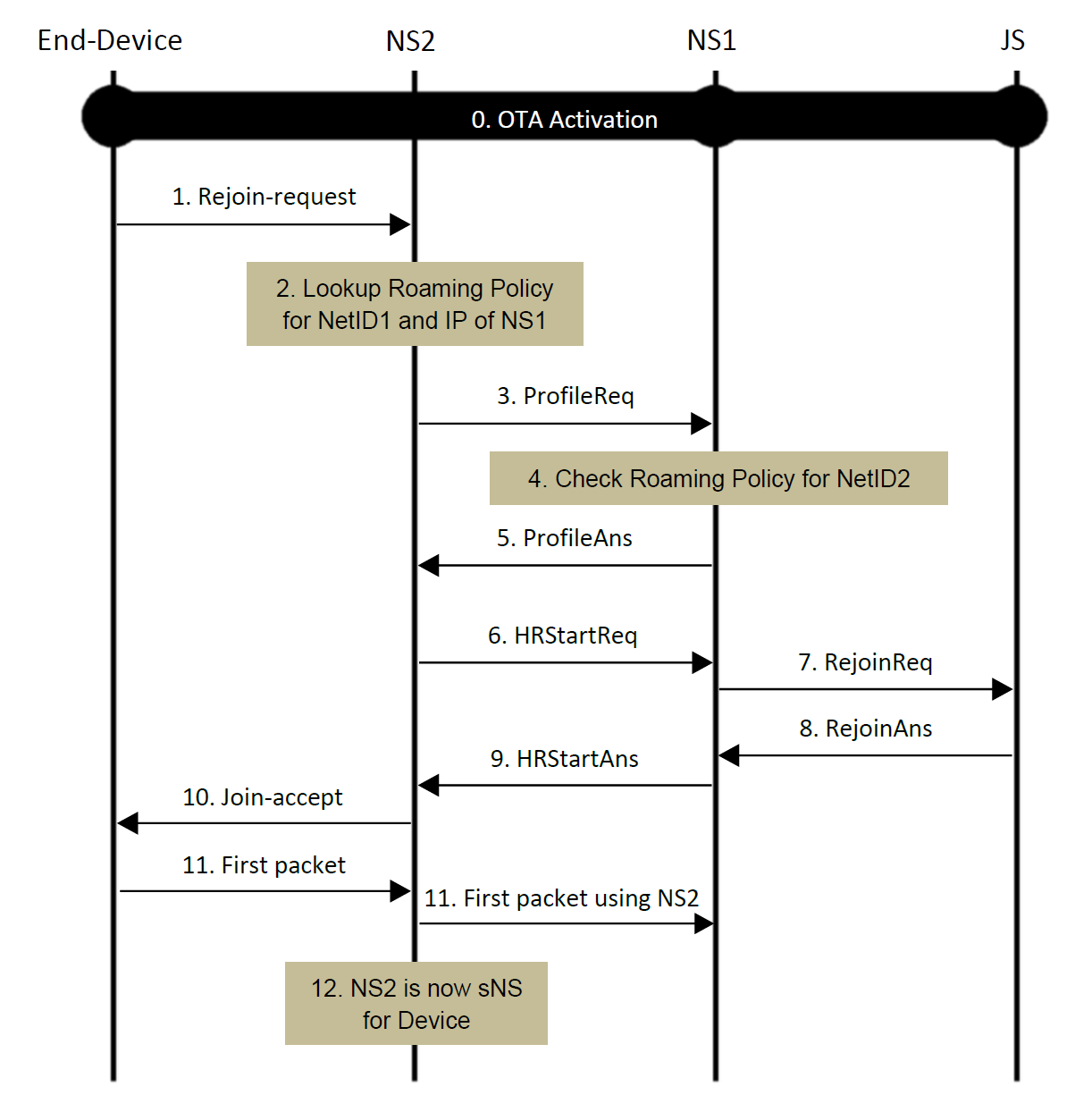
반면에 serving network 서버는 Result = UnKnownDevEUI인 PRStopAns 메시지를 forwarding network 서버에게 보낼 것이다.

Passive roaming 종료 이후에, serving network 서버와 forwarding network 서버는 end 디바이스가 설계한 서로에게 향하는 packet들을 보내는 것을 멈출 것이다.

**Handover Roaming**

이 과정은 오직 R1.1[LW11]end 디바이스와 network에서만 적용 가능하다.

**Handover Roaming Start**



이 그림은 end 디바이스의 LoRa 세션에게 가는 handover Roaming 과정에 대한 메시지의 흐름을 설명한다.

새로운 lora 세션의 handover roaming 활성화는 Section 12.1에서 제공한다.

**단계 0:** end 디바이스가 network 서버 1에 대해 활성화를 했다고 고려해라.

그러므로 network 서버1은 end 디바이스에 대해 home network 서버로서 활동한다.

**단계 1:** end 디바이스는 ForceRejoinReq MAC(보여주지않음) 명령어를 받는 것에 대한 답이거나 외부의 trigger없이 자동적으로 rejoin-request Type 0 메시지를 전송한다.

**단계 2:** 만약 network 서버2가 받은 DevEUI에 의해 구별되는 end 디바이스에 대해 serving network 서버로써 활동하고 있다면, step 6을 진행한다.

만약 network 서버2가 end 디바이스에 대해 serving network 서버가 아니라면, rejoin-request에서 network 서버2 NetID로 구별되는 operator와 자신의 roaming 정책을 살펴볼 것이다.

만약 network 서버2가 확인된 operator와 handover roaming이 가능하게 구성할 수 없다면, network 서버2는 rejoin-request를 무시하고 절차를 여기서 끝낸다.

그렇지않으면 만약 network 서버2가 미리 외부의 메커니즘에 의해 network 서버1의 IP주소/host name을 구성하지 않았다면 network 서버2가 DNS를 통해 network 서버1의 IP 주소를 발견한다.

**단계 3:** 만약 network 서버2가 자신의 cache에 end 디바이스의 Device Profile을 가지고있지 않는다면network 서버2는 DevEUI를 가지는 ProfileReq 메시지를 network 서버1에게 전송한다.

만약 ProfileReq가 보내지않았다면 단계 4 – 5는 넘어간다.

**단계 4:** network 서버1은 전달받은 NetID에 의해 구별되는 operator와 자신의 roaming 정책을 살펴본다.

**단계 5:** 만약 network 서버1이 end 디바이스에 대해 network 서버2와 Handover roaming이 가능하게 구성했다면 network 서버1은 network 서버2 에게 Result = Success, Device Profile, 그리고 Device Profile Timestemp(마지막 Device Profile의 변화의 timestemp를 가지는)인 ProfileAns를 보낸다.

만약 handover roaming이 불가능하다면, ProfileAns는 Result = NoRoamingAgreement 또는 DevRoamingDisallowed, 그리고 lifetime을 수반하고 절차를 여기서 종료한다.

Lifetime은 network 서버1이 network 서버2에게 lifetime이 만료될 때까지 end 디바이스에 대해 추가적인 ProfileReq를 보내지않는 것을 요구하는 것을 허락한다.

**단계 6:** 만약 network 서버2가 전달받은 DevEUI를 통해 식별되는 end 디바이스에 대해 serving network 서버로써 활동하고 network 서버2가 Rejoin-request 절차를 network 서버1에게 요구하지 않는다면 network 서버2는 network 서버1에게 rejoin-request 메시지, informative = True, MAC version, ULMetadata, Device Profile Timestamp와 함께 PHYPayload를 가지는 HRStartReq 메시지를 보낼 것이다.

반면에 network 서버2는 network 서버2에 의해 구별되고 end 디바이스에 적용될 rejoin-request 메시지, MAC version, ULMetadata, Device Profile Timestamp, DevAddr 파라미터들, DLSetting들, RxDelay, 그리고 부가적인 CFList와 함께 PHYPayload를 가지는 HRStartReq 메시지를 network 서버1로 보낸다.

Network 서버2는 MAC 버전의 값을 end 디바이스와 network 서버2사이에서 높은 일반적인 버전으로 정한다.

**단계 7:** 만약 handover roaming이 network 서버2에 대해 또는 end 디바이스에서 허락되지 않거나 메시지의 MIC 검증이 실패했다면, network 서버1은 아마 per-network 서버 또는 per-device roaming 정책 또는 end 디바이스가 이미 다른 serving network 서버로부터 제공받고 있는 동안의 handover roaming의 잠재적인 불필요성 때문에 단계 9. Handover Roaming rejection을 진행한다.

만약 network 서버1이 제공받은 Device Profile Timestamp에 의해 표시된 시간 덕분에 Device Profile이 변경되었다고 결정했다면, network 서버1은 network 서버2가 오래된(stale) 디바이스 profile을 가진다고 끝맺을 것이다.

이 경우에서 network 서버1은 단계 9를 진행한다.

만약 network 서버2가 전달받은 DevEUI에 의해 인증된 end 디바이스에 대해서 serving network 서버로 작동하고 network 서버2가 network 서버1이 Rejoin-request과정을 요구하지 않으면, network 서버1은 단계 9를 진행한다.

그렇지 않으면 network 서버1이 join 서버로 PHYpayload와 network 서버2로부터 받은 Rejoin-request 메시지, MAC 버전, DevEUI, DevAddr, DLSetting들, RxDelay, 그리고 CFList를 가지는 RejoinReq 메시지를 전달한다.

**단계 8:** 만약 Rejoin-Request가 join 서버에 의해 받아지면 join 서버는 MAC 버전에 따라 Rejoin-request를 진행하고 Result = Success, join-accept 메시지, SNwkSIntKey, NwkSEncKey, Lifetime과 함께 PHYPayload를 가지는 RejoinAns 메시지를 network 서버1에게 보낸다.

그렇지 않으면, join 서버는 Result = UnknownDevEUI 또는 MICFailed 또는 FrameReplayed인 RejoinAns를 network 서버1에게 보낸다.

Network 서버1은 전달받은 lifetime 값을 LoRa 세션에 의해 부여된 세션의 lifetime의 상한으로 취급할 것이다.

**단계 9:** 만약 network 서버1이 HRStartReq 메시지가 Rejoin-request가 처리되길 요청하지 않은 end 디바이스의 현재 serving network 서버로부터 왔다고 판단했다면, network 서버1은 Result = NoAction을 가지는 HRStartAns 메시지를 network 서버 2로 보내고 절차를 여기서 종료할 것이다.

만약 network 서버1이 단계 7에서 handover roaming을 허락하지않기로 결정했다면 network 서버1은 Result가 실패한 값으로 정하고(table 25보기), lifetime을 가지는 HRStartAns 메시지를 network 서버2로 보낼 것이다.

Lifetime은 network 서버1이 lifetime이 만료 될 때까지 network 서버2가 end 디바이스에 대해 추가적인 HRStartReq를 보내지 않도록 요구하도록 허락한다.

만약 network 서버1이 network 서버 2에게 알려진 Device Profile이 오래되었다는 것을 포함한다면 network 서버1은 Result = StaleDeviceProfile, 이전의 Device Profile, 그리고 Device Profile Timestamp를 가지는 HRSraerAns 메시지를 network 서버2에게 보낸다.

Network 서버2는 단순히 전달받은 새로운 Device Profile을 사용하기위해 단계 6으로 되돌아간다.

그렇지 않으면, network 서버1은 전달받은 RejoinAns 메시지의 payload와 DLMetadata 그리고 Service profile을 포함하는 HRStartAns 메시지를 network 서버2로 전달한다.

Network 서버1은 전달받은 SNwkSIntKey를 캐시에 저장할것이다. 그것이 뒤따라오는 rejoin Type 0 메시지의 MIC를 검증하기 위해 join 서버로 그것을 전달하기 이전에

**단계 10:** 만약 HRStartAns 메시지가 성공을 가르킨다면, network 서버2는 join-accept 메시지와 전달받은 PHYPayload를 end 디바이스에게 전달한다.

그렇지 않으면 network 서버2는 end 디바이스에게 어떠한 응답을 보내지 않는다.

만약 Rejoin 절차가 성공적이라면, network 서버2는 end 디바이스에서 network 서버1로 보내는 packet을 전달하기 시작하고 network 서버1은 network 서버2에서 오는 packet을 수용할 것이다.

또한 network 서버1은 application 서버로부터 network 서버2로 가는 packet을 전달하기 시작하고 network 서버2는 network 서버1로 오는 그러한 packet을 수용할 것이다.

**단계 11:** end 디바이스는 첫번째 uplink packet을 보낸다.

Network 서버2는 그 packet을 network 서버1로 전달할 것이다.

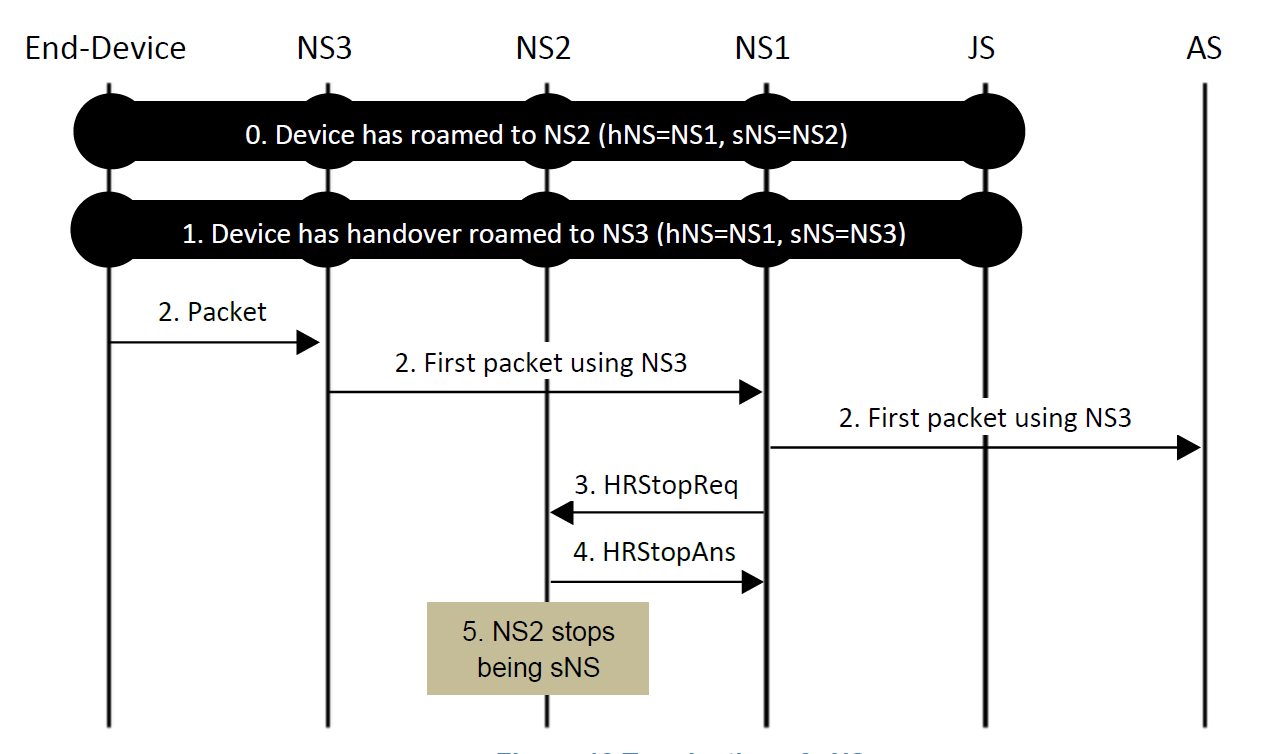
**단계 12:** network 서버2는 serving network 서버로서 제공하는 것을 시작하고 network 서버1은 end 디바이스로부터 첫번째 uplink packet을 제공받자마자 serving network 서버로서 제공하는 것을 멈춘다.

