

碩士學位論文

Mobile RFID Network에서 개인정보서비스와 EPC  
Network의 서비스 연동

Personal Information Services in Mobile RFID  
Networks and Service Interworking with EPC Networks

韓國外國語大學校 大學院  
컴퓨터 및 情報通信工學科  
韓 敏 奎

碩士學位論文

Mobile RFID Network에서 개인정보서비스와 EPC  
Network의 서비스 연동

Personal Information Services in Mobile RFID  
Networks and Service Interworking with EPC Networks

指導教授 洪 珍 杓

韓國外國語大學校 大學院  
컴퓨터 및 情報通信工學科  
韓 敏 奎

碩士學位論文

Mobile RFID Network에서 개인정보서비스와 EPC  
Network의 서비스 연동

Personal Information Services in Mobile RFID  
Networks and Service Interworking with EPC Networks

指導 洪 珍 杓 教授

이 論文을 碩士學位請求論文으로 提出합니다.

2005年 12月 7日

韓國外國語大學校 大學院  
컴퓨터 및 情報通信工學科  
韓 敏 奎

이 論文을 韓敏奎의 碩士學位 論文으로 認定함.

2005年 12月 7日

審査委員 金 明 珍 (印)

審査委員 朴 相 垣 (印)

審査委員 洪 珍 杓 (印)

韓國外國語大學校 大學院

## 감사의 글

먼저 지금까지 제가 힘들 때 제 기도를 들어 주셨던 하느님께 감사드립니다. 2년간의 석사과정을 지내면서 제가 가야할 길을 제시해주시며, 부족한 저를 항상 이끌어 주신 홍진표 교수님께 진심으로 깊은 감사를 드립니다. 어려운 일에 봉착했을 때마다 저를 올바른길로 이끌어 주셨으며, 학문뿐 아니라 모든면이 부족한 저를 일깨워주셨습니다. 항상 믿어주셨던 스승님이셨기에 다시금 감사의 말씀을 드립니다. 또한 많은 가르침을 주신 정일영 교수님, 김명진 교수님, 김희동 교수님, 정성호 교수님, 박상원 교수님, 윤병남 교수님, 진병문 교수님께도 감사드립니다.

2년동안 많은 말을 하진 않았지만 항상 믿음이 가는 친구 진섭이, 자신보다는 남을 생각하는 영준이형, 말은 안하지만 든든한 후배 우철이, 항상 변함없이 성실함으로 도와준 병희, 부족한 점이 있는 저의 모습을 모니터링 해주는 후배이자 귀여운 동생인 일우에게 고맙다는 말을 전하고 싶고, 저에게 많은 조언과 도움을 주셨던 대학원 선배 양중이형, 상담의 대가 철운이 대학원후배 회국이, 무성이, 대현이에게 감사를 드립니다. 그 밖에 항상 활기찬 모습으로 랩 분위기를 북돋아준 학부후배이자 대학원 후배가 될 준, 희준, 백동, 동욱이 이들 모두에게 고맙다는 말을 전하고 싶습니다.

항상 제 생각을 먼저 물어보고 그 결정에 대해 믿어주며 힘을 북돋아 주었고, 이기적인 저의 모습까지 사랑해주는 사랑하는 민희에게 고맙다는 말을 전하며, 마지막으로 이젠 결혼해서 항상 볼수 없어 아쉽지만 아직도 저를 챙겨주는 때론 누나 같은 때론 든든한 친구 같은 사랑스런 동생 미영이, 너무나 사랑이 많으신 분이기에 저의 특징을 다 받아 주시며 제가 선한길로만 걸을 수 있게 이끌어 주신 어머니, “네 뜻대로 해라, 우린 널 믿는다” 라고 하시며 제게 용기를 북돋아 주셨으며, “진정한 사람” 이 되는 길을 일깨워 주신 세상에서 제일 존경하는 아버지께 감사드립니다. 말로 부족하지만 고개 숙여 진심으로 감사드리며 이 논문을 드립니다.

“허물을 벗고 진정한 제가 되도록 노력하겠습니다”

2006년 1월

한 민 규 올림

# 목 차

<b>제1장 서론.....</b>	<b>1</b>
<b>제2장 FID 네트워크 개념과 서비스 표준화 동향 .....</b>	<b>4</b>
제1절 EPCGLOBAL 네트워크 구조 프레임 워크.....	4
제2절 VERISIGN RFID 네트워크 서비스 구성 .....	1 2
제3절 NIDA RFID 네트워크 서비스 구성 .....	1 4
제4절 MOBILE RFID 네트워크 서비스 구성 .....	1 7
<b>제3장 MOBILE RFID 네트워크를 위한 정보모델과 개인화된 고유 서비스.....</b>	<b>2 1</b>
제1절 MOBILE RFID 네트워크의 분류 .....	2 1
제2절 MOBILE RFID NETWORK TYPE-2에서 INFORMATION MODEL .....	2 4
제3절 PERSONAL-IS: 개인과 객체간 관계형 IS .....	2 6
<b>제4장 EPC NETWORK와 MOBILE RFID NETWORK간 서비스 연동 .....</b>	<b>3 3</b>
제1절 통상적인 EPC 네트워크 서비스.....	3 3
제2절 이동통신망에서 정보검색 서비스 구조 .....	3 4
제3절 서비스 연동을 위한 요구사항 .....	3 8
제4절 EPC-PROXY: CONTENTS와 PROTOCOL의 변환 .....	3 9
제5절 ONS를 이용한 CONFIGURATION HIDING .....	4 0
제6절 서비스 연동 시나리오.....	4 1
<b>제5장 EPC NETWORK와 서비스 연동 가능한 MOBILE RFID 네트워크 구성요소 구현</b>	<b>4 5</b>
제1절 ONS RESOLVER 설계 및 구현.....	4 5
제2절 MOBILE CLIENT 구현 .....	5 7
제3절 PERSONAL IS 구현 .....	5 9
제4절 EPC-PROXY 설계 및 구현.....	6 1
<b>제6장 MOBILE RFID 네트워크와 EPC 네트워크간 서비스 연동 시험 .....</b>	<b>7 3</b>
제1절 서비스 연동환경.....	7 3

제2절 서비스 연동 시험 .....	7 6
제7장 결론 및 향후 개선 방향 .....	8 3
참 고 문 헌.....	8 5

## 표 목 차

표 4-1 PRIMITIVE DATA TYPES.....	3 5
표 4-2 이동 RFID 네트워크의 LOCAL ONS의 ZONE파일 구성.....	4 0
표 5-1 MAIN METHOD LIST .....	6 5
표 5-2 XMLNODEPTR의 기능.....	7 0
표 5-3 MAIN METHOD LIST .....	7 1



# 그림 목 차

그림 2-1 EPCGLOBAL ARCHITECTURE 구조 .....	4
그림 2-2 EPCGLOBAL ARCHITECTURE FRAMEWORK.....	7
그림 2-3 ONS QUERY .....	1 1
그림 2-4 NAPTR RESOURCE RECORD FOR EPC .....	1 2
그림 2-5 EPC NETWORK 구성.....	1 3
그림 2-6 RFID 네트워크 구성도 .....	1 5
그림 2-7 MOBILE RFID 구조 .....	1 7
그림 2-8 MOBILE RFID 서비스 인프라 구조 .....	1 9
그림 3-1 IP TRANSPORT .....	2 1
그림 3-2 IP TRANSPORT + IS.....	2 2
그림 3-3 이동 RFID 네트워크 COMPONENTS.....	2 3
그림 3-4 RFMS RELATION MODEL.....	2 4
그림 3-5 PERSONAL-IS REPOSITORY DATA MODEL.....	2 5
그림 3-6 REGISTERPROFILE SERVICE FEATURE .....	2 7
그림 3-7 GETPROFILE SERVICE FEATURE .....	2 8
그림 3-8 REGISTERINTEREST SERVICE FEATURE .....	2 9
그림 3-9 REVIEWINTERETS SERVICE FEATURE.....	3 0
그림 3-10 REGISTERVISIT SERVICE FLOW .....	3 1
그림 4-1 EPC SERVICE.....	3 3
그림 4-2 통상적인 이동통신망에서 정보검색 서비스 구조 .....	3 4
그림 4-3 WAP 과 인터넷 .....	3 5
그림 4-4 UINTVAR DATA FORMAT .....	3 6
그림 4-5 WSP PDU .....	3 6
그림 4-6 PDU TYPE ASSIGNMMMENTS .....	3 7
그림 4-7 HEADER ENCODING.....	3 8
그림 4-8 EPC-PROXY의 기능 .....	3 9
그림 4-9 이동 RFID 네트워크의 APPLICATION 알고리즘 .....	4 1
그림 4-10 이동 RFID 네트워크 -> EPC 네트워크 .....	4 2