그동안에(meanwhile) network 서버1은 end 디바이스의 home network 서버로서 제공하는 것을 계속한다.

**Packet Transmission**

Handover roaming의 경우에서, home network 서버와 serving network 서버는 그들이 Passive roaming 에서 사용했던 같은 방식인 XmitDataReq/Ans 메시지를 사용할 것이다.

**Handover roaming Stop**



위 그림은 end 디바이스가 새 serving network 서버로 handover roaming을 수행한 이후에 이전에 제공된 Handover roaming와 home network 서버가 handover roaming을 끝내는 것을 설명한다.

**단계 0:** end 디바이스는 handover roaming을 network 서버 1과 network 서버 2 사이에서 실행한다.

**단계 1:** end 디바이스는 handover roaming을 network 서버1과 network 서버 3사이에서 실행한다.

**단계 2:** end 디바이스로부터의 첫번째 uplink packet은 network 서버1이 새로운 serving network 서버(network 서버3)를 통해 전달받는다.

**단계 3:** end 디바이스로부터 첫번째 packet을 새로운 serving network 서버(network 서버3)에 의해 받았을 때 network 서버 1은 DevEUI를 가지는 HRStopReq 메시지를 이전에 제공받았던 serving network 서버(network 서버2)로 전달한다.

HRStopReq 메시지는 이 DevEUI에 대해 지정된 시간내에 network 서버1가 다른 HRStartReq를 network 서버2로부터 수신하길 바라지않는 부가적인 Lifetime 수반한다.

**단계 4:** 이전에 제공받았던 serving network 서버(network 서버2)는 handover roaming을 종료할 것이고 만약 network 서버2가 제공받은 DevEUI를 구별되고 network 서버 1과 연관된 end 디바이스에 대해 handover roaming을 활성화 했다면 Result = Success를 가지는 HRStopAns를 network 서버 1에게 보낼 것이다.

만약 network 서버2가 network 서버1과 연관된 end 디바이스에 대해 handover roaming을 활성화하지 않았다면 network 서버2는 Result = UnknownDevEUI를 가지는 HRStopAns를 network 서버1로 보낼 것이다.

**단계 5:** network 서버2는 end 디바이스의 lora 세션을 serving network 서버로서 제공하는 것을 멈춘다.

만약 network 서버2는 다른 network 서버와 end 디바이스의 lora 세션에 대해 passive roaming이 가능했다면 network 서버 2는 그 network 서버와 passive roaming을 종료할 것이다.

End 디바이스에 대한 handover roaming이 HRStopReq 명령어를 통해 이전에 종료되었을 경우, lifetime의 값이 0인 새로운 HRStopReq 명렁어는 network 서버2가 새로운 rejoin-request Type 0 메시지를 수신하는 즉시 이 end 디바이스에 대해 HRStart requestㄴ를 다시 보내는 것이 가능하다.

Serving network 서버가 roaming을 종료하려고 결정했을 때가 Handover roaming 종료의 다른 case이다.

ForceRejoinReq 명령어를 end 디바이스에 보냄으로써 Serving network 서버는 종료 절차를 앞설지(preced)도 모른다

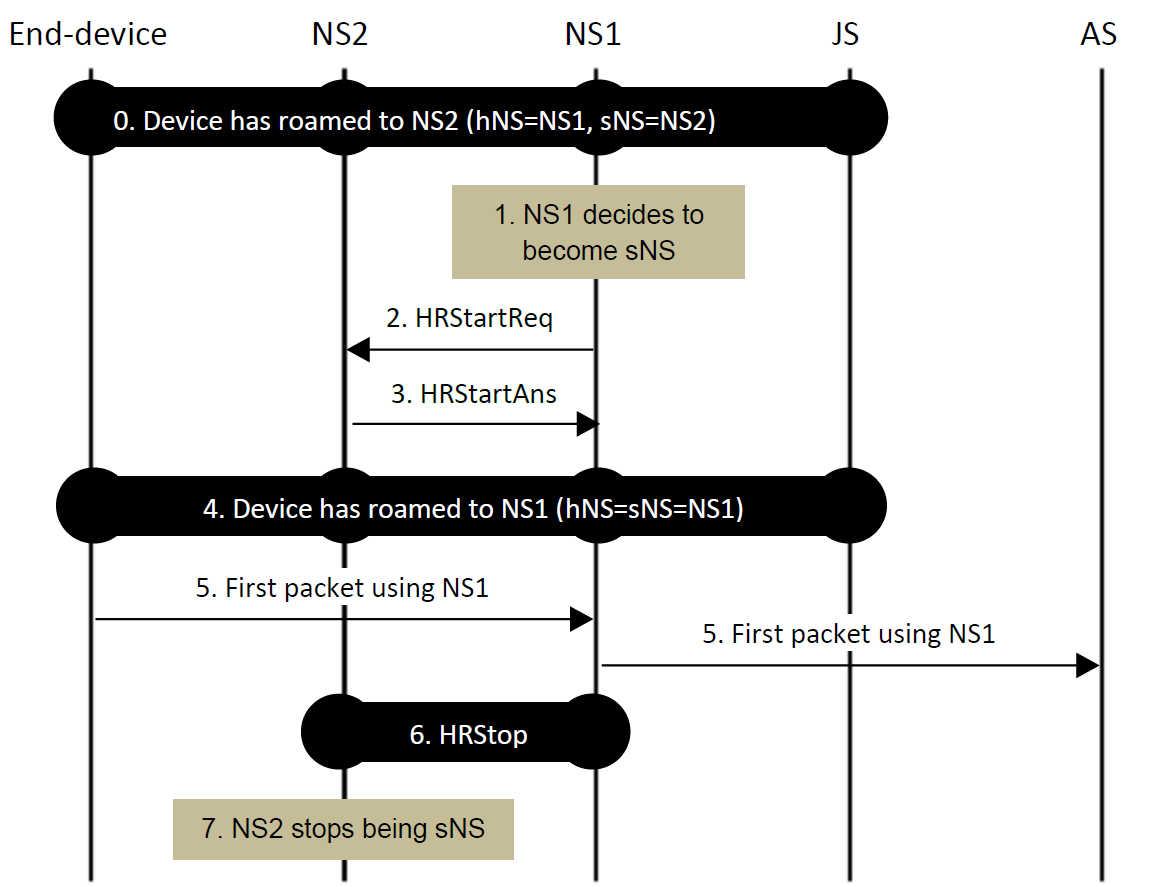
그러고 난 다음 Serving network 서버는 home network 서버에게 DevEUI를 가지는 HRStopReq를 보낼 것이다.

만약 home network 서버가 받은 DevEUI로 구분되고 serving network 서버와 연관된 end 디바이스에 대해 handover roaming을 활성화 했다면 Home network 서버는 handover roaming을 종료하고 Result = Success인 HRStopAns메시지를 serving network 서버에게 보낼 것이다.

만약 home network 서버가 serving netwok 서버와 연괸된 end 디바이스에 대해 handover roaming을 활성화 하지 않았다면, home network 서버는 result = UnknownDevEUI를 가지는 HRStopAns를 serving network 서버로 보낼 것이다.

심지어 serving network 서버가 실패된 결과를 home network 서버로부터 받았음에도Serving network 서버는 아마도 여전히 handover roaming을 종료할 수도 있다.

**Home network server Regaining Control**



위 그림은 현재 제공된 serving network 서버를 제어함으로써 home network 서버가 serving network 서버가 되는 메시지의 흐름을 설명한다.

**단계 0:** end 디바이스는 handover roaming을 network 서버1과 network 서버2 사이에서 수행한다.

**단계 1:** network 서버1은 serving network 서버로 되기를 결정한다.

**단계 2:** network 서버1은 handover roaming을 활성화 하기위해 DevEUI를 가지는 HRStartReq 메시지를 network 서버2(?)에게 보낸다.

**단계 3:** 만약 network 서버2가 전달받은 DevEUI를 통해 확인하고 network 서버1과 연관된 end 디바이스에 대해 handover roaming을 활성화 했다면 network 서버2는 result = Success를 가지는 HRStartAns를 network 서버1에게 전달한다.

그렇지않으면 result = UnknownDevEUI이다.

**단계 4:** network 서버2가 network 연결된(?) (handover roaming Start에 기술된) handover roaming을 시작할 것이다.

이 절차를 시작할 때 network 서버1이 End 디바이스가 network 서버1의 무선 커버리지 안에 있고 network 서버1이 network 서버2가 포함된 다른 network 서버들의 handover roaming을 거절하고 serving network 서버가 된다고 추측된다.

**단계 5:** 첫번째 uplink packet은 network 서버1에 의해 end 디바이스로부터 직접 전달 받는다.

**단계 6:** network 서버1은 handover roaming Stop 절차를 network 서버2와 수행 (**Handover Roaming Stop**에서 기술)한다.

**단계 7:** network 서버2는 end 디바이스의 lora 세션에 대한 serving network 서버로서 제공하는 것을 멈춘다

만약 network 서버2가 end 디바이스의 lora 세션에 대해 다른 network 서버와 passive roaming이 가능하다면 network 서버2는 역시 그 network 서버와 passive roaming을 종료할 것이다.

그 대신에 network 서버1은 end 디바이스가 자체적으로 handover roaming 시작하기로 결정하여 효과적으로 단계 2 그리고 3을 넘어가고 그리고 단계 4-7을 계속하며 기다릴 수 있다.(? 해석 다시)

1. OTA Roaming Activation Procedure

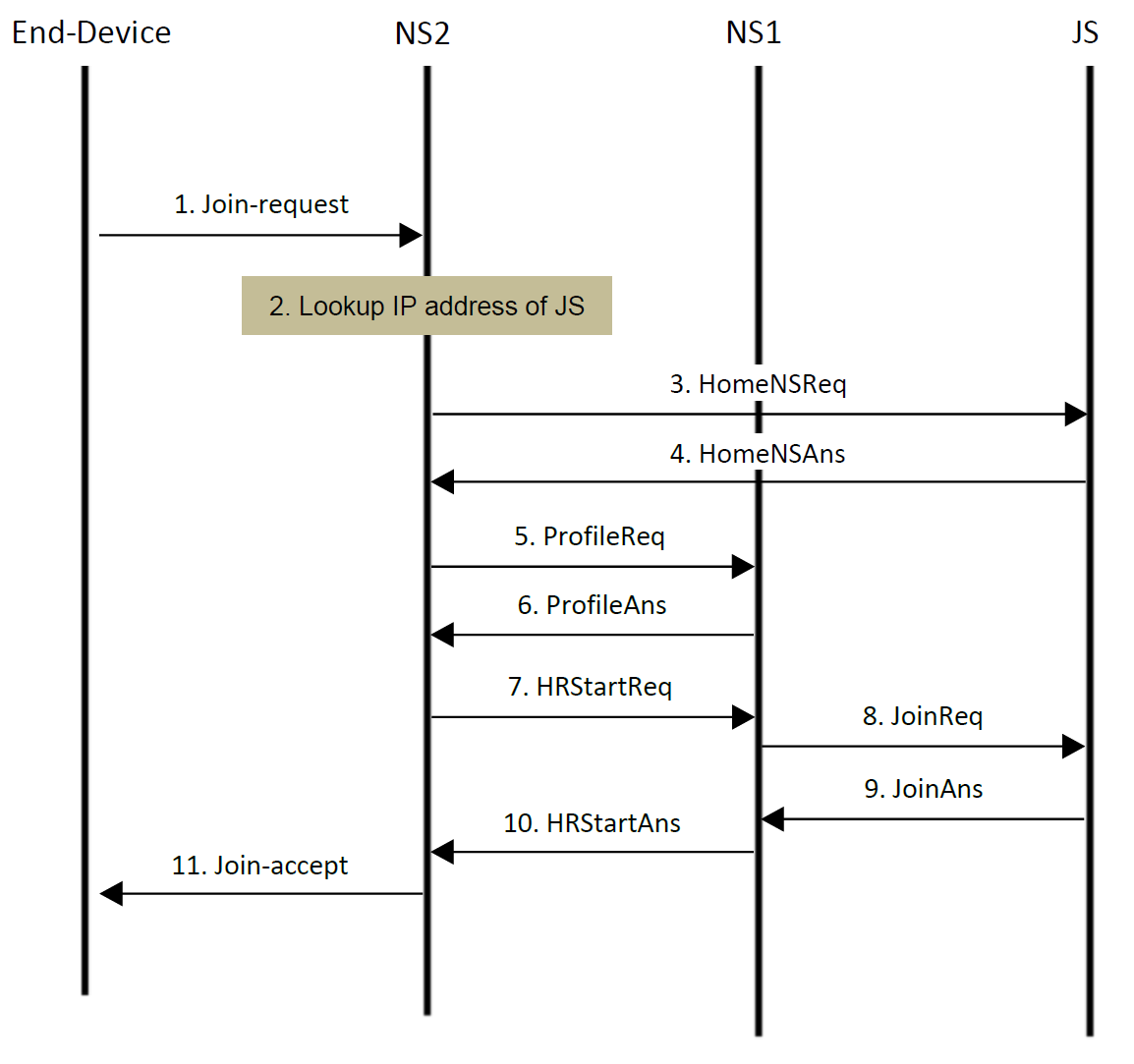
이 부분은 end 디바이스가 그의 home network 서버 커버리지 밖에 있으나 Visited network 서버안에 있을 때 새로운 lora 세션의 활성에 대한 절차를 소개하고있다.

Home network 서버가 Visited network 서버의 roaming 능력임을 인지했으며 home network 서버가 활성화의 방식(handover or passive roaming 기반)을 수행할지 결정했다는 것을 추측한다.

**Handover Roaming Activation**

이 절차는 R1.0[LW10, LW102]그리고 R1.1[LW11]의 end 디바이스와 network 둘 다 적용된다.

**Handover roaming Start**

****

위 그림은 OTA handover Roaming Activation 절차의 메시지의 흐름을 보여준다.

**단계 1:** end 디바이스는 join-request 메시지를 전송할 것이다.

**단계 2:** network 서버2가 join-request 메시지를 받았을 때, network 서버2는 DevEUI에 의해 확인된 end 디바이스에 대한 home network서버인지 아닌지를 결정할 것이다.

이 흐름에서 network 서버2는 end 디바이스의 home network서버가 아닌 것이 추측 가능하다.

다른 경우는 단계 8을 봐라.

Network 서버2는 JoinEUI에 의해 구별된 join 서버와 작동하기로 구성되든지 아닌지 결정된다.

만약 그렇게 구성되지 않는다면, network 서버2는 절차를 여기서 종료한다.