그림 4-11 EPC 네트워크 -> 이동 RFID 네트워크 .....	4 3
그림 5-1 DNS 메시지 포맷 .....	4 5
그림 5-2 DNS 메시지 HEADER SECTION 포맷 .....	4 6
그림 5-3 DNS 메시지 QUESTION SECTION 포맷 .....	4 7
그림 5-4 DNS 메시지 ANSWER, AUTHORITY 및 ADDITIONAL SECTION 포맷 .....	4 8
그림 5-5 NAPTR RR HEADER .....	4 9
그림 5-6 NAPTR ANSWER SECTION .....	4 9
그림 5-7 NAPTR RR 의 RDATA 의 각 필드들 .....	5 0
그림 5-8 URI형식의 RFID CODE 변환 .....	5 1
그림 5-9 EPC IS 처리 과정 .....	5 2
그림 5-10 ONS RESOLVER URITODN() .....	5 3
그림 5-11 ONS RESOLVER MKQNAME() .....	5 4
그림 5-12 ONS RESOLVER MKQUERYPKT() .....	5 5
그림 5-13 EPC CLIENT 에서 URI CONVERTING 과정 (1) .....	5 6
그림 5-14 EPC CLIENT 에서 URI CONVERTING 과정 (2) .....	5 6
그림 5-15 DNS QUERY PACKET 입력 .....	5 7
그림 5-16 WSP CLIENT FLOW .....	5 8
그림 5-17 WSP CLIENT DECODE() .....	5 9
그림 5-18 WSP SERVER FLOW .....	6 0
그림 5-19 MAKE WSP REPLY .....	6 0
그림 5-20 MOBILE RFID 네트워크의 정보이용 .....	6 2
그림 5-21 EPC 네트워크의 정보이용 .....	6 2
그림 5-22 PROGRAM SEQUENCE CHART .....	6 3
그림 5-23 EPC 네트워크로 정보요청 .....	6 4
그림 5-24 WSP와 HTTP HEADER의 매핑 테이블 .....	6 5
그림 5-25 WSP REQUEST를 HTTP REQUEST로 변환 .....	6 6
그림 5-26 WSP REPLY MESSAGE .....	6 6
그림 5-27 HTTP, WSP MAPPING CODE SEARCH METHOD .....	6 7
그림 5-28 WML TAG COMPILER .....	6 7
그림 5-29 WML TAG .....	6 8

그림 5-30 WML TAG TOKEN .....	6 8
그림 5-31 XMLDOC STRUCTURE.....	6 9
그림 5-32 XMLNODE STRUCTURE.....	6 9
그림 5-33 네스팅된 문서구조 .....	7 0
그림 5-34 MOBILE RFID 네트워크로 정보요청 .....	7 0
그림 5-35 WSP REQUEST.....	7 2
그림 5-36 PERSON-IS URL 해석.....	7 2
그림 6-1 TESTBED ONS DELEGATION CHAIN.....	7 3
그림 6-2 ROOT DNS 설정 .....	7 4
그림 6-3 "ICC"도메인의 ZONE파일 구성 .....	7 4
그림 6-4 "HUFIS"도메인의 ZONE파일 구성 .....	7 5
그림 6-5 테스트베드 구성 .....	7 5
그림 6-6 MOBILE RFID 네트워크에서 EPC 네트워크의 정보이용 흐름 .....	7 7
그림 6-7 WAP CLIENT(RESOLVING 결과화면).....	7 8
그림 6-8 EPC-PROXY 결과화면 .....	7 8
그림 6-9 EPC-IS 결과화면.....	7 9
그림 6-10 WAP CLIENT(WSP REQUEST 결과화면) .....	7 9
그림 6-11 EPC 네트워크에서 MOBILE RFID 네트워크의 정보이용 흐름 .....	8 0
그림 6-12 EPC CLIENT(ONS RESOLVING) 결과화면 .....	8 0
그림 6-13 EPC-PROXY 결과화면 .....	8 1
그림 6-14 PERSON-IS 결과화면 .....	8 1
그림 6-15 EPC CLIENT REQUEST 결과화면.....	8 2

## 제1장 서론

RFID 네트워크는 전자태그간의 통신(센서 네트워크)으로부터 시작하여 태그에 할당된 RFID 코드로부터 실제 태그에 맵핑되는 정보를 검색하거나 태그가 부착된 객체의 변화를 등록하는 일련의 과정을 위한 유무선 통신과 주체 모두를 말한다. RFID 검색시스템을 구축함으로써 단일기관의 네트워크를 벗어난 물건에 대한 지속적인 정보서비스를 제공받을 수 있다. 이로 인해 농축산물의 생산 및 유통에 관한 정보를 구매자가 얻을 수 있으며, 물류의 이동과 창고관리, 진품확인 등 다양한 분야에 걸쳐 한 지역에 국한되지 않고 전 세계 어느 곳에서도 전자태그의 코드를 읽음으로써 RFID코드에 맵핑되는 정보를 찾을 수 있다.

RFID 네트워크는 크게 인터넷망의 EPC 네트워크와 무선망의 이동 RFID 네트워크로 분리되어 발전되고 있다. 두 네트워크는 물리적인 시스템환경과 논리적인 프로토콜 데이터포맷이 틀려 데이터 통신이 불가능 하며 서비스도 각 네트워크의 기능에 따라 틀려질 것이다. 이에 두 네트워크에서 제공하는 서비스는 상호운용 및 호환으로 자연스럽게 연계되는 방안이 필요하다.

RFID 네트워크의 기준이되는 EPC 네트워크망은 HTTP로 통신하며, 데이터 포맷으로는 HTML/XML을 사용하는 반면, 이동 네트워크에서 사용되는 프로토콜은 WAP 이고 데이터 포맷으로는 WML을 사용한다. 이처럼 이동 RFID 네트워크와 EPC 네트워크는 표현방법과 프로토콜이 다르므로 상호정보 교환 및 공유 위해서 컨텐츠와 통신 프로토콜을 변환시켜줄 수 있는 시스템이 필요하다. 본 논문에서는 그 시스템을 EPC-Proxy로 정의 하였으며, 이 소프트웨어 모듈을 이용하면 RFID 리더와 RFID 코드가 인식된 핸드폰(RFMS:RFID Mobile Station)으로 다음과 같은 서비스가 가능하다.

먼저 Mobile RFID 네트워크의 가입자가 쇼핑을 하는경우 관심물품을 RFMS로 태깅하여 그 물품의 RFID 코드를 통하여 상세정보를 서비스 받을 수 있으며, 태깅한 물품이 마음에 든다면 자기만의 Personal-IS에 그 정보를 저장하여, 다른 물품과의 비교가 가능하다. 또한 EPC 네트워크의 DS서비스를 이용하여, 구매를 결정한 후에는 그 유통과정을 EPC-Proxy의 Convesion 서비스를 통해 EPC 네트워크에 가입한 EPC Client 또는 본인의 RFMS를 통해 상세하게 알 수 있는 장점이 있다. 이와 같은 서비스는 웹서비스에서 “장바구니 서비스”로 명칭되고 있다.

다음으로 다음과 같은 상황을 가정할 수 있다. RFMS는 각자 RFID 태그를 가지고 있으므로, “RFID 명함서비스” 또는 “RFID 이성 찾기” 등의 서비스를 생각할 수 있다. “RFID 명함 서비스”는 본인의 RFMS를 다른 RFMS가 태깅하여 본인의 정보를 상대방이 RFMS를 통하여 태깅한 사람의 Personal-IS에 접속하여, 그 태깅한 상대방 정보를 자신의 Personal-IS에 저장하여 명함리스트를 자동으로 관리할 수 있는 것이 가능하다. 상대방의 Personal-IS에 접속시 상대방으로 자신이 정보를 요청한 다는 요청정보를 보내 정보보안 과정을 거쳐야 하며, EPC 클라이언트 또는 자신의 RFMS를 통해 Personal-IS에 접속해 태깅한 상대방의 개인 정보를 관리할 수 있다.

“RFID 이성 찾기 서비스”는 “RFID 명함서비스”와 비슷한 유형으로 볼 수 있다. 우선 RFMS에 본인의 이상형 정보를 저장하고, 외출시에 RFMS를 통해 자기의 이상형이 감지될시 그 이성과 대화하기 위해 상대방에게 정보이용 수락요청을 보내 허락을 받은 후, 상대방의 Personal-IS에 접속하여 신상정보를 전달받고, 데이트 신청등의 서비스를 통해 이상형과 대화할 수 있는 서비스이다.