만약 network 서버2가 외부의 메커니즘에 의해 join 서버의 IP주소/호스트이름을 구성하지 않았다면 Network 서버2는 join request메시지에 기반한 join 서버의 IP주소를 보기위해서 DNS를 사용할 것이다.

만약 DNS에서 찾기에 실패했다면 network 서버2는 절차를 여기서 종료할 것이다.

**단계 3:** 만약 이미 network 서버2가 end 디바이스의 home network 서버의 존재를 알고 있다면, 단계 3 – 4는 넘어간다.

그렇지 않으면 network 서버2는 join request 메시지의 DevEUI를 가지는 HomeNSReq 메시지를 join 서버로 보낼 것이다.

**단계 4:** 만약 network 서버2가 Roaming 활성화에서 end 디바이스에게 제공하기 위해 join 서버안의 항목화 되어 인증된 네트워크 안에 있지 않을 경우에 join 서버는 Result = NoRoamingAgreement인 HomeNSAns 메지를 network 서버2에게 보낼 것이다.

Join 서버는 Result = Success, HNSID 그리고 end 디바이스의 HNetID(Network 서버1의 NetID)를 가지는 HomeNSAns 메시지를 network 서버2에게 보낼 것이다.

**단계 5:** 만약 network 서버2가 end 디바이스의 profile을 이미 알고 있고 network 서버2가 오직 network 서버1과 Handover roaming Agreement를 가지고 있다면, 단계 5-6은 넘어간다.

그렇지않고 만약 network 서버2가 미리 외부 메커니즘에 의해 network 서버1의 IP주소/hostname을 구성하지 않았다면 network 서버2는 받은 join-request 메시지안의 NetID에 기반한 network 서버1의 IP주소를 살펴보기 위해 DNS를 사용할 것이다.

만약 DNS에서 찾는 것이 실패했다면, network 서버2는 절차를 여기서 종료할 것이다.

Network 서버2는 DevEUI를 가지는 ProfileReq메시지를 network 서버1에게 보낼 것이다.

**단계 6:** 만약 network 서버1과 network 서버2간의 사업적인 동의가 없다면, network 서버1은 Result = NoRoamingAgreement를 가지는 ProfileAns 메시지를 network 서버2에게 보낼 것이다.

만약 network 서버1이 DevEUI로 end 디바이스를 식별할 수 없다면 network 서버1은 Result = UnknownDevEUI를 가지는 ProfileAns 메시지를 network 서버2에게 보낼 것이다.

만약 end 디바이스가 Roaming 활성화를 수행하게 허락되지않는다면, network 서버1은 Result = RoamingActDisallowed를 가지는 ProfileAns메시지를 network 서버2에게 보낼 것이다.

그렇지않으면, network 서버1이 Handover roaming이 가능하다고 결정하는 것을 추측하는 것은 Network 서버1가 Result = Success, RoamingActivationType = Handover, 디바이스 Profile 그리고(마지막 디바이스의 Profile 변화의 timestamp를 가지는)디바이스 Profile timestamp 를 가지는 ProfileAns메시지를 network 서버2에게 보낸다.

RoamingActivationType이 handover일 때 따라오는 단계들은 절차를 서술한다.

**단계 7:** 만약 들어오는 ProfileAns 결과가 성공을 가르키거나, 만약 단계 5-6이 넘어갔다면, network 서버2는 join-request메시지, MAC버전, ULMetadata, DevAddr, DLSettings, RxDelay, 부가적인 CFList, 그리고 디바이스 profile Timestamp와 함께 PHYPayload를 가지는 HRStartReq메시지를 network 서버1에게 보낸다.

Network 서버2는 MAC버전의 값을 end 디바이스와 network 서버2사이에 높은 공통버전으로 설정한다.

**단계 8:** 만약 단계 5-6이 넘어갔을 때, 만약 network 서버1과 network 서버2사이에 사업적 동의가 없거나 만약 network 서버1이 end 디바이스를 DevEUI을 통해 확인할 수 없거나 만약 end 디바이스가 Roaming activation 을 수행하도록 하지 못하게 허락하지않는다면 network 서버1은 단계 10을 진행할 것이다.

만약 network 서버1이 디바이스 Profile이 전달받은 디바이스 Profile Timestamp애 의해 가르켜지는 시간 때문에 변경되었다고 결정했다면, network 서버1은 network 서버2가 오래된 디바이스 Profile 정보를 가졌다고 결론을 맺을 것이다.

그 경우에서 network 서버1은 단계 10을 진행한다.

그렇지 않으면 network 서버1은 join 서버에게 network 서버2로부터 받은 join-request메시지, MAC 버전, DevEUI, DevAddr, DLSettings, RxDelay 그리고 CFList와 함께 PHYPayload를 가지는 joinReq 메시지를 보낸다.

**단계 9:** join 서버는 join-request 메시지를 MAC버전에 따라 처리하고 성공의 경우는 Result= Success, join-accept 메시지, network 세션키들(SNwkSIntKey, FNwkSIntKey,그리고 NwkSEncKey R 1.1경우에, NwkSkey R1.0/R1.0.2의 end 디바이스), 암호화된 AppSkey 또는 SessionKeyID 또는 둘다, LifeTime과 함께 PHYPayload를 가지는 joinAns를 network 서버1에게 보내고 Result = UnknownDevEUI 인 경우의 end 디바이스는 join 서버에 의해 인식되지않는다, Result = FrameReplayed 인 경우의 DevNonce는 이전에 사용되었고, Result = JoinReqFailed는 다른 오류경우이다.

Network 세션키들 그리고 AppSKey들은 LoRaWAN specification[LW10,LW11]에 기반해서 생성된다.

AppSKey는 join 서버에서 network 서버로 전달 될 때 join 서버와 application 서버간에 공유되는 키를 통해 암호화 된다.

**단계 10:** 만약 network 서버1과 network 서버2간의 사업적 동의가 없다면 network 서버1은 Result = NoRoamingAgreement를 가지는 HRStartAns 메시지를 network 서버2로 보낸다.

만약 network 서버1가 DevEUI로 end 디바이스를 확인하지 못한다면 network 서버1은 Result = UnknownDevEUI를 가지는 HRStartAns 메시지를 network 서버2에게 보낸다.

만약 end 디바이스가 Roaming Activation을 수행하도록 허락되지않는다면 network 서버1은 Result = RoamingActDisallowed를 가지는 HRStartAns 메시지를 network 서버2에게 보낼 것이다.

만약 network 서버1이 network 서버2에게 알려준 디바이스 Profile 오래되었다고 결론을 맺는다면 network 서버1은 Result = StaleDeviceProfile, 최신의 디바이스 Profile그리고 그 디바이스 Profile Timestamp를 가지는 HRStartAns 메시지를 network 서버2에게 보낸다.

이 경우에서 network 서버2는 받은 새로운 디바이스 Profile을 사용하기위해 단계 7로 돌아갈 것이다.

그렇지 않으면 network 서버1은 HRStartAns 메시지를 network 서버2에게 보낸다.

HRStartAns는 단계9에 설명된 JoinAns 메시지와 같은 대상과 end 디바이스의 Service Profile이 포함한다.

R1.1의 end 디바이스의 경우에는 network 서버1은 역시 전달받은 SNwkSIntKey를 뒤에 오는 Rejoin Type 0 메시지의 MIC를 검증하기위해 join 서버로 전달하기 이전에 캐싱한다.

**단계 11:** 만약 HRStartAns 메시지가 성공을 가르킨다면 network 서버2는 전달받은 PHYPayload와 Join-accept 메시지를 end 디바이스에게 전달한다.

End 디바이스는 전달받는 join-accept 메시지에 따라 LoraWAN Specification[LW10, LW11]에 기반해서 network 세션키들, AppSKey를 생성할 것이다.

Join 서버에서 Application 서버로 만약 암호화된 AppSKey가 이용가능하지 않다면 application 서버는 **OTA Activation at Home Procedure**의 단게 8과 같은 방법을 통해 join 서버를 통해 직접적으로 검색할 것이다.

**Packet Transmission**

End 디바이스에 대해 Home network 서버와 serving network 서버가 Roaming Activation에 참여하게 한 이후에 Home network 서버와 serving network 서버간의 uplink 그리고 downlink packet 전송의 세부사항은 Handover Roaming의 경우와 같고 **Packet Transmission**에 서술 되어있다.

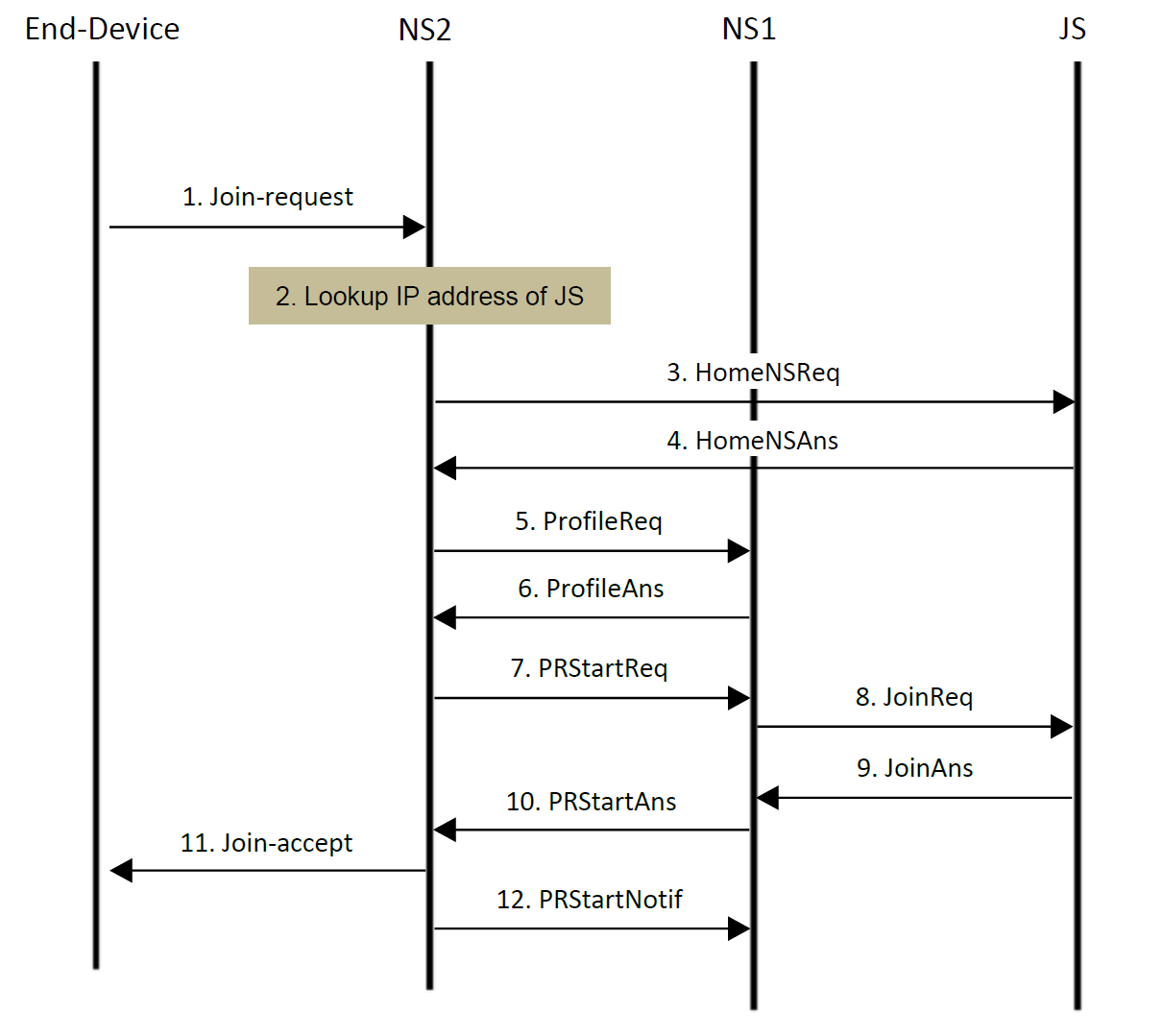
**Handover Roaming Stop**

Handover Roaming Stop 절차는 (**Handover Roaming Stop 구역**) home network 서버 또는 serving network 서버가 roaming을 끝내기로 결정했을 때 이용된다.

**Passive Roaming Activation**

절차는 R1.0[LW10,LW102]그리고 R1.1[LW11] end 디바이스와 network에서 적용된다.

**Passive Roaming Start**



위 그림은 OTA Passive Roaming Activation 절차를 위한 메시지의 흐름을 설명한다.

**단계 1:** end 디바이스는 join-request 메시지를 전송한다.

**단계 2:** network 서버2가 join-request 메시지를 받았을 때, network 서버2는 DevEUI로 구분되는 end 디바이스에 대한 home network 서버인지 아닌지를 결정한다.

이 흐름에서 network 서버2가 end 디바이스에 대한 home network 서버가 아니라고 추측된다.

다른 경우는 **OTA Activation at Home Procedure**를 봐라

Network 서버2는 그것이 JoinEUI로 구별된 join 서버와 함께 작동하기 위해 구성되었는지 아닌지를 결정할 것이다.

만약 그렇게 구성되지 않는다면, network 서버2는 여기서 절차를 종료할 것이다.

만약 network 서버2가 외부 메커니즘에 의해 join 서버의 IP 주소/hostname을 미리 구성하지 않았다면 Network 서버2는 join-request 메시지(**JoinEUI and NetID Resolution**에 자세한 설명)에 기반한 Join 서버의 IP 주소를 보기위해 DNS를 사용한다.

만약 DNS에서 찾는 것이 실패 했다면 network 서버2는 여기서 절차를 종료한다.

**단계 3:** 만약 network 서버2가 미리 end 디바이스의 home network 서버의 Identity를 안다면 단게 3 그리고 4는 넘어간다.

그렇지 않으면 network 서버2는 join-request 메시지의 DevEUI를 가지는 HomeNSReq 메시지를 join 서버에게 보낸다.

**단계 4:** 만약 network 서버2가 end 디바이스에게 Passive roaming Activation을 제공하기위해 join 서버안의 항목화 되어진 허가된 network 안에 있지 않다면 join 서버는 network 서버2에게 Result = NoRoamingAgreement인 HomeNSAns 메시지를 보내고 절차를 여기서 종료할 것이다.

Join 서버는 Result = Success, HNSID 그리고 end 디바이스의 HNetID(network 서버1의 NetID)를 가지는 HomeNSAns 메시지를 network 서버2에게 보낼 것이다.

**단계 5:** 만약 network 서버2가 오직 network 서버1과 Passive Roaming Agreement이 가능하다면 단계 5 그리고 6은 넘어간다.

그렇지 않고 만약 network 서버2가 외부 메커니즘에 의해 network 서버1의 IP 주소/hostname을 미리 구성하지 않았다면 network 서버2는 join 서버로부터 받은 NetID에 기반한 network 서버1의 IP주소/hostname을 찾아보기 위해 DNS를 사용할 것이다.

만약 DNS에서 찾는 것이 실패했다면 network 서버2는 절차를 여기서 종료할 것이다.

Network 서버2는 DevEUI를 가지는 ProfileReq 메시지를 network 서버1에게 전달한다.

**단계 6:** 만약 network 서버1과 network 서버2간의 사업적인 동의가 없다면 network 서버1은 Result = NoRoamingAgreement인 ProfileAns 메시지를 network 서버2에게 보낼 것이다.

만약 network 서버1이 DevEUI를 통해 end 디바이스를 검증할 수 없다면 network 서버1은 Result = UnknownDevEUI인 ProfileAns 메시지를 network 서버2에게 보낸다.

만약 end 디바이스가 Roaming Activation을 수행하도록 허락되지않는다면 network 서버1은 network 서버2에게 Result = RoamingActDisallowed인 ProfileAns 메시지를 보낼 것이다.

그렇지 않으면 network 서버1이 Passive Roaming Activation이 가능하다고 결정을 가정할 때, network 서버1은 network 서버2에게 Result = Success, RoamingActivationType를 가지는 ProfileAns 메시지를 보낸다.

RoamingActivationType가 Passive일 때 따라오는 것은 행동을 서술한다.

**단계 7:** 만약 들어오는 ProfileAns의 결과가 성공을 가르키거나 단계 5 그리고 6이 넘겨졌다면 network 서버2는 PHYPayload와 join-Request 메시지, ULMetadata, 그리고 만약 network 서버2가 이 packet에 중복제거를 수행했다면 DedupWindowSize를 가지는 PRStartRReq 메시지를 network 서버1에게 보낼 것이다.

**단계 8:** 단계 5 그리고 6이 넘어겨졌을 때, 만약 network 서버1과 network 서버2간의 사업적 동의가 없었거나 만약 network 서버1이 DevEUI를 통해 end 디바이스를 구분할 수 없었거나 만약 end 디바이스가 Roaming Activation을 수행하도록 허락되지않거나 만약 network 서버1이 network 서버2를 통해 Passive Roaming Activation을 하도록 바라지않는다면 network 서버1은 단계 10을 진행할 것이다.

그렇지않으면 network 서버1은 PHYPayload와 network 서버1에의해 구분되는 Join-request 메시지, DevEUI, DevAddr, DLSettings, RxDelay 그리고 부가적인 CFList를 가지는 joinReq 메시지를 join 서버에게 보낸다.

**단계 9:** 성공에 경우에는 join 서버 join-request 메시지를 처리하고 Result = Success, PHYPayload와 Join-accept 메시지, network 세션키들(SNwkSIntKey, FNwkSIntKey, 그리고 NwkSEncKey R1.1의 경우에서, 그리고 NwkSkey는 R1.0/R1.0.2의 end 디바이스에서), 암호화된 AppSkey 또는 SessionKeyID또는 둘다, Lifetime을 가지는 JoinAns를 network 서버1에게 보내고 Result = UnknownDevEUI인 경우는 end 디바이스는 join 서버에 의해 인지되지않고 Result = MICFailed인 경우는 join-request의 MIC는 인증에 실패했다, Result = FrameReplayed의 경우는 이전에 DevNonce가 사용되었다, Result = JoinReqFailed는 다른 오류 경우들이다.

Network 세션키들 그리고 AppSKey는 LoRaWAN specification [LW10, LW102, LW11]에 기반해서 생성된다.

AppSkey은 join 서버에서network서버로 전달될 때 join 서버와 application 서버간의 공유된 키를 통해 암호화 된다.

**단계 10:** 만약 network 서버1과 network 서버2간에 사업적인 동의가 없다면, network 서버1은 Rssult=NoRoamingAgreement인 PRStartAns메시지를 network 서버2에게 보낼 것이다.

만약 network 서버1이 DevEUI를 통해 end 디바이스를 검증할 수 없다면 network 서버1은 Result = UnknownDevEUI인 PRStartAns 메시지를 network 서버2에게 보낸다.

만약 end 디바이스가 Roaming Activation을 수행하도록 허락되지않는다면 network서버1은 Result = Roaming ActDisallowed인 PRStartAns 메시지를 network 서버2에게 보낸다.

만약 network 서버1dl netwok 서버2를 통해 Passive Roaming activation을 하기를 바라지않는다면 network 서버1은 Result = Deferred, Lifetime을 가지는 PRStartAns를 network 서버2에게 보낼 것이다.

Network 서버2는 network 서버1에게 같은 end 디바이스에 대해 전달받은 이 메시지에 따라 Lifetime이 완료될 때까지 더 이상의 PRStartReq를 보내지않을 것이다.

만약 network 서버1이 이미 network 서버2로부터의 같은 uplink packet 다른 복사본에 반응했다면 network 서버1은 이전의 조건들에 따라는 결과와 DupUL를 가지는 PRStartAns를 network 서버2에게 보낼 것이다.

그렇지않으면 network 서버1은 downlink forwarding network 서버/gateway을 고르기 위해 보내진 PRStartReq에 반응할 때 Result = Success, PHYPayload와 Passive Roaming과 연관된 join-request 메시지, DLMetadata, ServiceProfile, 그리고 Lifetime을 가지는 PRStartAns를 network 서버2에게 보낸다.

만약 network 서버2가 stateful한 forwarding network 서버로서 동작하면 Network 서버1은 역시 DevEUI를 포함한다. 그리고 PRStartAns메시지에 FCntUP그리고FNswIntKet(R1.1의 경우)또는 NwkSKey(R1.0/R1.0.2의 경우)포함해야한다.

만약 network 서버1-network 서버2간의 Passive Roaming 동의가 필요하다면 network 서버2는 uplink packet들에 대해 MIC 검증을 해야한다.

만약 network 서버1가 이미 network 서버2로부터온 같은 uplink packet의 다른 복사본에 이미 응답했다면 network 서버1은 Result = Success 그리고 DupUL을 가지는 PRStartAns를 network 서버2에게 보낼 것이다.

만약 network 서버2가 network 서버1에 의해 downlink forwarding network 서버로 정하지않았고 network 서버1이 network 서버2로부터 온 같은 uplink packet의 다른 복사본에 이미 반응하지 않았다면 network 서버1은 Result = Success인 PRStartAns를 network 서버2에게 보낼 것이다.

**단계 11:** 만약 PRStartAns 메시지가 성공을 가르키고 network 서버1 로부터 받은 downlink parameter들을 사용한다면 network 서버2는 전달받은 PHYPayload와 join-accept 메시지를 end 디바이스에게 전달한다.

End 디바이스는 전달받은 join-accept 메시지를 LoRaWAN specification [LW10, LW102, LW11]에 기반해서 network 세션키들과 AppSKey들을 생성한다.

**단계 12:** 만약 network 서버2가 network 서버1로부터 PHYPayload와 join-accept packet을 받았다면, 성공적인 전송에서 network 서버2는 DLFreq1 그리고 DLFreq2(packet이 RX1또는 RX2 또는 둘다 전송되었는지에 따라달려있다) 하나 또는 둘다와 Result = Success를 가지는 PRStartNotif 메시지를 network 서버1로 보낸다.

그리고 그렇지않으면 Result = XmitFailed 값이다.

Join 서버에서 Application 서버로 network 서버 때문에 만약 암호화된 AppSKey가 이용 가능하지 않다면 application 서버는 **OTA Activation at Home Procedure**의 단게 8과 같은 방법을 통해 join 서버를 통해 직접적으로 검색할 것이다.

절차가 성공적으로 완료되었다면 새롭게 생성되는 lora 세션의 network 서버2는 forwarding 서버가 되고 network 서버1은 serving network 서버가 된다.(게다가 home network 서버가 된다)

**Packet Transmission**

End 디바이스에 대해 Serving network 서버와 forwarding network 서버가 Roaming Activation에 참여하게 한 이후에 Serving network 서버와 forwarding network 서버 간의 uplink 그리고 downlink packet 전송의 세부사항은 Passive Roaming의 경우와 같고 **Packet Transmission**에 서술 되어있다.

**Passive Roaming Stop**

Serving network 서버이거나 forwarding network 서버 둘다 roaming 을 끝내기로 결정했을때 Passive Roaming Stop 절차는(**Passive Roaming Stop**)은 사용된다.

1. Geolocation

LoraWAN network end 디바이스의 위치를 결정하기 위해 이용할 수 있는 정보들에 기반한 광범위한 기술들을 활용할 수 있다.(TDoA(Time Difference of Arrival),RSSI(Received Signal Stregth Indication)등)

Network node에서 실행되는 지리위치 알고리즘은 uplink frames의 metadata를 사용하고 end 디바이스의 지리적인 좌표를 생성한다.

지리좌표는 network에 대한 부가적인 특징이다.

주어진 network는 지리좌표를 생성하기 위해서 ULMetadata를 활용하는 것을 가능하게 할지도 모르고 그러한 좌표들을 상위 network node(ex: application server는 home network 서버의 상위 node, home network 서버는 serving network 서버, serving network 서버는 forwarding network 서버)에게 보내거나 geolocation-specific ULMetadata를 상위 network node로 보내는 것이나 둘 다하거나 전혀하지 않을 지도 모른다.

Network operator들은 외부 매커니즘을 사용해서 그들의 지리적인 능력에 대해 그들의 파트너들과 협상을 하기를 바랬다.

Network가 geolocation-specific ULMetadata 또는 지리좌표를 제공하기를 동의했을 때, 장치별로 그렇게 하도록 지시할 수 있다.

상위의 network node에 의해 제공되는 ServiceProfile은 주어진 end 디바이스에 대해 만약 하위의 network 노드에서 상위 하나로 보내지길 기대되는 정보의 하나 또는 둘 모두를 나타낸다.

구체적으로 이 목적에 대해 정의된 SendLoc, LocSolverAuxData, AddLocMetadata 대상들을 봐라

하위의 network node가 geolocation-specific metadata를 제공할 때, 그러한 데이터들은 uplink packet에 따라 날라지는 ULMetadata에(AntennaID, FineRecvTime, FRTContext 그리고 ADRBit 대상들을 봐라) 더해진다.

FIneRecvTime 값은 GW에 의해 암호화 될지도 모른다. 이경우에서 FRTContext는 복호화 key를 검증하기위해 제공될 것이다.

FineRecvTime의 소비자에 의한 복호화 키를 검색하는 것은 이 specification의 범위를 넘어간다.

하위 network node가 지리좌표를 제공할 때, 그 정보는 상위 node에게 보내질 전용의 메시지(XmitLocReq를 봐라)를 다른다.

XmitLocReq 메시지의 생성 시간은 지리좌표를 생성하는 지리위치의 알고리즘에 의해 달려있다.

network 서버 또는 application 서버에서 분리된 노드에서 실행하도록 하는 지리위치 알고리즘을 요구되는 인터페이스는 specification의 범위를 벗어난다.

1. DevAddr Assignment

NetID는 Lora Alliance에 의해 LoraWAN network들에 부여된 24bit network 식별자이다.

값들 0x000000그리고 0x000001은 실험직인 network들을 위해 예약 되었고 network들은 roaming을 사용하지 않는다.

이러한 값들은 Lora Alliance의 허락을 얻는 것 없이 어느 network에 의해 사용될 수 있다.

LoraWAN network들 Lora Alliance에 의해 부여되는 unique한 NetID 값을 얻기 위해 roaming을 사용한다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 따라오는 필드들로 구성된(composed) NetID의 형식을 설명한다.

1. Type: NetID의 3MSB(Most Significant Bits)는 NetID의 Type을 (0~7) 가르킨다.
2. ID: NetID의 가변길이 LSB(Least Significant Bits)는 Lora Alliance에 의해 부여된다.  
   ID 필드의 길이는 NetID의 Type에 달려있다.
3. RFU: Type과 ID 필드들이 소비된 이후에 만약 NetID안에 사용되지 않는 어떠한 bit들이 있다면 그들은 표시된다 RFU로 그리고 0으로 설정된다.  
   만약 이러한 필드들이 아직 NetID의 24bit들을 소비함으로 써 사용하지않았다면 이러한 RFU bit들은 Type과 IDbit들 사이에 위치하고 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

이 그림은 각각의 NetID Type에 대한 Type 필드의 설정들, RFU bits들의 수, 그리고 ID 필드의 길이의 세부사항 제공한다.

예를 들어 NetID 값 0x000003은 Type 0 NetID와 ID = 3, 그리고 값 0x6000FF은 Type 3 NetID와 ID = 255이다.

테이블이(가) 표시된 사진

자동 생성된 설명

DevAddr은 LoraWAN network에 의해 부여된 end 디바이스 식별자이다.

위 그림은 다음 필드들로 구성된 DevAddr의 형식을 설명한다.

1. Type Prefix: 부여하는 network의 NetID Type을 가르키는 가변 길이 MSB
2. NwkID: Type Prefix 뒤에 따라오는 가변길이 bit들. 그들은 network를 확인하는 것에 사용된다.  
   NwkID의 값은 NetID의 ID 필드의 LSB의 미리정해진 숫자로 정해진다.
3. NwkAddr: network에 의해 end 디바이스에 부여되는 가변 길이 LSB

Specification안의 Type Prefix 그리고 LoraWAN link-layer specifications(ex: LoraWAN L2 1.0.4 Specification[LW104]안의 AddrPrefix 필드, L2 Specification의 이전 버전의 NwkID 필드)안의 NwkID 필드들의 연결은 NwkAddr 필드 앞에 있는 DevAddr 필드처럼 같은 값을 취한다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 Type Prefix 필드. NwkId의 크기 그리고 각각의 NetID의 Type NwkAddr 필드의 길이와 설정의 세부사항을 제공한다.

그것의 NetID에 기반한 그것의 end 디바이스의 DevAddr를 부여할 때 Network 서버는 위 그림에서 정의된 파라미터를 사용할 것이다.

NwkID bit들의 수가 NetID(Type 3~7)의 ID 필드안의 bit들의 수보다 적다면 이는 다수의 NetID들이 같은 NwkID 값과 연결되어 있음을 의미한다.

**Passive Roaming**은 어떻게 forwarding network 서버가 다수의 network 서버들에서 end 디바이스의 serving network 서버를 찾아내는지를 설명한다.

1. Periodic Recovery

Rejoin-request Type 1 메시지는 serving network 서버에 완벽한 상태를 잃었을 경우 end 디바이스와 연결성 회복에 대해 정의되어있다.

메시지는 주기적으로 (periodically) serving network 서버에게 회복할 기회를 주기위해 end 디바이스에 의해 전송된다.

Network 서버가 Rejoin-request Type 1을 받았을 때, network 서버는 그것이 유효한 Lora 세션과 전달받은 DevEUI에 의해 검증된 end 디바이스를 가졌는지 결정한다.

만약 network 서버가 end 디바이스에 대해 serving network 서버로 활동하지 않는다면 network 서버는 들어오는 Rejoin-Request Type 1을 그것을 join-request으로 처리하는 정확히 같은 방식으로 취급한다.(즉 network 서버로부터 join 서버로 join-request 메시지 대신 Rejoin-request 메시지를 전달함으로써 Activation at Home 또는 Roaming Activation Procedure를 따른다)

만약 network 서버가 end 디바이스에 대해 serving network 서버로 활동한다면, network 서버는 [LW11]의 Section 6.2.4.4처럼 행동한다.

이 절차들은 오직 R1.1[LW11]end 디바이스와 network 서버에서 적용된다.