이런 서비스는 우선 다음과 같은 개념적 모델이 우선되어야 한다.

- RFID를 검색할 수 있는 검색서비스 모델
- 검색서비스를 통해 받은 실제 정보의 URL로의 연결 과정 모델
- 연결 시도시 각 서비스마다 맞는 보안 시스템 모델
- 필요한 정보는 자기의 저장소에 저장할 수 있는 Register 서비스 모델

위에 제시된 모델은 현실에 초점을 맞춘 서비스모델이며, 네트워크 관점에서 접근하면 EPC 네트워크와 Mobile RFID의 연동모델이 제시되어야 한다. 즉 Mobile RFID 네트워크의 정보는 EPC 네트워크를 통해 서비스 받을 수 있어야하며, EPC 네트워크의 정보는 Mobile RFID 네트워크를 통해 서비스 받을 수 있어야 한다. 3장과 4장에서는 위에 제시된 서비스 모델과 네트워크 관점에서 두 네트워크의 연동방안과 그 서비스 흐름을 제시할 것이다.

본 논문은 개념 정립단계를 넘어 테스트 단계에 있는 RFID 네트워크의 대표적 구축모델인 EPC Network와 RFID를 Mobile 환경에 적용한 Mobile RFID Network의 Service Interworking을 위한 개념적 모델을 Technical 문서 [24],[25],[26]를 통해 연구된 결과를

제시하며, 그 제시된 결과의 검증을 보여준다. Technical 문서는 다음과 같은 사항이 정의되었다.

- Mobile RFID Relational Model

- 사물에서 개인으로의 개념적 객체전환 모델

- Mobile RFID Network의 개념적 모델의 타입별 분류

- Mobile RFID Network의 타입별 분류에 적용된 새로운 구성 Component 정의

- RFMS : RFID Mobile Station(Include Tag, Reader)

- Personal-IS : Mobile 환경에 적용할 수 있는 다양한 Repository 모델

- EPC-Proxy : EPC 네트워크와 Mobile RFID Network의 Service Interworking Function

- 제안한 모델의 적용 가능한 Service Model

- Service Model의 구체적인 Sequence Chart

위에 정의된 사항은 3,4장에 걸쳐 설명하며, 5장은 각 구성 Component들의 디자인과 구현에 대한 상세사항을 설명하였다. 6장에서는 구현에 대한 결과를 통해 Service Interworking의 가능성을 검증한다.

## 제2장 RFID 네트워크 개념과 서비스 표준화 동향

### 제1절 EPCglobal 네트워크 구조 프레임 워크

#### 1. 개요

##### 가. EPCglobal Architecture

그림 2-1은 EPCglobal Subscribers에 의해 수행되는 행동 양식과 EPCglobal Architecture의 각 구조를 이루는 요소들의 역할에 대해 나타낸다. EPCglobal Architecture 구조 안에서 행해지는 활동은 다음과 같이 세가지로 구분된다.[3]

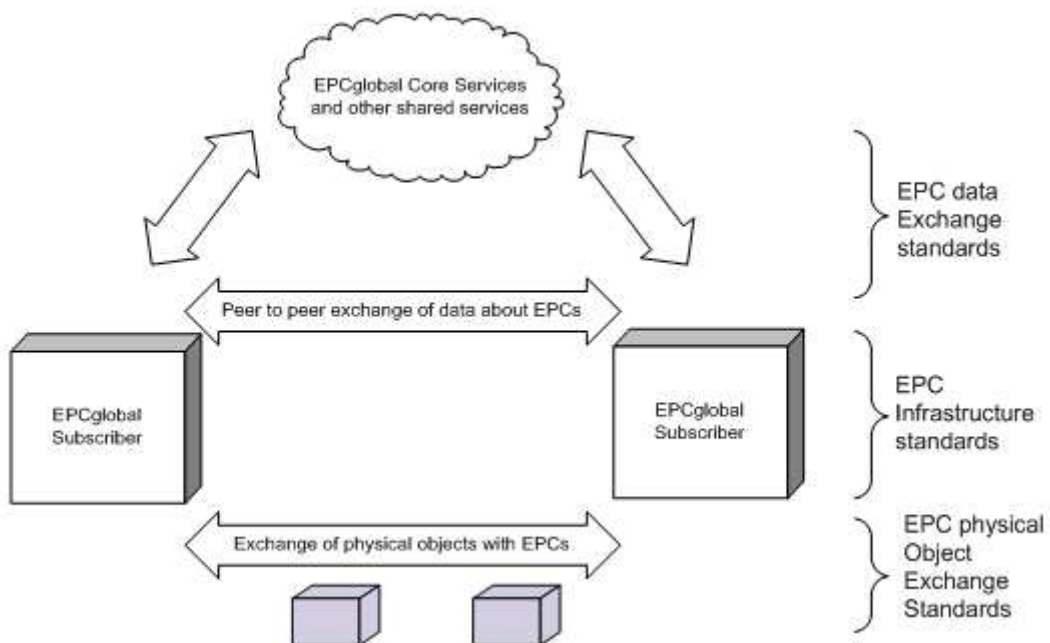


그림 2-1 EPCglobal Architecture 구조

#### 1) EPCglobal Physical Object Exchange

Subscriber는 Electronic Product Codes내에서 인식 가능할 수 있는 물리적인 요소들이 이

동하면서 정보교환을 한다. 즉, 최종 사용자 입장에서 physical objects는 무역 상품들이라 할 수 있고, 이동방식은 무역선을 통한 이동으로 볼 수 있다. 각 EPCglobal Subscriber 들 간의 물품의 이동에 따라 물품의 경로는 재갱신 되고, 가입한 어느 곳에서도 그 물품의 경로 및 상세정보를 알 수 있다.[3]

## **2) EPC Data Exchange**

EPCglobal 네트워크가 가입자들에게 제공하는 가장 큰 장점은 물리적인 요소들, 즉 물품들의 이동 경로 및 수량에 대해 확실한 투명성을 보장하는 것이다. 즉, 이동 경로 선상에 놓여있는 가입자들은 물품의 정보를 공유하여 언제든 살펴볼 수 있고, 또한 EPCglobal Core Service를 통해 물품의 이동 및 교환에 대한 정보를 얻을 수 있다.[3]

## **3) EPC Infrastructure**

EPC 데이터를 공유하기 위해서, 각 가입자들은 EPC 태그가 부착된 물품을 가지고 센싱을 통하여 자료를 수집하고 수집한 정보를 모아서 시스템으로 저장하는 역할을 해야한다. 이를 위해, EPCglobal Architecture에서는 EPC 데이터 수집과 기록하는 방식을 정의함으로써 가입자들이 상호 내부의 시스템에서 사용하는 내부 요소들을 이용할 수 있도록 하였다.[3]

# **2. 기반요소**

## **가. Electronic Product Code**

EPC(Electronic Product Code)는 각각의 물품을 구별할 수 있게 구체화한다. EPC는 사업상의 용도로 EPCglobal 네트워크 안에서 이동되는 물품, 적재, 재산, 기타 다른 물건 등에 할당된다. EPC는 태그가 부착된 모든 물건이 EPC 네트워크상에서 이동되고 이를 모두 묶을 수 있는 역할을 한다. 또한 이 인프라를 더욱 굳건히 하고 효율성과 일관성을 유지하기 위해 같은 인터페이스를 갖게 함으로 더욱 큰 장점을 가져다 준다.[3]

## **나. EPC Manager**

EPCglobal Architecture Framework에서 물품에 대한 식별성을 높이는 가장 큰 요인은 바로 비집중화이다. 이 비집중화는 바로 EPC Manager를 통해 가능해지는데, 이때 EPC Manager란



Issuing Agency에 의해 EPC Namespace의 한 부분을 사용할 권리가 있는 EPCglobal subscriber를 의미한다. 즉, Issuing Agency는 효율적으로 EPC Manager를 임명하여 EPC Manager가 독립적으로 EPC Code를 부여하여, 하나 이상의 그 코드 구역을 담당할 수 있도록 지정한다. [3]

## **다. Class Level Data와 Instance Level Data**

EPC는 하나의 물품을 구별할 수 있게 할당된다. 예를 들어, 하나의 물품, 체리소다 음료수는 유일한 하나의 EPC 코드가 입력된다. 몇몇 경우, 일개 하나의 상품에 대해 구별할 필요도 필요하지만, 한 묶음 음료수등의 작은 주제로 묶을 경우가 필요하다. EPC 코드는 크게 세가지의 단위로 구성되는데, EPC Manager Number, Object Class ID, 그리고 Serial Number로 하나의 물품을 구별할 수 있다. Serialized Global Trade Item Number (SGTN) 코드 부착 기술이 한 예이다. 여기서 EPC Manager Number과 Object Class ID는 사실 GTIN으로 바뀌질 수 있고, 이를 밖에서 Product Class로 사용될 수 있다. [3]