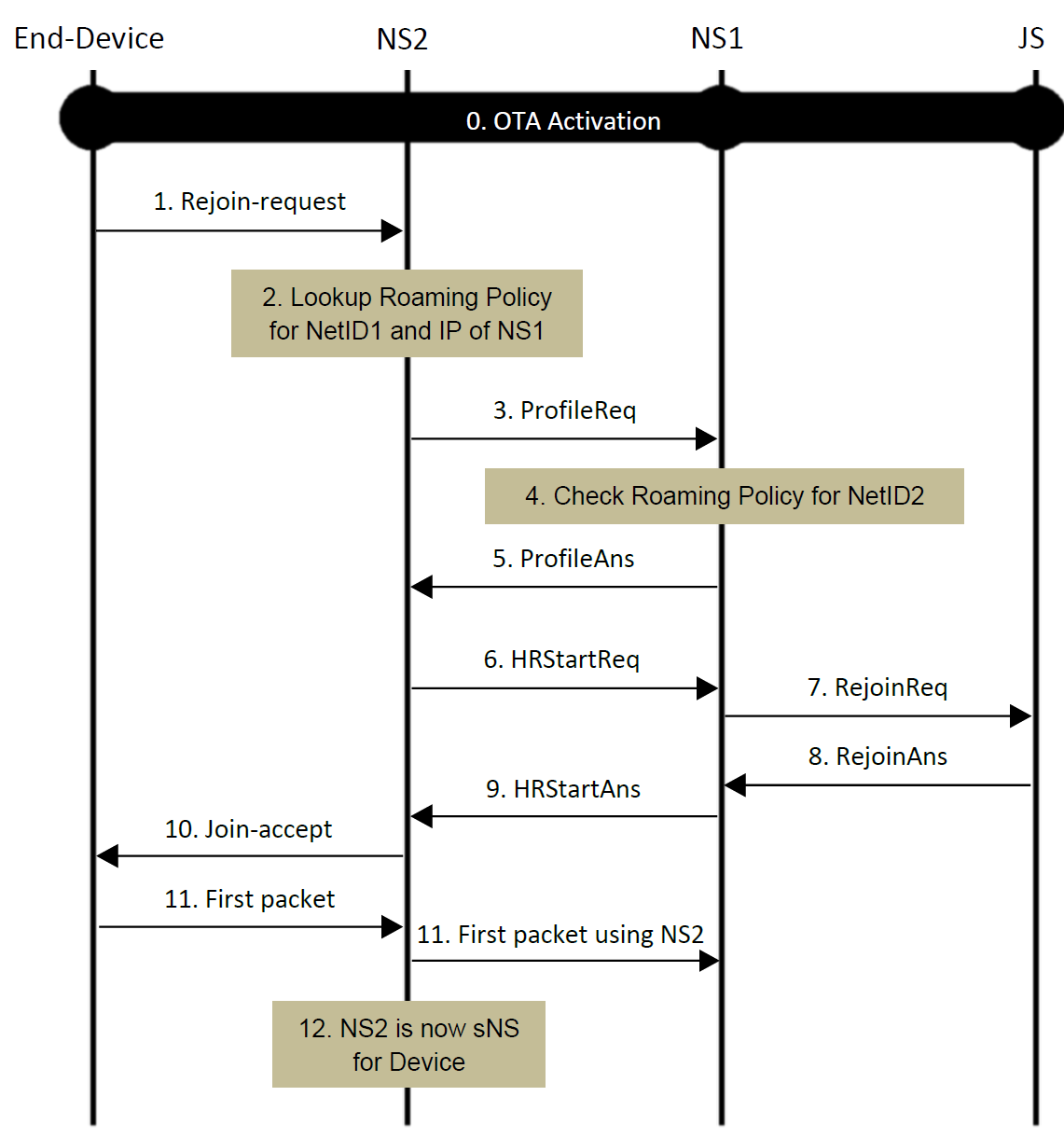
1. Rekeying and DevAddr Reassignment

만약 serving network 서버가 세션키들을 새롭게하고 frame counter들을 재설정하거나 채널정의를 변경없이 end 디바이스에 대해 새로운 DevAddr을 부여히기로 결정했다면, serving network 서버는 RejoinType 2 MAC 명령어와 ForceRejoinReq를 end 디바이스에게 보낸다.

End 디바이스는 serving network 서버로부터 ForceRejoinReq를 받았을 때 Rejoin-request Type 2 메시지를 보낸다.

End 디바이스는 유효한 ForceRejoinReq와 RejoinType 2 serving network 서버로 받지 않는한 Rejoin-request Type 2 메시지를 보내지 않는다.

만약 serving network 서버가 ForceRejoinReq와 RejoinType 2 MAC 명령어를 end 디바이스에게 보내지않았다면 Serving network 서버는 전달받은 Rejoin-Request Type 2를 무시한다.



Rejoin-request Type 2 메시지의 저리과정은 전달받는 network 서버(위그림의 Network 서버2)가 이미 serving network 서버임을 고려하면 **Handover roaming Start**에서 서술한 것처럼 Rejoin-request Type 0의 처리과정과 같다.

만약 end 디바이스가 serving network 서버로부터 ForceRejoinReq와 RejoinType 2 MAC 명령어를 전달받지 않고 세션키들을 새롭게 하거나 frame counter들을 초기화하기로 결정했을 때 end 디바이스는 join-request를 보낸다.

이 절차는 오직 R1.1[LW11] end 디바이스와 network에서 적용된다.

1. Packet Metadata

**UL Packet Metadata**

Lora 시스템에 의해 전달받는 각각의 uplink pakcet들은 무선 수신기에서 얻어진 파라미터들의 집합과 end 디바이스의 Lora 세션의 지역 문맥과 연관된다.

그러한 파라미터들은 uplink 전송을 지원하기 위해 packet payload와 함께 metadata의 형태로 통신 network 요소들 사이에 공유된다.

테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

이 그림은 uplink packet들에 대해 metadata의 세부사항을 설명한다.

Note 1: stateless forwarding network 서버의 경우는, 적어도 두 정보의 요소들의 하나가 출력되야한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Information  Element | Generated  by | Carried over  fNS-sNS interface | Carried over  sNS-hNS interface | Description/notes |
| DevEUI | fNS | Y | Y | 수신된 패킷 또는 local context를 통해 보낸 사람이 사용할 수 있는 경우 포함된다. |
| DevAddr | fNS | Y | Y | 수신된 패킷 또는 local context를 통해 보낸 사람이 사용할 수 있는 경우 포함된다. |
| FPort | sNS | N | Y | sNS는 FRM Payload(PHYPayload가 아님)를 hNS로 전송하므로 누락된 FPort가 별도로 전달된다. |
| FCntDown | sNS | N | Y | 사용 가능한 경우 End Device에 사용된 마지막 downlink application 카운터이다. 32비트 카운터를 사용하는 경우 True 32비트이다.  R1.1을 사용하는 경우 AFCntDown이 발생한다. |
| FCntUp | sNS | N | Y | sNS는 FRMPayload(PHYPayload가 아님)를 hNS로 보내기 때문에 누락된 FCntUp은 32비트 카운터를 사용하면 True 32비트로 별도로 전달된다.  R1.1을 사용하는 경우 AFCntUp를 운반한다. |
| Confirmed | sNS | N | Y | MType이 Confirmed Data Up이면 True로, 그렇지 않으면 False로 설정한다. |
| ADRBit | sNS | N | Y | ADR bit가 설정되어 있으면 True로 설정되고 그렇지 않으면 False로 설정된다. |
| DataRate | fNS | Y | Op | NS가 수신 GW를 제어하여 생성된다. |
| ULFreq | fNS | Y | Op | UL packet의 전송 빈도입니다. NS가 수신 GW를 제어하여 생성한다. |
| Margin | fNS | N | Op | Service Profile에 의해 요청된 경우 보고된다. |
| Battery | fNS | N | Op | Service Profile에 의해 요청된 경우 보고된다. |
| FNSULToken | fNS | Op | N | fNS에 의해 생성된 불투명 값. 이 값은 나중에 다운링크 패킷 전송을 통해 fNS를 지원할 수 있는 보조 매개 변수를 인코딩한다. (Note 1을 봐라) |
| RecvTime | fNS | Y | Y | packet 도착 timestamp (GPS 시간 1초 정밀도)이다. NS가 수신 GW를 제어하여 생성된다. |
| RFRegion | fNS | Y | N | fNS의 무선 지역 |
| GWCnt | fNS | Op | Op | 미리 구성된 시간 초과 기간 내에 동일한 UL packet을 수신한 gateway의 수이다.  NS가 수신 GW를 제어하여 생성된다. |
| GWinfo | fNS | Y | Op | 각 GW에 대한 parameter목록이다. (아래참고) (AntennaID가 있을 때 각 GW antenna에 대해)  각 GW는 동일한 UL packet을 수신한다. 수신 GW를 제어하는 NS에 의해 생성됩니다. fNS가 DL을 보낼 수 있는 경우에만 fNS에 대해 의무적이다. |
| >GWID | fNS | Op | Op | GW 식별자 |
| >AntennaID | fNS | Op | Op | 안테나 식별자 |
| >FIneRecvTime | fNS | Op | Op | RecvTime 이내Nanosec, 아마도 암호화될지도 모른다. |
| >FRTContext | fNS | Op | Op | FineRecvTime context.  포함되면, FinRecvTime은 암호화된다. |
| >RFRegion | fNS | Op | Op | GW의 무선 지역 |
| >RFParamSetID | fNS | Op | Op | GW에서 사용하는 RF parameter 집합의 ID이다. ID 및 관련 RF parameter는 대역 외 메커니즘에 의해 fNS와 sNS 간에 교환된다. |
| >RSSI | fNS | Y | Op | Received signal strength indication |
| >SNR | fNS | Y | Op | Signal-to-noise ratio |
| >Lat | fNS | Op | Op | Latitude(위도) of the GW/antenna |
| >Lon | fNS | Op | Op | Longitude(경도) of the GW/antenna |
| >Alt | fNS | Op | Op | Elevation(고도) of the GW/antenna |
| >ULToken | fNS | Op | Op | GW에 의해 생성된 불투명 값은 나중에 downlink packet 전송을 통해 동일한 GW를 지원할 수 있는 보조 parameter로 인코딩한다. (Note 1을 봐라) |
| >DLAllowed | fNS | Y | N | GW로부터 가능한downlink 전송에 대한 리소스 가용성에 대해 알려준다. |

**DL Packet Metadata**

Lora 시스템에 의해 전달되거나 만들어지는 각각의 downlink packet은 application 서버로부터 얻어지거나 end 디바이스의 Lora 세션의 지역적인 context에 의해 얻어진 parameter들의 집합과 연관이 되어있다.

그러한 parameter들은 downlink 전송을 지원하기위해 packet payload와 함께 metadata의 형태로 communicating network elements 사이에 공유된다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 downlink packet에 대한 metadata의 세부사항을 보여준다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Information element | Generated  by | Carried  over hNS- sNS interface | Carried  over sNS-fNS interfacae | Description/notes |
| DevEUI | hNS | Y | Y |  |
| FPort | hNS | Y | N | hNS FRMPayload를 sNS에게 보낸다,FPort가 개별적으로 오기 때문에.  FPort = 0은 허락되지않는다.  sNS는 Result=InvalidFPort를 돌려줄 것이다. |
| FCntDown | hNS | Y | N | AFCntDown은 R1.1에서 |
| Confirmed | hNS/sNS | Y | N | 부가적으로 전송의 확인을 표시하기위해 사용된다 |
| DLFreq1 | sNS | N | Y | RX1에 대한 전송 주기 |
| DLFreq2 | sNS | N | Y | RX2에 대한 전송 주기 |
| RXDelay1 | sNS | N | Y | RX1에 대한 전송 delay |
| ClassMode | sNS | N | Y | DL에 대한 디바이스 모드 |
| DataRate1 | sNS | N | Y | RX1에 대한 전송률 |
| DataRate2 | sNS | N | Y | RX2에 대한 전송률 |
| FNSULToken | sNS | N | Y | 만약 이용가능하면, fNS에서 전달받은 마지막 FNSULToken의 복사본 |
| GWInfo | sNS | N | Op | 최신의 UL packet을 받은 각각의 GW에서 ULToken parameter들의 리스트 (아래를 봐라)  값들은 최신의 ULMetadata에서 복사된다. |
| >UL Token | sNS | N | Y | 각각의 GW에서 전달받은 ULToken의 복사본.  만약 ULMetadata에서 제공된다면, 그것은 DLMetadata에서 보여질 것이다. |
| HiPriorityFlag | sNS | N | Y | fNS는 최선의 packet 전송을 한다.(예를 들어, 보낼 때 RejoinSetupRequest 명령어를 설정한다. |

1. Profiles

**Device Profile**

디바이스 프로파일은 LoraWAN 무선 접속 서비스를 설정하기위해 network 서버에 필요한 end 디바이스의 capabilities와 boot parameters를 포함한다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

위 그림은 디바이스 프로파일을 포함하고있는 information elements들을 설명한다.

이러한 information element들은 end 디바이스의 제조사에 의해 제공할 것이다.

|  |  |  |
| --- | --- | --- |
| Information element | M/O | Description/notes |
| DeviceProfileID | M | End 디바이스의 parameters의 집합을 위한 유니크한 식별자 |
| SupportsClassB | M | End 디바이스는 ClassB를 지원한다 |
| ClassBTimeout | O | End 디바이스에게 MAC 요청 또는 DL frame 확인에 대한 Maximum delay(만약 classB를 지원하면, 의무사항이다(Mandantory)  [LW104]에서 CLASS\_B\_RESP\_TIMEOUT으로 사용한다. |
| PingSlotPeriod | M | 만약 Class B모드를 지원하면 의무사항이다. |
| PingSlotDR | M | 만약 Class B모드를 지원하면 의무사항이다. |
| PingSlotFreq | P | 만약 Class B모드를 지원하면 의무사항이다. |
| SuppoetsClassC | M | End 디바이스가 Class C를 지원한다. |
| ClassCTimeout | O | End 디바이스에게 MAC 요청 또는 DL frame 확인에 대한 Maximum delay(만약 class C를 지원하면, 의무사항이다(Mandantory)  [LW104]에서 CLASS\_C\_RESP\_TIMEOUT으로 사용한다. |
| MACVersion | M | End 디바이스를 지원하는 LoRaWAN의 버전 |
| RegParamesRevision | M | End 디바이스에 의해 지원되는 Pegional Prameters 문서들의 Revision |
| SuppoortsJoin | M | End 디바이스는 join(OTAA) 또는 not(ABP)를 지원한다. |
| RXDelay1 | O | Class A RX1 지연시간(ABP에서 의무사항이다) |
| RXDROffset1 | O | RX1 data rate offset(ABP에서 의무사항이다) |
| RXDataRate2 | O | RX2 data rate(ABP에서 의무사항이다) |
| RXFreq2 | O | RX2 채널 주파수(ABP에서 의무사항이다) |
| FactoryPresetFreqs | O | 공장 주파수의 리스트(ABP에서 의무사항이다) |
| MaxEIRP | M | End 디바이스에서 지원하는 최대 EIRP |
| MaxDutuCycle | O | End 디바이스에서 지원하는 최대 duty cycle |
| RFRegion | M | 무선 지역 이름 |
| Suppoort32bitFCnt | O | End 디바이스는 32bit FCnt 사용한다.(LoRaWAN 1.0 end 디바이스에서 의무사항이다) |

M/O열의 안의 “M”은 “의무적으로 포함되어야 함” .(봐라 아래의 추가적인 고려사항을)과 “O”는 “선택적으로 포함되어야함”을 가르킨다.

DeviceProfile 대상의 보내는 이는 대역외의 메커니즘을 통해 수신기와 parameter들을 교환할 것이다.

이점을 고려해서, 이 specification의 내부의 메커니즘을 통해 DeviceProfile이 전달될 때. 보내는 이는 오직 DeviceProfileID(예를 들어, 모든 다른 parameter들은 생략되었다 심지어 위 그림에서 의무사항이라고 적혀있는 것 역시)또는 DeviceProfileID와 모든 다른 의무적인 parameter들 둘다(그리고 부가적으로 비의무적인 것들 또한) 포함할 것이다.

보내는이는 일단 parameter들이 DeviceProfileID에 속해있다면 parameter들을 수정하지않을 것이다.

**Service Profile**

ServiceProfile은 Lora 무선 접속 서비스에 대한 설정과 application 서버와 상호작용하기위해 network 서버에 의해 필요한 service profile을 포함한다.

테이블이(가) 표시된 사진

자동 생성된 설명

그림 7은 service Profile에 포함되는 information element들을 표현한다.

|  |  |  |  |
| --- | --- | --- | --- |
| Information element | Carried over hNS-sNS interface | Carried over sNS-fNS interface | Description/note |
| ServiceProfileID | M | M | Service parameter들의 집합을 위한 유니크한 식별자 |
| ULRate | O | N/A | Token bucket이 채워지는 비율, ACKs을 포함하는(packet/h) |
| ULBuketSize | O | N/A | Token bucket의 버스트 크기 |
| ULRatePolicy | O | N/A | ULRate를 넘어갈 때, Drop 또는 Mark |
| DLRate | O | N/A | Token bucket이 채워지는 비율, ACKs을 포함하는(packet/h) |
| DLBucketSize | O | N/A | Token bucket의 버스트 크기 |
| DLRatePolicy | O | N/A | DLRate를 넘어갈 때, Drop 또는 Mark |
| AddGWMetadata | O | O | GW metadata(RSSI, SNR, GW geoloc.. 등등)은 packet을 AS에게 보낼 때 추가된다. |
| DevStatusReqFreq | O | N/A | End 디바이스 상태 요구를 착수시키는 빈도(request/day) |
| ReportDevStatusBattery | O | N/A | End 디바이스의 배터리 단계를 AS에게 알린다. |
| ReportDevStatusMargin | O | N/A | End 디바이스의 차이(?)를 AS에게 알린다 |
| DRMin | O | N/A | 최소한의 허락되는 data rate.  ADR을 이용한다. |
| DRMax | O | N/A | 최대한의 허락되는 data rate.  ADR을 이용한다. |
| ChannelMask | O | N/A | 체널 가리기.  sNS는 복종할 필요가 없다(ex: 유익한?) |
| PRAllowed | O | N/A | Passive Roaming Allowed |
| HRAllowed | O | N/A | Handover Activation Allowed |
| RAAllorwed | O | N/A | Roaing Activation allowed |
| SendLoc | O | O | 정보의 지리적 위치의 생성을 가능하게 한다. |
| LocSolverAuxData | O | O | SendLoc = True일 때, 지리 위치 알고리즘에 의해 필요할 수 있는 보조적인(auxiliary) data |
| AddLocMetadata | O | O | Geolocation-specific ULMetadata 의 추가를 가능하게 한다. |
| TargetPER | O | N/A | Target Packet Error Rate |
| MinGWDiversity | O | N/A | 받는 GW들의 최소한의 수(유익함?) |

M/O열의 안의 “M”은 “의무적으로 포함되어야 함” .(봐라 아래의 추가적인 고려사항을)과 “O”는 “선택적으로 포함되어야함” 그리고 “N/A”는 “해당하지않음”을 가르킨다.

만약 부가적인 정보 parametger를 보내지않는다면, 연관된 설정들은 수신받는 network 서버의 재량에 결정된다.

Service Profile 대상의 보내는 이는 대역 외의 메커니즘을 이용한 수신기와 parameter들을 교체할지도 모른다.

이점을 고려해서, serviceProfile이 이 specification 안의 메커니즘을 통해 도달한다면, 보내는 이는 오직 ServiceProfileID(ex: 모든 다른 parameter들은 심지어 위그림에서 의무적이라고 표시된 prameter들도 생략된다.) 또는 ServiceProfile과 모든 다른 의무적인 parameter들(그리고 부가적으로 비의무적인 것들 역시) 둘 다 포함한다.

보내는 이는 일단 그들이 ServiceProfileID의 범위안에 있다면 parameter들을 수정하지않을 것이다.

**Routing Profile**

Routing Profile은 application 서버와 함께 data-plane을 설정하기위해 network 서버가 필요하는 정보를 포함한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 그림은 Routing Profile에 포함되어있는 information element들을 보여준다.

|  |  |  |
| --- | --- | --- |
| Information element | M/O | Desciption/notes |
| RoutingProfileID | M | Routing parameter들의 집합의 유니크한 식별자 |
| AS-ID | M | Application 서버의 ID |

M/O열의 안의 “M”은 “의무적으로 포함되어야 함” .(봐라 아래의 추가적인 고려사항을)과 “O”는 “선택적으로 포함되어야함” 을 가르킨다.

RoutingProfile 대상의 보내는 이는 대역 외의 메커니즘을 이용한 수신기와 parameter들을 교체할지도 모른다.

이점을 고려해서, RoutingProfile이 이 specification안의 메커니즘을 통해 배달되었을 때, 보내는이는 오직 RoutingProfileID(ex: 모든 다른 parameter들은 심지어 위 그림에서 의무적이라고 표시된 prameter도 무시된다.)를 포함하거나 RoutingProfileID 그리고 모든 다른 의무적인 parameter들(그리고 부가적으로 비 의무적인 것역시)을 포함한다.

보내는 이는 일단 그들이 RoutingProfileID의 범위안에 있다면 parameter들을 수정하지 않을 것이다.

1. Usage Data Records

**Network Activation Record**

Network activation Record는 Roaming Activation을 하는 end 디바이스를 주시하기위해 이용된다.

Roaming Activation 절차가 일어날 때, network 서버는 한달동안 적어도 한번 end 디바이스를 활성화시키는 다른 network 서버의 각각의 ServiceProfileID에 대해 월별로 Network Record를 생성하고 다른 network 서버에서 각 end 디바이스의 activation과 deactivation에 대한 Network Activation Records을 제공한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이 그림은 network activation Record의 세부사항을 설명한다.

|  |  |
| --- | --- |
| Information element | Description/notes |
| NSID | Roaming 상대 network 서버의 ID |
| NetID | Roaming 상대 network 서버의 NetID |
| ServiceProfileID | Service Profile ID |
| IndividualRecord | 만약 이것이 개별적인 (de)activation record라면 가리킨다  (한 달 동안 active인 end 디바이스의 누적 record가 아니다.) |
| TotalActiveDevices | 한 달 동안 active 됐었던 End 디바이스의 숫자.  만약 이것이 누적 record라면 포함된다. |
| DevEUI | (de)activation을 수행했던 End 디바이스의 DevEUI.  만약 이것이 (de)activation의 이벤트에 대해 IndividualRecord라면 포함한다. |
| ActivationTime | Activation의 일/시간.  만약 이것이 activation 이벤트에 대해 IndividualRecord라면 포함한다. |
| DeactivationTime | Deactivation의 일/시간.  만약 이것이 deactivation 이벤트에 대해 IndividualRecord라면 포함한다. |

**Network Traffic Record**

Network Traffic Record는 roaming end 디바이스에 대해 제공된 traffic의 양에 대해 기록해두기 위해 이용된다.

Roaming 을 허락한 network 서버는 월마다 각각의 roaming type(passive/handover)에 대해 적어도 한번 자신의 network으로 end 디바이스 roaming이 있는 다른 network 서버의 각 ServiceProfileID 아래에 Network Traffic Record를 생성한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 그림은 Network Traffic Record의 세부사항을 보여준다.

|  |  |
| --- | --- |
| Information element | Description/notes |
| NSID | Roaming 상대방의 network 서버의 ID |
| NetID | Roaming 상대방의 network 서버의 NetID |
| ServiceProfileID | Service Profile ID |
| RoamingType | Passsive Roaming or Handover Roaming |
| TotalULPackets | Uplink packet의 수 |
| TotalDLPackets | Downlink packet의 수 |
| TotalOutProfileULPackets | ULRate를 초과하나 ULPolicy당 어떠한 방법으로 전달되는 Uplink packet의 수 |
| TotalOutProfileDLPackets | DLRate를 초과하나 DLRatePolicy당 어떠한 방법으로 전달되는 downlink packet의 수 |
| TotalULBytes | Uplink byte의 총량 |
| TotalDLBytes | Downlink byte의 총량 |
| TotalOutProfileULBytes | Service Profile 밖으로 떨어진 uplink Byte의 총량 |
| TotalOutProfileDLBytes | Service Profile 밖으로 떨어진 downlink Byte의 총량 |
| TotalLoc | 보고된 지리좌표의 수 |

Packet 과 payload counter는 오직 사용자가 만들어낸 traffic에 기반한다.

Payload counter는 FRMPayload 필드의 크기에 기반한다.

1. JoinEUI and NetID Resolution

Network 서버가 join-request또는 Rejoin-request 메시지를 받았을 때, network 서버는 첫번째로 DevEUI 그리고 JoinEUI를 연결함으로써 join 서버의 IP 주소를 확인(해결)하고 만약 이 해결이 실패했다면, network 서버는 오직 JoinEUI를 사용함으로써 확인(해결)할 것이다.

유사하게 NetID값은 network 서버가 Rejoin-request 메시지로 받았을 때 연결된 network 서버의 IP 주소를 확인(해결)할 것이다.

주소확인의 2가지 타입은 DNS를 사용함으로써 수행된다.

몇몇 단체들은 network를 가동하지않은채 join 서버를 가동해야한다, 그래서 join 서버 확인 메커니즘은 NetID를 할당없이 활동할 수 있음을 유의해야한다.

**NetID and JoinEUI Conversion for the DNS Configuration**

Lora Alliance는 JOINEUIS.lorawan.net과 NETIDS.lorawan.net에 뿌리를 두는 Join 서버들과 NetIDs들을 확인하기 위해 두 전용의 서브도메인을 각각 설립하고 운영할 것이다.

24bit NetID, 예를 들어 :5292746”, 10진수 형태는 16진수 형태로 나타난다.

: 0x60050A

각 16진수 자리는 한입(nibble?)이고 인코딩 순서는 높은 nibble순서에서 낮은 nibble순서를 따른다.

접미사로써 “NETIDS.lorawan.net” 도메인 이름과 인코딩된 NetID의 16진수 변환과 연결하는 것은 FQDN(Fully Qualified Domain Name)안의 다음과 같은 결과일 것이다.

: 60050a.NETIDS.lorawan.net

64bit joinEUI(IEEE EUI-64)는 다음과 같은 16진법 형태로 표현한다.

: 0x00005E100000002F

유사하게, 64bit DevEUI는 다음과 같은 16진법형태로 표현된다.

: 0x0102030405060708

각 16진수 자리는 nibble이다(?) 그리고 인코딩 순서는 낮은 nibble순서에서 높은 nibble순서를 따른다.

nibble들이 역순으로 인코딩되고 종점은 각 16진수 사이에 추가된다.

기본적으로, 도메인 이름 “JOINEUIS.lorawan.net”을 접미사로 인코딩된 JoinEUI의 16진수 변환에게 연결하는 것은 FQDN의 결과일 것이다. 다음과 같이.

: f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0.JOINEUIS.lorawan.net

이 경우, 여기서 동일한 JoinEUI가 다른 join 서버들을 가리켜야하는 경우는 DevEUI는 위의 JoinEUI의 변환과 연결된다.(역순과 점은 각 16진수 DevEUI값에 더해진다.)

: 8.0.7.0.6.0.5.0.4.0.3.0.2.0.1.f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0.JOINEUIS.lorawan.net

Note: JoinEUIs 또는 DevEUI와 joinEUI의 연결은 역순으로 DNS에서 hierarchical provisioning의 이점인 영향을 주기위해 인코딩된다.

동일한 JoinEUI가 여러 join 서버로 확인 되는 경우 DNS에서 Provisioning은 DNS 운영자의 전문지식에 맡겨야 한다.

DNS 운영자는 그런 경우의 제한을 설명하고 provisioning은 적절한 테스팅 이후에 될 것이다.

이것들은 사건별로 처리가 된다.

**NetID and JoinEUI Provisioning**

NetID는 “NETIDS.lorawan.net”의 안에서 provision될 것이다.

NetID에 해당하는 자원은 다른 DNS 자원 record 형태로 provision 될 것이다.

(예를 들어 network 서버, CNAME,A,AAAA)

60050a.NETIDS.lorawan.net IN CNAME example.com.

60050a.NETIDS.lorawan.net IN A 192.0.2.0

유사하게, 다음과 같은 요구사항에 따라 “JOINEUIS.lorawan.net”의 구역은 다른 DNS 자원 record 형식으로 provision될 수 있다.

오직 joinEUI와 함께: f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0. JOINEUIS.lorawan.net. IN CNAME example.net

8.0.7.0.6.0.5.0.4.0.3.0.2.0.1.f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0. JOINEUIS.lorawan.net. IN AAAA 2001:db8:85a3::8a2e:370:7334

운영적 효율성의 경우에는, 연결은 DNS의 Wildcard [RFC4592]의 특징을 사용함으로써 될 것이다.

\*.0.4.0.3.0.2.0.1.f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0.JOINEUIS.lorawan.net. IN CNAME example.com.

**NetID Resolution**

Input parameter는 end 디바이스에서 방문했던 network의 network 서버로 보낸 Rejoin-request 메시지에 전달되는 24bit NetID이다.

Visited network 서버는 Rejoin-request 메시지안에 전달받은 NetID를 **NetID and JoinEUI Conversion for the DNS Configuration**에서 서술한 DNS query로 변환 할 것이다.

Network 서버는 home network 서버이 IP 주소를 확인하기 위해 DNS 변환기를 사용할 것이다.

**JoinEUI and DevEUI-JoinEUI Concetanation Resolution**

Input parameter는 64bit JoinEUI 또는 DevEUI의 연결이고 JoinEUI는 end 디바이스에서 Home network 서버의 network 서버로 보내진 Join-request 메시지에게 전달되거나 end 디바이스에서 Visited network 서버의 network 서버로 보내진 Rejoin-request 메시지에 전달된다.

전달받는 Network 서버는 첫번째로 DevEUI와 JoinEUI를 연결시킨 DNS query를 만들 것이다. 그리고 만약 해결에 실패했다면, 그것은 해결하기위 JoinEUI를 사용함으로써 페일 백한다.(**NetID and JoinEUI Conversion for the DNS Configuration에 설명되어있다.)**

Network 서버는 Join 서버의 IP 주소를 확인하기위해 DNS 변환기를 사용할 것이다.