## **라. EPC Information Service (EPC-IS)**

EPC-IS 데이터는 가입자들 사이에서 공유되는 정보이며, 이를 통해 이 물품 또는 물품의 묶음이 어디로 어떻게 이동되었고, 또는 물품의 정보가 갱신되었는지를 파악 할 수 있는 정보를 가지고 있다. 따라서 이 정보는 크게 5가지로 나뉘어 질 수 있다. [2][3][18]

- Static Data: 제조부터 폐기 될 때까지, 변하지 않는 정보
- Class-Level Static Data: 이 정보는 물품의 Class에 대한 정보를 의미한다.
- Instance-Level Static Data: 이는 주어진 Class정보 안에서 변하지 않는 정보로 제조날짜, 폐기날짜, 제조번호를 의미한다.
- Transactional Data: 제품이 제조된 후의 상황에 따라 바뀌어지는 정보를 의미한다.
- Instance Observations: 하나 이상의 EPC 코드가 부착된 물품의 상황에 따라 정보가 갱신되어 기록된다. 한 예로, “2005년 3월 10일 오후 4시 10분경에 EPC X가 부산 선착장 통일호 선박에 선착됨” 등과 같은 정보가 기록되며 이 물품이 선박을 통해 다른 항구에 도착될 시 그 항구의 리더기를 통해 같은 형식의 정보가 기록되거나 혹은 갱신된다. 이 정보는 4가지의 정보(시간, 위치, 하나 이상의 EPC Code, Business 절차 또는 경로)로 구성된다.

- Quantity Observation: 이 정보는 특정 Class 단위 내에서 물품의 수량을 기록되는 정보이다. 예를 들어, “2005년 5월 10일 오후 3시 20분경 도곡지점 웨미리 마트에서 Class C단위로 물품 4100개가 관찰됨”와 같다. 이 때의 정보는 5가지의 정보(시간, 장소, Class, 수량, business 단계)로 구성된다.

### 3. 네트워크 프레임워크

#### 가. EPCglobal 구조 프레임워크

그림 2-2는 태그로부터 정보를 읽고 이를 처리하기 위한 각 요소들의 역할과 Local ONS, 다른 EPC-IS 및 기타 어플리케이션과의 정보흐름 및 처리과정을 나타낸다. [3]

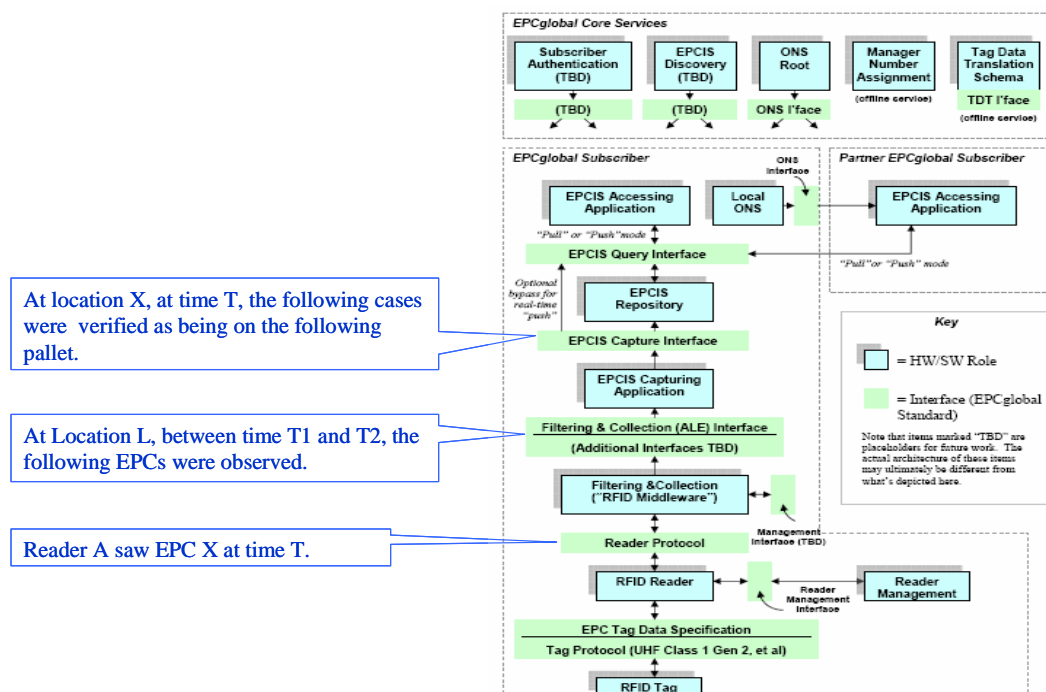


그림 2-2 EPCglobal Architecture Framework

그림 2-2에서 일련의 과정 중 각 요소들이 하는 역할은 다음과 같다.

#### 1) RFID Tag

- EPC Code를 내장함

- EPC Code는 이후의 제조 과정에서 재갱신될 수 있다. 단, 변하지 않는 일회성 정보는 갱신 불가능함
- Lock, Kill, Access Control등과 같은 부가적인 기능도 포함함

## **2) EPC Tag Data Specification (Interface)**

- Electronic Product Code에 대한 전체적인 구조를 정의
- Specific EPCglobal coding schemes을 정의

## **3) Tag Protocol (Interface)**

- 리더와 태그 사이의 명령 교신
- 태그와 리더의 명령 교신에 대한 응답 방법
- 리더기의 범위 안에 인식된 하나 이상의 태그와의 교신 방법
- 서로 간의 간섭 최소화

## **4) RFID Reader (Role)**

- 인식 가능한 범위 내의 태그 정보를 태그 프로토콜에 따라 host Application으로 전달함
- Host Application의 정보를 태그에 기록가능
- Host Application을 통하여 태그의 상태 및 기능을 조절가능
- 읽어들이는 태그의 정보들로부터 자료통합 및 Filtering 가능

## **5) Reader Protocol (Interface)**

- 태그 읽기/쓰기 및 수집/목록작성
- 태그의 사용자, 접근 방법, 태그 인식 설정 및 기타 local, kill등에 대한 설정
- RFID Reader management functions
- RFID Reader operation의 기능 설정(간섭조절, 측정 방식, 데이터등)
- Tag Protocol operation의 설정(파라미터들에 관한 것들)

## **6) Reader Management Interface (Interface)**

- RFID Reader의 설정 및 옵션에 관한 관리
- RFID Reader의 상태에 대한 관리 및 유지

- RFID Reader의 작동 방식에 대한 관리 및 유지
- 그외 기타 RFID Reader의 관리 정보 갱신 및 유지

### **7) Reader Management (Role)**

- 하나 이상의 RFID Reader기들에 대한 관리 및 유지
- 그외 기타 RFID Reader의 관리 정보 갱신 및 유지

### **8) Filtering & Collection (Role)**

- 하나 이상의 가공되지 않은 태그들의 정보를 읽는다
- EPC의 정보를 가공한다(Transforming, Filtering, Grouping, Aggregating, counting)
- 가공된 정보를 올바른 절차에 따라 다음 요소로 전달한다
- 하나 이상의 Application의로부터 정보 요구를 받을 수 있다.

### **9) Filtering & Collection ALE Interface (Interface)**

- 여러 client applications이 정보를 요청하기 위한 방법을 제공
- 하나의 리더기로 여러 Application들이 정보 요청 방법 제공

### **10) EPC-IS Capturing Application (Role)**

- 이벤트 발생시 처리와 이에 대한 관련된 요소들에게 정보 전달
- 여러 이벤트 발생시 들어오는 정보들에 대한 처리
- 물리적으로 수행 가능

### **11) EPC-IS Capture Interface (Interface)**

- EPC-IS Capturing Applications에 의해 발생된 이벤트를 EPC-IS간에 통신 할 수 있게 함

### **12) EPC-IS Query Interface (Interface)**

- EPC-IS 자료를 얻기 위해 해당 Application으로 요청할 수 있는 수단
- 상호 인증 절차

### 13) EPC-IS Accessing Application (Role)

- enterprise business에 대한 총체적인 처리과정을 담당

### 14) EPC-IS Repository (Role)

- EPC-IS Capturing Applications에 의해 발생된 이벤트 정보를 EPCIS 레벨 단계로 기록

### 15) Object Name Service (ONS) Interface (Interface)

- EPC-IS service alc 기타 다른 서비스를 위해 참조 가능한 수단 제공

## 나. Object Name Service (ONS)

ONS는 기존 인터넷에서 사용되는 DNS를 사용하여, EPC에 대한 정보를 얻는다. 따라서, ONS에 질의 및 응답 하기 위해서는 DNS 표준에 따라야 한다. 물론 EPC에 맞게 도메인 이름, 결과 반환 값 역시 DNS Resource Record형태가 되어야 한다. 여기서 DNS 용어와 ONS 용어가 혼동될 우려가 있지만, 반드시 구별해야 한다. DNS 용어는 원래의 DNS System을 이용할 때 쓰이는 용어이고, ONS는 EPC에 관련된 DNS 이용 시에 쓰이는 용어이다. 예를 들어, MX Record는 DNS Query이다. 설령 DNS가 하는 일지라도 EPC에 관련된 Query이면 ONS Query이다.[1][4]

### 1) ONS's Usage of DNS

어떤 한 물품에 대해서 정보를 얻기 위해서 DNS를 이용하게 되는데 그 물품의 EPC는 반드시 DNS가 이해할 수 있는 형태로 질의를 해야한다. 태그로부터 읽어들이는 순수한 태그 정보는 비트들의 순수한 나열이 될 것이다. 이것을 다시 URI의 형태로 바꾸어 DNS로 질의를 해야한다. 그림 2-3는 이와 같은 질의 방법 및 형식에 대해 나타내고 있다. [1]

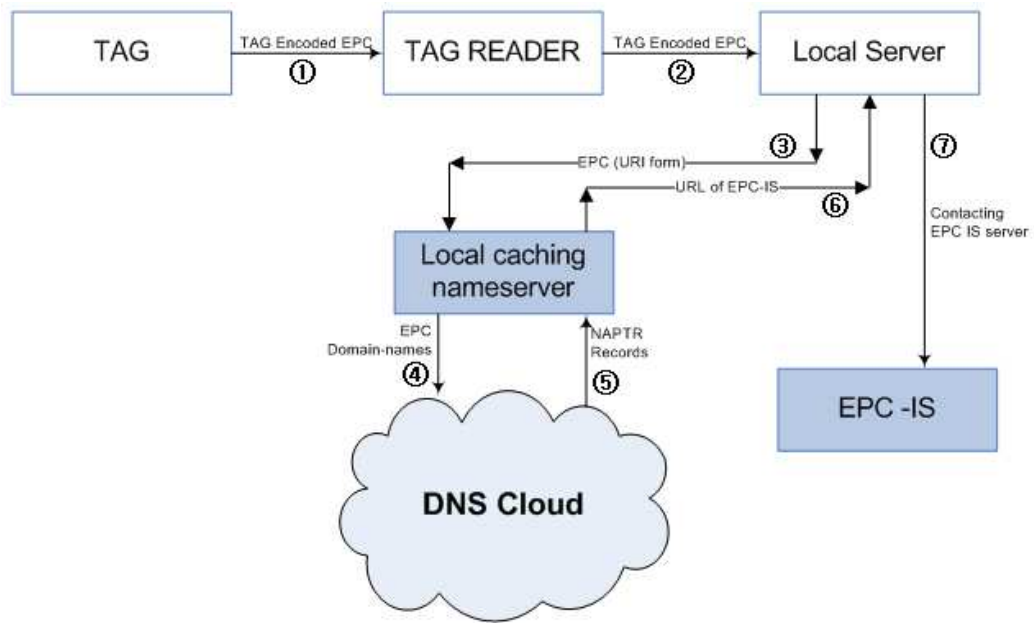


그림 2-3 ONS Query

- ① 하나의 EPC를 포함하는 비트들의 나열을 읽어 들인다.
- ② 이 읽은 태그 정보를 다시 Local System으로 보낸다
- ③ Local System은 이 정보를 URI의 정보로 변환한다
- ④ 리졸버는 URI를 다시 도메인 이름과 DNS query NAPTR형식으로 변환한다
- ⑤ DNS는 이 질의를 받아 하나 이상의 서비스 가능한 URL들을 변환한다.
- ⑥ Local Resolver는 반환된 DNS정보로부터 선택하여 Local server로 다시 반환한다.
- ⑦ 최종적으로 반환된 URL 정보를 이용하여 정보를 가지고 있는 EPC-IS로 접속한다.

## 2) DNS Query Format

EPC에 대한 정보를 얻기 위해 DNS 질의 시, 위의 특정 URI는 도메인 이름 형태로 변환시켜야 한다. 일반적으로 다음과 같은 일련의 과정을 따른다. [1]

- ① ‘urn:epc:’를 제거한다(id:sgtn:2.24.400)
- ② Serial Number Field를 제거한다(id:sgtn:2.24)
- ③ 나머지 필드들의 순서를 바꾼다(24.2:sgtn:id)

- ④ “:” => “.” 로 변환(24.2.sgtn.id)
- ⑤ ‘.onsepc.com’을 붙인다(24.2.sgtn.id. onsepc.com)

### 3) Result Format

질의에 대한 응답은 하나 이상의 NAPTR의 형태로 반환된다. DNS Resolver는 많은 형태로 그 정보를 알려준다. 하지만, 현재 DNS API 표준이 없기 때문에, 바이너리 DNS 정보를 파싱할 수 있는 정확한 방법은 DNS Resolver에 달려있다. 일단 ONS API는 변환 정보 값을 어떻게 파싱할 것인가는 감추어져 있다. 실제적인 DNS 값의 내용은 이론적으로 그림 2-4와 같다. [1][12][20]

Order	Pref	Flags	Service	Regexp	Replacement
0	0	u	EPC+epcis	!^.*\$!http://example.com/cgi-bin/epcis!	. (a period)

그림 2-4 NAPTR Resource Record for EPC

- Order Field : 반환된 결과값의 순서를 DNS가 보장해 주지 못하기 때문에, 정확한 순서로 해석하기 위한 순서를 표시한다
- Flags Field : ‘u’라는 문자는 Regexp 필드값이 완성된 값인 URI의 값이 작성되었다는 표시를 나타낸다.
- Service Field : 주어진 URI에서 제공하는 서비스에 대한 타입을 가르킨다
- Replacement Field : 이 필드 값은 EPC Network에서는 쓰이지 않는다. 단지 빈 공간 대신 ‘.’으로 자리를 메꾼다.
- Service Field : 중요한 필드로서 서버들의classes를 생성한다. 이 classes는 구체적으로 EPC Network Parameter Registry와 함께 저장된다. 이는 ‘이 물건에 대해 가능한 서비스에 대한 목록’에 대해 구체화할 때 사용되는 것이다.
- Regexp Field : URI에 대한 정보를 나타낸다. 이전의 ONS에서는 단순 IP 주소의 결과를 표시 하였다. 하지만, 여러 이용 측면에서 비효율적이기 때문에 형태가 바뀌었다.

## 제2절 Verisign RFID 네트워크 서비스 구성

VeriSign사는 2004년 1월 EPC 네트워크 표준개발 및 감독을 담당하는 EPCglobal로부터 root ONS를 포함한 EPC 네트워크 운영에 대한 권한을 획득하여 미국동부와 서부에 각각 2

개씩 4개의 EPC 네트워크를 시범운영 중이며 최종적으로 13개의 root ONS를 운영할 계획에 있다. 현재까지는 기초적인 시스템만 구축되어 있는 상태이며, 실제 등록된 EPC도 매우 적은 실정이다. VeriSign사는 공급망을 위한 EPC 네트워크 서비스 도입 및 EPC 네트워크를 위한 Root ONS 관리, 기타 무료 관리 서비스(Local ONS, EPC Information Service, EPC Discovery Service, EPC Trust Service) 제공을 목표로 사업을 추진하고 있다.[4]

## 1. 네트워크 구성

VeriSign의 EPC 네트워크는 그림 2-5와 같은 구성을 가지고 있으며, 각 구성 요소의 기능은 다음과 같이 정의 된다. [4][17]