1. Transport Layer

LoRaWAN backend 인터페이스들은 제어신호들과 데이터 packet들을 전달하기위해 Join 서버, network 서버, 그리고 application 서버와 같은 network 요소사이에 상호연결을 수반한다.

따라오는 network 인터페이스들은 현재 specification의 범위 안이다.

1. Application 서버 – Join 서버(optnal)
2. Join 서버 – Network 서버
3. Network 서버 – Network 서버

JoinEUI는 join 서버를 구별한다, 반면 network 서버는 NetID에 의해 구별된다.

다수의 JoinEUI들은 같은 join 서버를 구별할지 도 모른다.

JoinEUI와 NetID둘다 DNS를 사용해서 각각의 서버들의 IP주소로 확인할 수 있다.

**Network 요소들은 상호 end-point 인증, 무결성과 replay 보호 그리고 서로 대화를 할 때 진실성을 제공할 수 있는 보안해결책에 의존하고있다.**

이러한 특성을 달성하기 위한 메커니즘의 선택은 전개에 남아있다

(IPsec VPN(IPSec VPN은 **IP계층**에서의 안전한 전송/통신을 위해서 사용되는 **3계층 터널링 프로토콜**입니다.), HTTPS, physical security, 등등)

Network 요소들은 HTTP 1.1[RFC2616]을 사용하고 JSON을 사용해 payload를 인코딩한다.

메시지(신호 또는 데이터)를 양방향 전달을 지원하기 위해선, HTTP의 연결 쌍은 두 end-point 사이에서 설정되어야한다.

각 end-point는 초기화되고 다른 end-point와 HTTP 연결로 유지된다.

HTTP end-point들은 지속적인 연결을 사용할 것이다.

1. Key Transport Security

LoRa 세션동안 몇몇 번 키들은 서버들 사이에서 교체 되어야한다.(사례에서 join 서버 – application 서버, join 서버 – network 서버 또는 network 서버 – network 서버 인터페이스들)

이러한 키들을 안전하게 전송하기 위해서는, RFC 3394에 정의된 래핑 과정을 따르면서 Key Encryption Keys(KEK)은 그들의 암호화에 사용될 것이다.

또한, 각 Key Encryption Key는 Key Encryption Key Lebel(KEKLabel)과 RFC 3394에 정의된 래핑 알고리즘과 연관 되어있어 래핑 제 동작 동안 올바른 키와 올바른 알고리즘을 선택하는 것을 선택할 수 있다.

KEK의 집합, 연관된 KEKLabels 그리고 알고리즘은 이 specification의 부분은 아닌 오프라인 프로세스동안 서버들 사이에서 생성되고 교체된다.

서버는 2가지 종류로 구성된다: 키 요청자와 키 발신자.

키를 감싸거나 감싸지 않는 결정은 항상 키를 전달하는 책임을 가진 entity에 의해 이루어진다.(ex: 키 발신자)

텍스트이(가) 표시된 사진

자동 생성된 설명

위 그림은 키들을 감싸는데 사용되는 KeyEnvelope Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Information element | M/O | Descript/notes |
| KEKLabel | O | 이 label은 AESKey를 해제하기위해 사용될 키를 구별한다.  만약 이 값이 없으면 AESKey가 명확하게 전송됨을 의미한다. |
| AESKey | M | 만약 KEKLabel 필드가 있으면, AESKey는 RFC3394(AES)로 감싸진 키를 나른다.  만약 KEKLabel 필드가 없으면, AESKey는 명확히 Key를 나른다. |

1. Messages and Payloads

**Encoding**

HTTP 는 backend request, 대답, 그리고 알림 메시지를 전달하기 위해 transport layer로써 사용된다.(예를 들어: JoinReq, JoinAns, ErrorNotif).

다음 인터페이스들은 HTTP request를 통해 backend request와 대답 메시지를 나르는 반면 간단히 전송을 인정하기 위해 HTTP Response들을 사용한다.

Ex) forwarding network 서버 – serving network 서버, serving network 서버 – home network 서버.

전송이 성공했을 때, backend 응답 결과와 상관없이, HTTP Response는 응답의 HTTP 2XX state-code를 사용해야한다.

오직 전송이 성공적이지 않을 때(ex: 잘못된(malformed) HTTP request), HTTP 응답은 HTTP 4XX 또는 5XX 응답의 Status-Code 사용해야한다.

그 경우 backend 요청은 응답되지않을 것이다.

다음 인터페이스들은 HTTP request들을 통해 backend request 메시지들을 나르고 반면 backend answer 메시지들은HTTP Response 또는 하위시퀀스 HTTP Request를 통해 날라질 것이다:home network 서버 – join 서버, visited network 서버 – join 서버, application 서버 – join 서버.

각 Backend 상대를 위한 Join 서버가 사용하는 방법은 범위 밖의 결정이다.

알림 메시지들은 한방향 메시지이다.

그들은 HTTP Requests을 통해 전달되는 반면 돌아온 HTTP Response들은 간단히 그들의 배달은 인정한다.

Network 요소들은 요청, 대답,그리고 알림 메시지를 전송하기위해 JSON data 형태를 사용한다.

Network 요소가 HTTP request에서 다른 network 요소에게 보낼 메시지가 있을 때, network 요소는 HTTP POST Request을 해당(target) URL에 대해 생성한다.

해당(target) URL은 서로 인터페이스하는 두 network 요소들 사이에 동의된 parameter로 구성되어있다.

예를 들어, 주어진 network 서버에서 join 서버의 대상 URL은 <https://js.lora_operrator.com>이 돨 수 있다.

왜냐하면 주어진 network 서버가 같은 시간에 다수의 역활들을 제공할 것이기 때문에(forwarding network 서버, serving network 서버 그리고 home network 서버로 행동) 요청의 발신자는 해당(target) URL에 아래의 확장들 중 하나를 추가함으로써 예정된 해당(target) network 서버의 수신자 가리킨다.: “/fns”, ”/sns”, ”/hns”

서버의 home network 서버에게 요청을 보내기위한 예시 해당(target) URL은 <https://ns.lora_operator.com/hns>.

Roaming Activation의 경우, visited network 서버의 역할은 home network 서버로부터 ProfileAns 메시지를 받을 때까지 결정하는 것이 아니다.

그 결정 이전에 전송된 Backend answer 메시지들(구체적으로, HomeNSAns 그리고 ProfileAns 메시지들)의 송신자는 visited network 서버의 forwarding network 서버의 URL을 사용할 것이다.

HTTP는 request, answer 그리고 알림 메시지들을 다양한 object들과 JSON 암호화된 payload로 나른다.

메시지들은 “application/json” Media 타입(HTTP content-Type 헤더 필드)을 사용될 것이다.

주어진 메시들에 포함될 것인 object들의 이름들은 자세한 메시지의 흐름들을 설명하는 구역에서 제공된다.

각 object 타입의 인코딩은 **“Error Notification Messages**”에서 제공된다.

각 메시지는 이 specification의 설명으로 인해 값이 “1.1”으로 정해진 ProtocolVersion object, 그 메시지에 대한 필요한 행동을 정의한 MessageType object, SenderID 그리고 ReceiverID를 포함한다.

메시지의 발신자는 발신자가 network 서버 또는 join 서버 또는 application 서버 각각에 따라 SenderID를 NetID, JoinEUI 또는 Application 서버-ID 로 설정할 것이다.

간단하게, 메시지의 발신자는 ReceverID를 수신자가 network 서버 또는 join 서버 또는 application 서버 각각에 따라 의도된 수신자의 NetID, JoinEUI, 또는 Application 서버-ID로 설정할 것이다.

만약 network 서버라면, 발신자는 SenderNS를 포함하고 만약 수신자가 network 서버라면 ReceiverNSID를 포함한다.

Network 요소들이 전달받은 메시지들과 보류(pending)중인 요청/응답 메시지와 일치시킬 수 있도록 하기 위해 TransactionID가 사용된다.

Request 메시지의 발신자는 발신자의 재량(discretion)으로 값들이 설정되는 TransactionID를 메시지안에 포함한다.

응답이나 알림 메시지의 발신자는 같은 응답 또는 알림 메시지에 쏴졌던 메시지에서 받았던 TransactionID를 포함할 것이다.

만약 network 요소가 TransactionID 값과 요청 또는 응답이 없는 응답이나 알림 메시지를 받았다면, network 요소는 받은 메시지를 무시한다.

만약 전달받은 메시지의 ProtocolVersion이 “1.0” 또는 “1.1”로 설정되었지 않다면, 수신 network 요소는 Result = IncalidProtocolVersion을 가지는 메시지를 돌려준다.

만약 전달받은 메시지의 SenderID 또는 ReceiverID를 수신 network 요소에 대해 모른다면, Result = UnknownSender 또는 UnknownReceiver를 가지는 메시지를 돌려준다.

만약 MessageType을 수신 network 요소에 알려지지않은 경우, Result = MalformedRequest 을 가지는 메시지와 동일한 MessageType을 돌려준다.

만약 해당(target) network 요소가 같은 end 디바이스와 연관된 각 서브시퀀스 메시들에 대해 ReceiverToken안에 같은 값을 반영할 것을 예상한다면 Network 요소는 아마 SenderToken을 메시지 안에 포함할 것이다.

Serving network 서버는 stateless forwarding network 서버와 통신할 때 SenderToken을 보내지않을 것이다 forwarding nertwork 서버가 그 token을 저장할 수 없기에.

만약 같은 end 디바이스에 대해 해당(target) network 요소로부터 SenderToken을 전달 받았다면 Network 요소는 그의 메시지안에 ReceiverToken을 포함할 것이다.

이 경우 network 요소는 전달받은 SenderToken의 값을 전송된 ReceverToken에 복사할 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

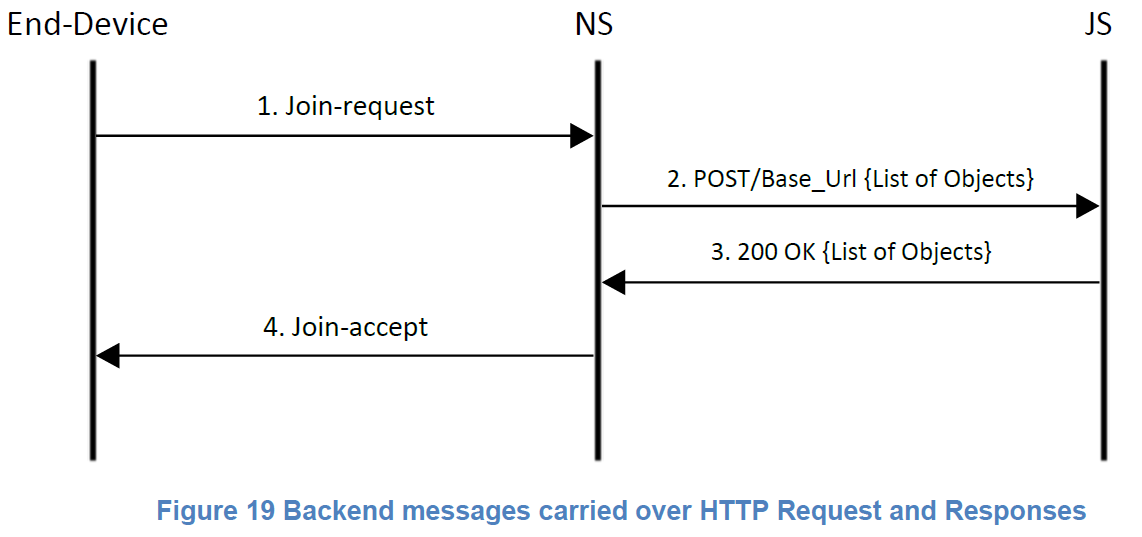


그림 18과 19는 home Procedure에서 OTA activation를 위한 HTTP 메시지의 두 흐름의 변형를 표현한다.

이러한 그림들이 HTTP의 세부사항을 보여주는 반면 이 문서의 나머지 그림은 오직 backend 메시지를 표현한다.(ex: backend 메시지를 payload로 전달하지 않는 한 HTTP Response들은 보여주지 않는다.)

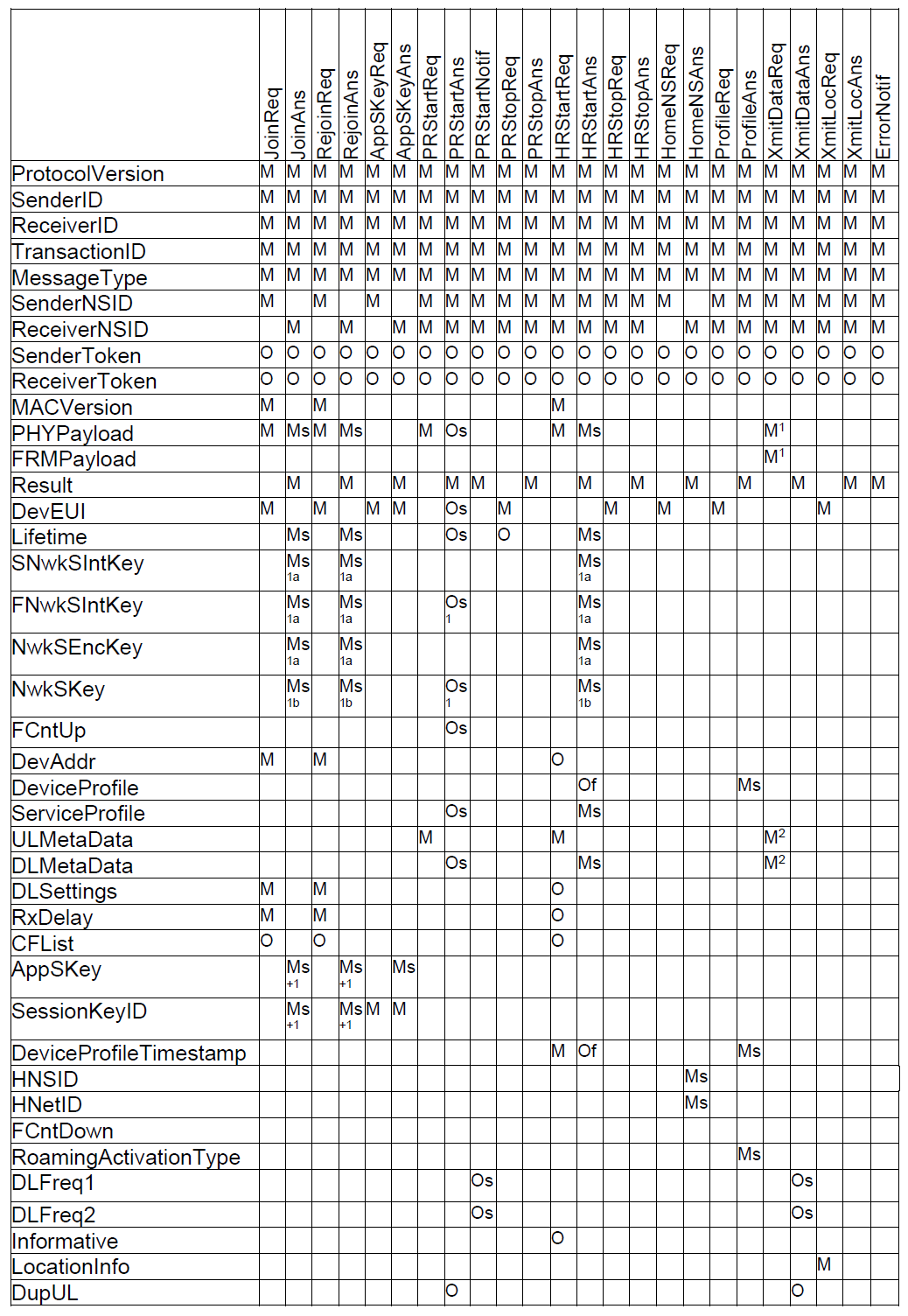
**Backend Message Types**

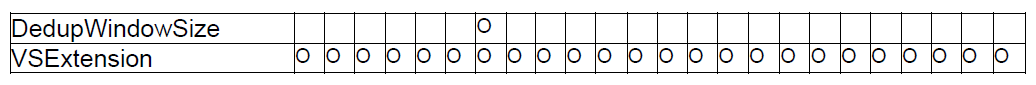
텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

위 사진은 해당되는 경우, backend 메시지의 타입들의 리스트를 쌍으로 제공한다.