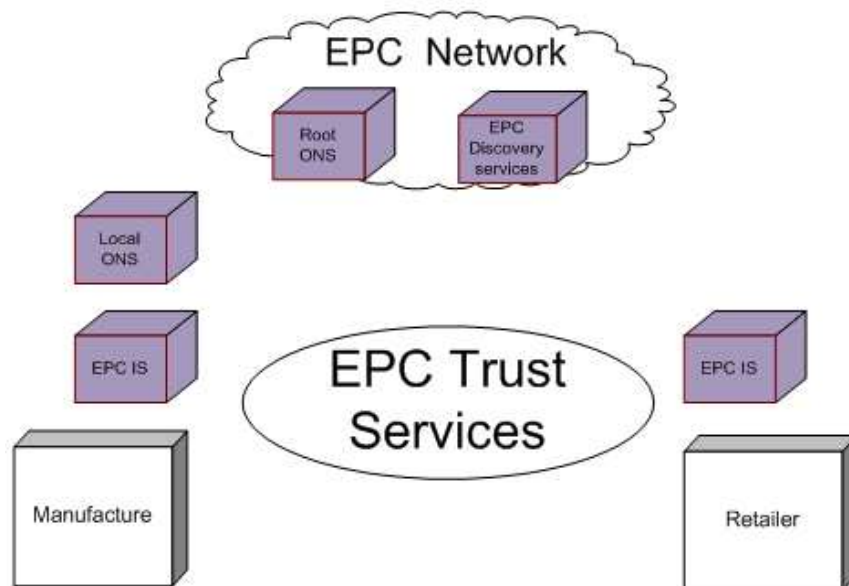


그림 2-5 EPC Network 구성

### 가. Local ONS

Local ONS는 Local DNS와 같은 기능을 한다. DNS는 A와 PTR DNS RR(Resource Record)을 사용하여 존파일을 구성하여 DNS를 관리한다. ONS도 존파일로 RFID Code Prefix를 관리하지만, DNS와 다른점은 NAPTR RR을 이용하여 다양한 서비스를 지원할 수 있는 기능을 한다. [1][3][4][17][18]



## 나. Root ONS

모든 ONS의 최상위 ONS로써 ONS의 Supply Chain을 구성하여, ONS의 Location을 알수 있는 기준 역할을 한다. [1][3][4][17][18]

## 다. EPC IS

ONS에서 관리하는 데이터는 RFID Code의 실제 정보를 관리하는 EPC-IS의 URI이다. EPC-IS는 RFID Code에 상응하는 객체정보를 가지고 있는 저장소 역할을 한다.[2][3][17][18]

## 라. EPC Discovery Services (DS)

EPC-IS는 RFID Code가 만들어지고 수록되는 최초 저장소이며, 이 RFID Code가 EPC-IS에서 다른 EPC-IS로 수록되고자 할 때, DS(Discovery Server)에 그 이동 EPC-IS의 Pointer를 등록하여 현재 RFID Code가 어디로 이동하였는지 알수있게 해주는 서비스이다.[3][4][17][18]

## 마. EPC Trust Service

Trust Service는 EPC 네트워크의 암호화 및 인증을 통하여 정보보호 수단 제공한다. 현재 어떤 Spec.이 제시된 것은 아니며 기존 인터넷에서 사용되는 인증방법을 사용할 것으로 예상된다.[4]

# 제3절 NIDA RFID 네트워크 서비스 구성

한국인터넷진흥원(NIDA)는 VeriSign의 움직임에 대응하기 위해 VeriSign의 ONS를 우리나라의 실정에 맞게 변경한 ODS(Object Directory System)를 시범서비스하고 있다. ONS는 VeriSign에서 관리하는 코드체계로써 EPCglobal에서 규격화한 EPC Code를 사용한다. 그러나 NIDA의 ODS는 EPC Code, UCode, ISO/IEC 정의코드에 대해서 멀티코드를 지원하게끔 설계하여 ONS보다 Code체계의 호환성을 확장하였다.[11][12]

## 1. 네트워크 구성

MDS는 Multi-code Directory System의 약자로 EPC, ISO/IEC 정의 코드 정의 코드, ucode 등 다양한 RFID 코드를 인식할 수 있는 검색 서비스 이다.

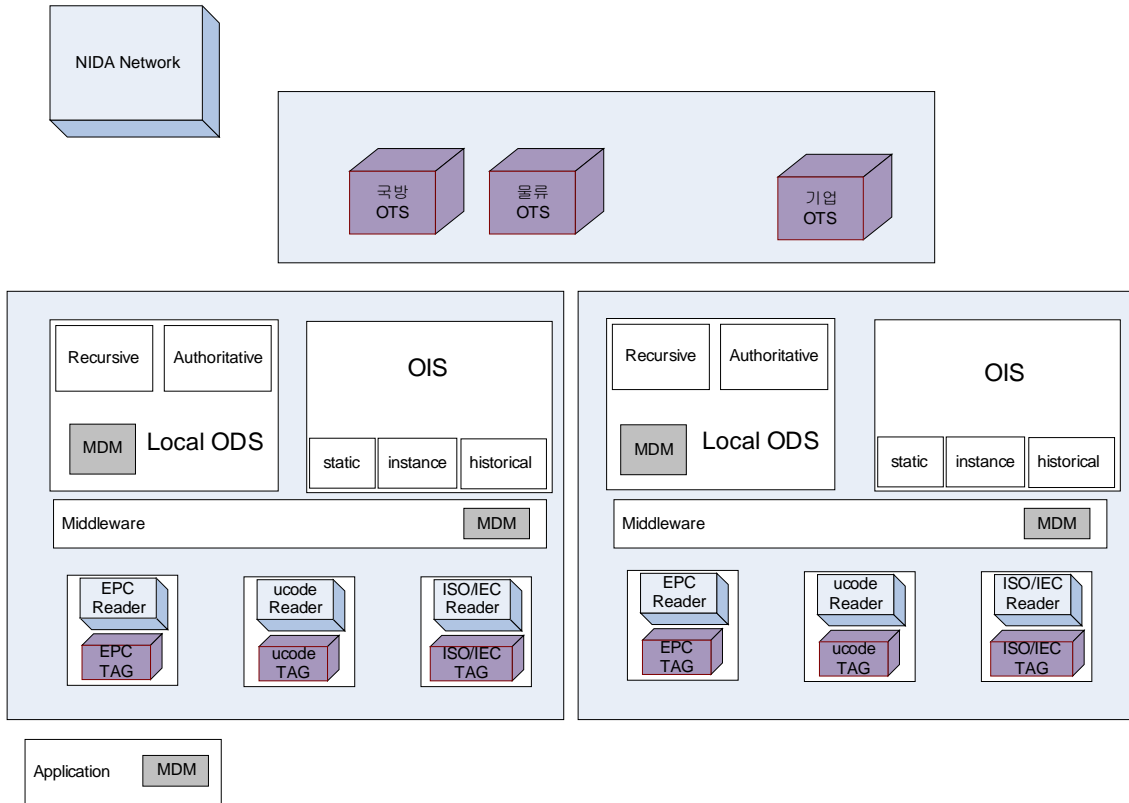


그림 2-6 RFID 네트워크 구성도

그림 2-6 에서와 같이 RFID 네트워크는 크게 객체검색서비스(ODS : Object Directory System), 객체이력서비스(OTS : Object Traceability Service) , 객체 정보서비스 (OIS : Object Information Service) , 미들웨어 (Middleware)등으로 구성된다.

ODS는 RFID 코드에 해당하는 OIS 및 OTS 의 위치정보를 검색하는 시스템으로 국가객체검색 서비스 (National ODS)와 로컬 객체검색서비스(Local ODS)로 이루어 진다. [11][12]

## 가. Object Directory System (ODS)

### 1) Local ODS

Local ODS는 DNS 의 Local DNS와 같은 역할을 수행하며, 존 파일 내에 기관의 OTS 및 OIS 의 위치정보 (Service URI)를 저장하고, RFID 코드 질의에 대해 RFID 코드에 해당하는 OTS 및 OIS 의 위치정보를 제공한다. [11][12]

### 2) National ODS

National ODS는 RFID 네트워크에서 DNS의 Root DNS와 유사한 역할을 수행하며, 국가적 차원에서 관리되는 ODS이다. National ODS는 각 기관의 Local ODS의 위치정보(IP Address)를 존(zone) 파일 내에 저장하고, RFID 코드 질의에 대해 Local ODS의 위치정보를 제공한다. [11][12]

## 나. Object Information Service (OIS)

객체정보서비스(OIS)는 객체의 정적(Static)정보 및 동적(Instance, Historical)정보를 저장하고 제공한다. OIS는 각 기관 내부의 데이터를 저장하는 데이터베이스로써 다양한 형태로 저장된다. 이러한 다양한 형태의 저장된 데이터에 접근하기 위해서는 OIS 접근을 위한 인터페이스가 필요하다. [11][12]

## 다. Object Traceability Service (OTS)

객체 이력서비스(OTS)는 객체가 이동해간 객체정보서비스(OIS)의 위치추적정보를 제공한다. 이러한 OTS 는 각 기관별 혹은 저장되는 데이터의 용도에 따라 국가, 공공기관(물류, 유통, 국방등) 혹은 각 기관별로 관리가 가능하다. [11][12]

## 라. Middleware

미들웨어(Middleware)는 이기종 RFID 환경에서 발생하는 대량의 데이터, 즉 RFID 바코드 등 다양한 형태의 태그들로부터 리더나 센서 등을 통해 수집된 데이터를 전달받아 데이터를 처리, 데이터 표현, 데이터 교환, 정보 네이밍 서비스 등을 위해 필터링 하여 의미있는

정보로 요약하여 응용서비스에 전달한다. [11][12]

## 제4절 Mobile RFID 네트워크 서비스 구성

### 1. 네트워크 구성

Mobile RFID는 국내에서는 모바일 포럼, 국외에서는 노키아가 개념을 정립해가고 있다. RFID 네트워크가 개념정립 단계에서 테스트 단계로 넘어가고, 그 확장방안으로 Mobile을 접목한 Mobile RFID 네트워크도 구상되고 있다. Mobile RFID 네트워크는 단말기에 RFID 태그 리더를 장착하고, 언제 어디서나 태그정보를 이용할 수 있다는 취지이다.

그림 2-7과 같이 RFID 리더가 탑재된 휴대폰으로 Mobile RFID 서비스 용도로 곳곳에 부착된 태그를 읽고, 이 태그 정보를 이용해 태그 ID와 URL의 매핑 정보를 가지고 있는 ODS(Object Directory Service)로 콘텐츠의 URL을 요청하게 된다. ODS는 관련 콘텐츠의 URL을 반환하고, 휴대폰은 반환된 URL을 이용하여 해당 콘텐츠 서버에 해당 콘텐츠를 요청한다. ODS는 관련 콘텐츠의 URL을 반환하고, 휴대폰은 반환된 URL을 이용하여 해당 콘텐츠 서버에 해당 콘텐츠를 요청한다.[14]

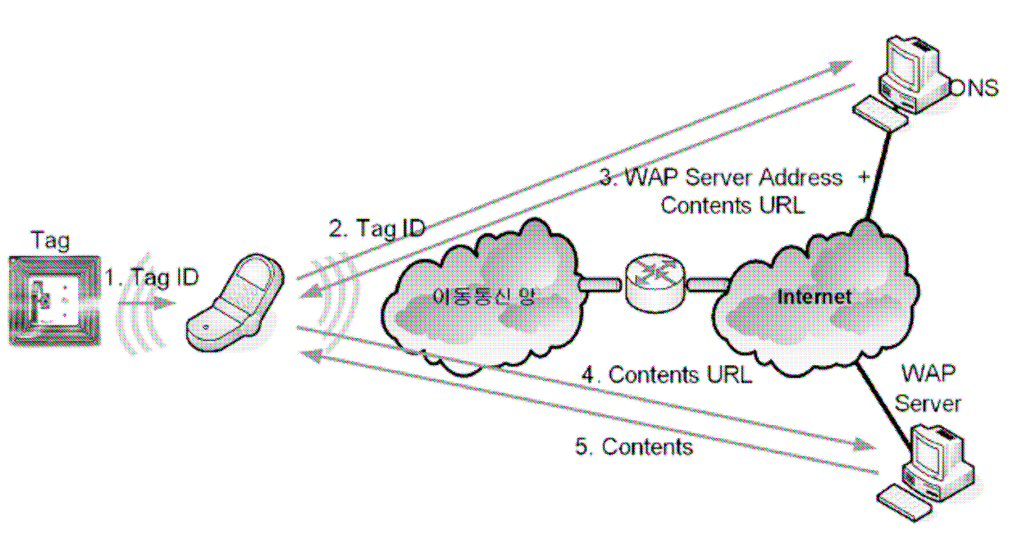


그림 2-7 Mobile RFID 구조

## 2. 표준화 동향

Mobile RFID는 휴대폰에 소형 RFID 리더를 탑재하여 휴대폰으로 RFID 태그를 읽었을 때, 여러 가지 서비스를 이동통신망을 이용하여 제공하는 서비스로 국내에서 처음으로 시도하는 서비스이며 2005년부터 표준화가 진행 중에 있다.

### 가. 국내 동향

Mobile RFID 서비스를 위한 신규 시장확보를 목적으로 지난 2005년 2월, 국내에 Mobile RFID 포럼이 구성되었으며, 포럼 산하에 단말분과, 네트워크 분과, 응용서비스 분과, 정보보호 분과, 시험·인증 분과를 각각 구성하고, Mobile RFID 서비스를 위한 국내 표준 규격을 개발하고 있다. Mobile RFID 포럼은 국내 RFID/USN 협회에서 사무국 역할을 수행하고 있으며, 한국전자통신연구원 표준연구센터에서 Mobile RFID 서비스를 위한 국내 규격 작업을 총괄함으로써 올해 내에 관련 표준 규격작업을 마무리한다는 계획이다.

이를 위해 단말분과에서는 RFID 리더 칩이 휴대폰 내에 탑재 시 요구되는 하드웨어 인터페이스 규격과 휴대폰의 기존 미들웨어 플랫폼인 WIPI와의 연계를 위한 WIPI HAL(Handset Adaptation Layer) 확장규격과 휴대폰 Form Factor 등의 단말규격 개발을 포함한 RFID 태그와 휴대폰 내의 리더간 에어 인터페이스 표준을 개발하고 있다. 특히 무선구간의 에어 인터페이스 표준은 국제표준인 ISO/IEC 18000-6을 채택한 바 있다. ISO/IEC 18000-6는 기존의 type A와 B를 포함하고, 최근 JTC1에 제출된 EPCglobal Class 1 Gen2를 type C로 포함하는 규격을 의미한다.

또한 응용서비스분과에서는 서비스 시나리오를 중심으로 Mobile RFID 서비스 종류 별 응용 요구사항(ARP: Application Requirement Profile)을 작성하고 있다. 여기에는 영화 포스터 안내 서비스, 주류 마케팅 서비스, 버스노선 안내 서비스, 물품정보 조회 서비스, 지하철 주변 정보안내 서비스, 문화재 정보제공 서비스 등이 있다.

네트워크 분과에서는 Mobile RFID 서비스를 제공하기 위한 네트워크 표준 규격을 작성하고 있다. 대표적인 표준 규격으로는 네트워크 서비스를 위한 WIPI API 확장 규격, Mobile RFID 콘텐츠 협상 프로토콜, Mobile ODS(Object Directory Service) 규격, Mobile OIS(Object Information Server) 규격, Mobile OTS(Object Tracking Service) 규격 등을 표준화하고 있다.

또한 정보보호 분야에서 추진하고 있는 국내표준 규격에는 물류유통에서의 프라이버시 보호 규격, 오프라인 하이퍼텍스트 기반 프라이버시 보호 규격, 태그가 부착된 물품 구매에서의 프라이버시 보호 규격을 작성하고 있다. 끝으로 시험·인증 분야에서는 향후 Mobile RFID 서비스 구현을 위해 요구되는 시험·인증 규격을 작성 중에 있다.

한편, Mobile RFID 포럼에서는 상기 5개 분야의 각 표준 규격 작업 간의 상호조율 목적과 Mobile RFID 포럼 내에서의 지적재산권 관리정책 수립, 그리고 이들 규격작업 결과를 국제표준화에 반영하기 위한 목적으로 가칭 ‘표준화 전략 TFT’를 구성하였다. 여기서는 각 분야 별로 진행되고 있는 표준 규격작업을 하나로 묶기 위한 Mobile RFID 서비스 구조를 규격화하고 있으며, 각 분야 별 규격작업을 위한 양식개발 노력을 추진하고 있다(그림 2-8). 또한, 포럼 차원의 지적재산권 정책수립과 함께 국내 규격작업 결과를 국제표준화에 반영하기 위한 표준화 창구개설 노력을 경주 하고 있다. [13][14][15][16]