메시지의 타입 이름들은 대소문자(case-sensitive)를 구분한다.





만약 위 그림과 연관된 절차의 설명 사이에 불일치(discrepancy)가 항상 일어난다면 (연관된 절차) 후자가 우선된다.

다음 표기법들은 사용된다 위 그림에서

M: 의무적

O: 부가적

Ms: Result = Success일 때, 의무적

Mf: Result = Fail일 때, 의무적

Os: Result = Success일 때, 부가적

Of: Result = Fail일 때, 부가적

MX: 같은 값 X로 표시된 두object(그룹) 중 하나를 정확히 포함해야 의무적  
MXY를 보였을 때, 같은 값 Y로 표시된 object들은 그룹으로써 고려된다.

M+x: 같은 값 X로 표시된 두 object 중 적어도 하나를 포함해야 의무적

빈칸은 object가 지정된 메시지와 함께 사용되지 않음을 가리킨다.

**Error Notification Message**

ErrorNotif는 유효 하지 않은 응답 메시지(ex: 필수적인 object 누락 또는 잘못된 컨텐츠를 가지는 object, 알수없는 SenderID/ReceiverID 등) 에 대답으로 생성된 한 방향 알림 메시지로 써 정의된다.

잘못된 응답 메시지의 받는 사람은 Result 값이 성공이 아닌 ErrorNotif 메시지를 응답 메시지의 송신자에게 보낸다.

**Data Types**

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 이 specification에 정의된 다양한 메시지 payload들에 JSON object 세부사항을 제공한다.

이 specification에 정의된 한 object가 2016년에 LoRaWAN specification에서 정의된 parameter(ex: DevEUI, SNwkSIntKey 등)에 해당한다면, 그 정의서 안의 parameter의 세부사항은 역시 이 specification에 해당하는 object 값들에 적용된다. (ex: DevEUI는 64bit, SNwkSIntKey는 128bit)

VSExtension(Vendor-Specific Extension)으로 이름 지어진 Object는 구체적인 배포 시나리오가 필요한 서버 간에 독점(proprietary)object들을 전달하는 것을 허락한다.

Content의 정의는 구체적인 구현은 남겨두었다.

만약 이것이 서버에 의해 인지된다면, 서버는 받은 VSExtension Object를 처리하고 그렇지않으면 버린다.

String 타입의 JSON object 발신자는 한 byte를 사용해서 각 character 값을 암호화할 것이다.

|  |  |  |
| --- | --- | --- |
| Object Name | Value Type | Notes |
| ProtocolVersion | String | Backend specification의 버전ex)”1.1” |
| SenderID | String | NetID 또는 JoinEUI를 포함하는 경우 ASCII 형식16진수 표현.  AS-ID의 경우는 ASCII String(최대 128 characters) |
| ReceiverID | String | NetID 또는 JoinEUI를 포함하는 경우 ASCII 형식16진수 표현.  AS-ID의 경우는 ASCII String(최대 128 characters) |
| TransactionID | Number | 32bit 값 |
| MessageType | String | Table 12의 값들의 String 표현  (ex: JoinReq) |
| SenderNSID | String | ASCII 형식16진수 표현. |
| ReceiverNSID | String | ASCII 형식16진수 표현. |
| SenderToken | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| ReceiverToken | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| PHYPayload | String | ASCII 형식16진수 표현. |
| FRMPayload | String | ASCII 형식16진수 표현. |
| Result | Object | Table 15 보기 |
| DevEUI | String | ASCII 형식16진수 표현. |
| Lifetime | Number | 단위: 초 |
| SNwkSIntKey | Object | Table 16 보기 |
| FNwkSIntKey | Object | Table 16 보기 |
| NwkSEncKey | Object | Table 16 보기 |
| NwkSKey | Object | Table 16 보기 |
| DevAddr | String | ASCII 형식16진수 표현. |
| HNSID | String | ASCII 형식16진수 표현. |
| HNetID | String | ASCII 형식16진수 표현. |
| DeviceProfile | Object | Table 17 보기 |
| ServiceProfile | Object | Table 18 보기 |
| RoutingProfile | Object | Table 19 보기 |
| ULMetaData | Object | Table 20 보기 |
| DLMetaData | Object | Table 22 보기 |
| DLSettings | String | ASCII 형식16진수 표현. |
| RxDelay | Number |  |
| CFList | String | ASCII 형식16진수 표현. |
| AppSKey | Object | Table 16 보기 |
| SessionKeyID | String | ASCII 형식16진수 표현.  (최대 16 characters) |
| DeviceProfileTimestamp | String | 마지막 Device Profile 변화의 Timestamp  (ISO 8601) |
| RoamingActivationType | String | 받아들일 수 있는 값:”Passive”, ”Handover” |
| Informative | Boolean | 항상 True로 설정 |
| LocationInfo | Object | Table 23 보기 |
| DupUL | Boolean | 항상 True로 설정 |
| DedupWindowSize | Number | 단위: 밀리초 |
| VSExtension | Object | Table 24 보기 |

16진수 ASCII 프린트가능한 값 표현은 “0x”로 시작하고 대문자 또는 소문자를 사용할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 그림은 Result Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| ResultCode | String | “성공” 또는 Table 25에 정의된 오류 문장들 중 하나 |
| Description | String | ResultCode와 연관된 세부적인 정보(부가적, 최대 128 characters) |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 KeyEnvelope Object의 세부사항을 제공한다.

이 object 형식은 SNwkSIntKey, FNwkSIntKey, NwkSEncKey, NwkSKey, 그리고 AppSKey Object에 사용된다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| KEKLabel | String | 최대 16 characters |
| AESKey | String | ASCII 형식16진수 표현. |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 DeviceProfile Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| DeviceProfileID | String | 최대 64 character |
| SupportsClassB | Boolean |  |
| ClassBTimeout | Number | 단위: 초 |
| PingSlotPeriod | Number |  |
| PingSlotDR | Number |  |
| SuppoetsClassC | Boolean |  |
| ClassCTimeout | Number | 단위: 초 |
| MACVersion | String | Ex) “1.0.2”[LW102] |
| RegParamsRevision | String | Ex) “B” [RP102B] |
| RXDelay1 | Number |  |
| RxDROffset1 | Number |  |
| RxDataRate2 | Number | Ex)(DR 0): 0.  Regional Parameters 문서의 Data rate table을 봐라 |
| RXFreq2 | Number | Frequency의 값, ex) 868.10 |
| FactortyPresetFreqs | Array of Numbers |  |
| MaxEIRP | Number | dBM 안에 |
| MaxDutyCycle | Number | Ex) 0.10는 10%를 가리킨다. |
| SupportsJoin | Boolean |  |
| RFRegion | String | Note 2를 봐라 |
| Suppoets32bitFCnt | Boolean |  |

Note2: RF region의 이름은(ex: “EU868”, “US902”).

유효한 값들은 각 지역을 위해 Regional Paraments 문서에 정의된 RFRegion parameter들에 의해 제공된다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 ServiceProfile Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| ServiceProfileID | String | 최대 64 characters |
| ULRate | Number |  |
| ULBucketSize | Number |  |
| ULRatePolicy | String | 받아들일 수 있는 값: “Drop”, “Mark” |
| DLRate | Number |  |
| DLBuketSize | Number |  |
| DLRatePolicy | String | 받아들일 수 있는 값: “Drop”, “Mark” |
| AddGWMetadata | Boolean |  |
| DevStatusReqFreq | Number | 단위: 요청-당-날 |
| ReportDevStatusBatery | Boolean |  |
| ReportDevStatusMargin | Boolean |  |
| DRMin | Number |  |
| DRMax | Number |  |
| ChannelMask | String | ASCII 형식16진수 표현. |
| PRAllowed | Boolean |  |
| HRAllowed | Boolean |  |
| RAAllowed | Boolean |  |
| SendLoc | Boolean |  |
| LocSolverAuxData | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| AddLocMetadata | Boolean |  |
| TargetPER | Number | Ex) 0.10는 10%를 가리킨다 |
| MinGWDiversity | Number |  |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 RoutingProfile Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| RoutingProfileID | String | 최대 64 characters |
| AS-ID | String | IP 주소, DNS 이름 등의 값이 될 수 있다.  (최대 128 characters) |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 ULMeataData Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| DevEUI | String | ASCII 형식16진수 표현,빅 엔디언, 구분자 없음. |
| DevAddr | String | ASCII 형식16진수 표현. |
| FPort | Number | 정수 |
| FCntDown | Number | 정수 |
| FCntUp | Number | 정수 |
| Confirmed | Boolean |  |
| DataRate | Number | Regional Parameters 문서의 Data rate table을 봐라 |
| ULFreq | Number | Float(MHz) |
| Margin | Number | DevStatusAns에서 End 디바이스에 의해 보고되는 정수 값 |
| Battery | Number | DevStatusAns에서 End 디바이스에 의해 보고되는 정수 값 |
| FNSULToken | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| RecvTime | String | ISO 8601을 사용 |
| RFRegin | String | Note 2(위)를 봐라 |
| GWCnt | Number | 정수 |
| GWInfo | Array of GWInfoElement Objects | Table 21을 봐라 |

**테이블이(가) 표시된 사진

자동 생성된 설명**

위 그림은 GWInfoElement Object의 세부설명을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| GWID | String | ASCII 형식16진수 표현.  (note 3을 봐라) |
| AntennaID | Number |  |
| FinRecvTime | Number |  |
| FRTContext | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| RFRegion | String | Note 2를 봐라(위의) |
| RFParamSetID | String |  |
| RSSI | Number | Signed integer, 단위: dBm |
| SNR | Number | 단위: dB |
| Lat | Number | 단위: DD, WGS84에 기반 |
| Lon | Number | 단위: DD, WGS84에 기반 |
| Alt | Number | 단위: 미터, WGS84에 기반 |
| ULToken | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| DLAllowed | Boolean |  |

Note 3: Class B 비콘들은 gateway를 확인하기위해 오직 24bit 값들만 전달할 수 있다.

Network가 이 값들을 나르기로 했을 때 GWID의 24LSB들은 비콘 payload에서 사용될 수 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 DLMetaData Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| DevEUI | String | ASCII 형식16진수 표현. |
| FPort | Number |  |
| FCntDown | Number |  |
| Conformed | Boolean |  |
| DLFreq1 | Number | 적어도 DLFreq1 또는 DLFreq2가 있어야 한다. |
| DLFreq2 | Number | 적어도 DLFreq1 또는 DLFreq2가 있어야 한다. |
| RxDelay1 | Number |  |
| ClassMode | String | 오직 “A” 그리고 “C” 값만 지원된다. |
| DataRate1 | Number | 오직 LFreq1이 있을 때 존재한다. |
| DataRate2 | Number | 오직 LFreq2이 있을 때 존재한다. |
| FNSULToken | String | ASCII 형식16진수 표현.  (최대 512 characters) |
| GWInfo | Array of GWInfoElement Objects | Table 21을 봐라 |
| HIProrityFlag | Boolean |  |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 LocationInfo Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| LocTime | String | ISO 8601을 사용 |
| Lat | Number | 단위: DD, WGS84에 기반 |
| Lon | Number | 단위: DD, WGS84에 기반 |
| Alt | Number | 단위: 미터, WGS84에 기반 |
| LocRadius | Number | 수평 오차, 단위: 미터 |
| AltRadius | Number | 수직 오차, 단위 미터 |
| FCntUp | Number | 계산에 사용된 가장 최근FCntUp |

테이블이(가) 표시된 사진

자동 생성된 설명

위 그림은 VSExtension Object의 세부사항을 제공한다.

|  |  |  |
| --- | --- | --- |
| Object name | Value Type | Notes |
| VenderID | String | 공급업체의 OUI, ASCII 형식16진수 표현  (최대 10 characters) |
| Object | Opaque | Object의 본질은 정의되지 않음 |

**Result Codes**

텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 그림은 Result Object에 제공될수있는 값들의 리스트를 제공한다.

|  |  |
| --- | --- |
| Value | Description |
| “Success” | 성공, ex) 요청이 승인되었다. |
| “NoActtion” | Home network 서버에서 현재 serving network 서버의 Rejoin-request 수신 획인하는데 사용된다. |
| “MICFailed” | MIC 검증이 실패했다. |
| “FrameReplayed” | 전달받은 frame이 반복이다.  (DevNonce/RJCount/FCntUP가 재사용) |
| “JoinReqFailed” | JoinReq의 Join 서버 처리가 실패했다. |
| “NoRoamingAgreement” | 운영자 사이에 Roaming agreement가 없다. |
| “DevRoamingDisallowed” | End 디바이스는 roam하기 허락되지않는다. |
| “RoamingActDisallowed” | End 디바이스는 roaming동안 activation을 수행하는 것은 허락되지 않는다. |
| “ActivationDisallowed” | End 디바이스는 activation을 수행하는 것은 허락되지 않는다. |
| “UnKnownDevEUI” | DevEUI와 연관된 Context가 없다. |
| “UnKnownDevAddr” | DevAddr과 연관된 Context가 없다. |
| “UnKnownSender” | SenderID 또는 SenderNSID가 알 수 없거나 둘 간에 맞지 않는다. |
| “UnKowmReceiver” | ReceiverID 또는 ReceiverNSID가 알 수 없거나 둘 간에 맞지 않는다. |
| “Deferred” | 일정 시간동안 Passive Roaming이 허락되지않는다. |
| “XmiFailed” | Forwarding network 서버가 DL packet을 전송하는 것을 실패했다. |
| “InvalidFPort” | DL에 대한 유효하지않은 FPort(ex: FPort = 0) |
| “InvalidProtocolVersion” | ProtocolVersion이 지원되지 않는다. |
| “StaleDeviceProfile” | Device Profile이 오래됨. |
| “MalformedMessage” | JSON 구문분석이(parsing) 실패  (누락된 object 또는 잘못된 context) |
| “FrameSizeError” | PHYPayload 또는 FRMPayload의 잘못된 크기 |
| “Other” | 아직 표준화기 되지않은 인코딩 오류 케이스들에 사용된다. |

사용될 때, 결과 object의 설명 필드는 부가적으로 오류세부를 표현한다