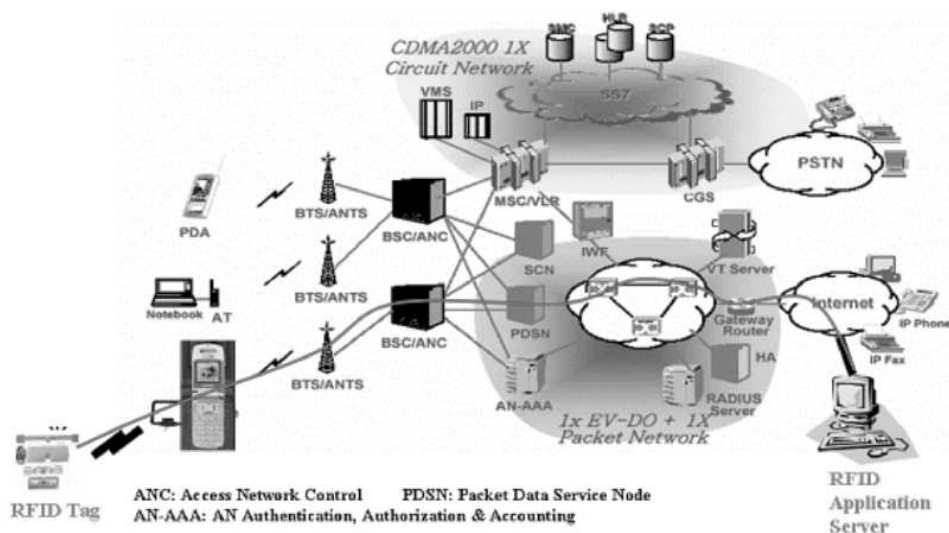


그림 2-8 Mobile RFID 서비스 인프라 구조

## 나. 국외 동향

Mobile RFID 서비스는 유통물류 중심의 RFID 기술과 달리 이동통신 네트워크와 연계되어 동작하는 서비스로써 관련 국제표준화는 아직 진행되고 있지 않다. 그러나 최근 ITU-T 등의 국제표준화 기구에서 RFID와 통신이 결합된 형태의 기술들에 대한 중요성이 부각되고

있으며, 관련 표준화 활동이 시작될 움직임을 보이고 있다.

아시아·태평양 지역의 정보통신 표준화 협의체인 ASTAP에서는 2005년 3월 회의에서 RFID 전문가그룹(EG)이 구성되었으며, 이 그룹에서는 기존의 유통물류 중심의 RFID 뿐만 아니라 통신망과 연계된 RFID 기술에 대해 연구를 진행할 계획으로 있다. 또한 ITU-T에서는 2005년 3월에 열린 ITU-T TSAG(ITU-T 표준화 자문 위원회) 회의에서 RFID 기술의 중요성에 대해 인식하고 향후 ITU-T의 표준화 활동 방향을 수립하기 위해 RFID CG(Correspondent Group)를 구성하였다. 이 CG에서는 ITU-T의 각 SG뿐만 아니라 JTC1과 같은 외부 표준화 기구를 포함하는 구성원들로부터 ITU-T의 RFID 표준화 활동에 필요한 의견을 수집할 계획으로 있다.

2005년 5월초에 열린 ITU-T SG11과 SG13에서도 네트워크 기반의 RFID 기술에 대한 표준화의 중요성이 언급되었으며, ITU-T 내에서 어떤 형태로든 RFID 관련 표준화 작업이 진행될 것으로 예상되고 있어 국내 Mobile RFID 서비스를 위한 규격 작업결과가 ITU-T를 통한 국제표준화로 이어질 수 있도록 세심한 노력이 요구되고 있다. [4][17][18]

# 제3장 Mobile RFID 네트워크를 위한 정보모델 과 개인화된 고유 서비스

## 제1절 Mobile RFID 네트워크의 분류

### 1. Mobile RFID Network Type-1 (IP Transport)

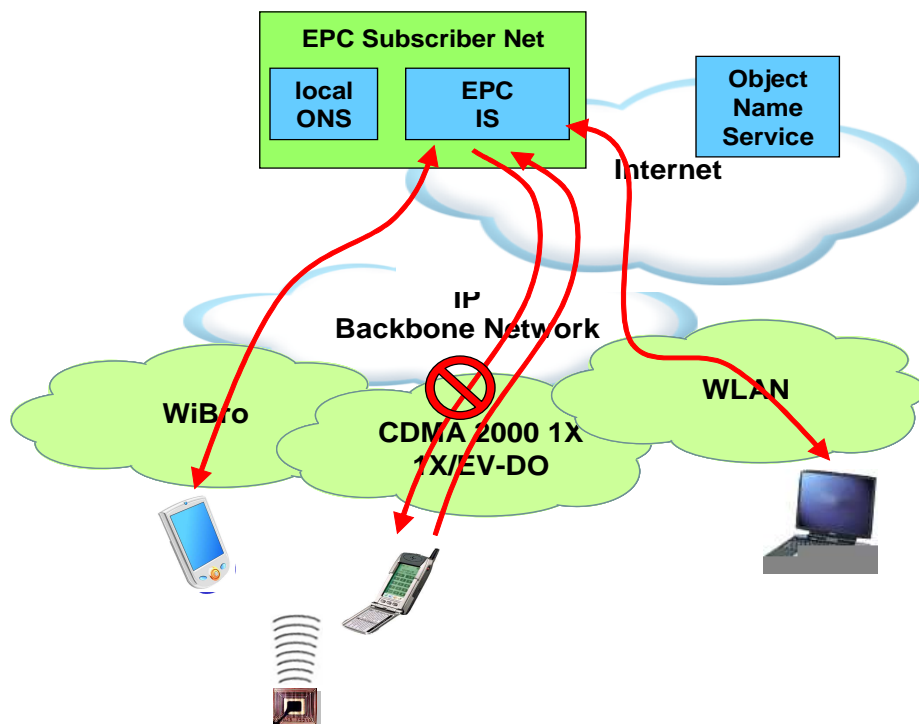


그림 3-1 IP Transport

- EPC 네트워크로의 Wireless Internet Access Service 만 제공
- 단말기에서는 EPCglobal표준에 따르는 미들웨어와 EPC-IS querier/parser 및 웹브라우저를 장착해야 함



## 2. Mobile RFID Network Type-2 (IP Transport + IS)

### 가. 네트워크 특성

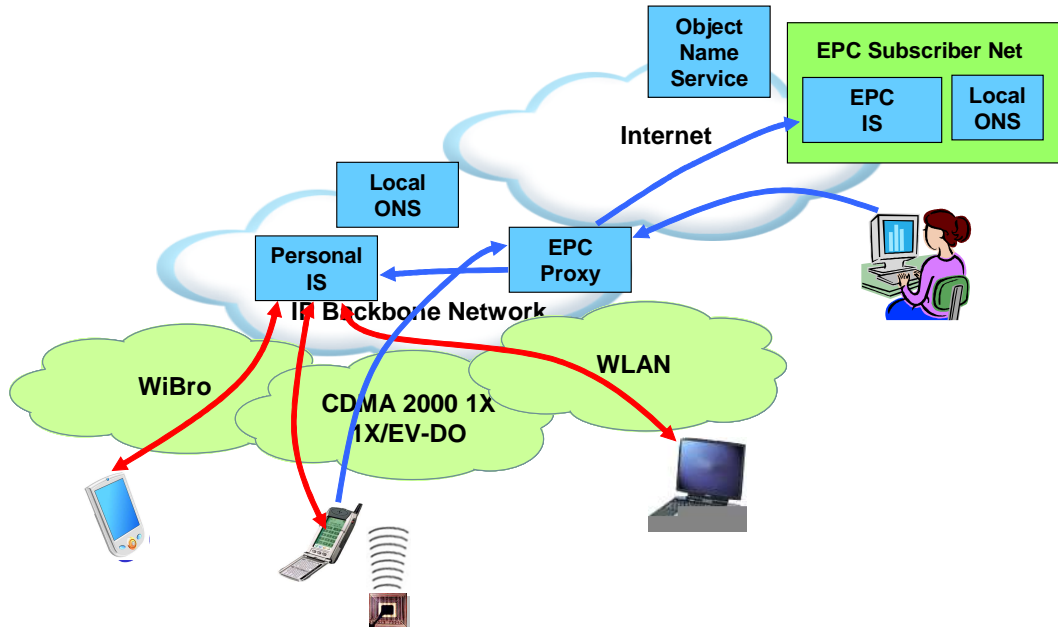


그림 3-2 IP Transport + IS

- Personal-IS Service : 이동통신 사업자만이 할 수 있는 서비스
- EPCIS conversion Service : 휴대용 단말 Capability에 맞게 변환(WAP<->HTTP/SOAP, WML<->HTML/XML)
- Type-1 Service를 기본적으로 포함

### 나. Mobile RFID Network Type-2 구성요소

EPC 네트워크에서 이루어지는 서비스는 국가 정책에서의 물류, 유통과 관련된 사업으로 개발, 발전하고 있지만 단순히 산업적으로 국한되지 않고 궁극적으로는 개인 핸드폰에 자신만의 태그 또는 리더기가 결합된 이동 단말기에도 반드시 응용이 될 것이다. 따라서 사용자들은 자신만의 리더기를 통한 자신을 기준으로 더 이상 물류가 아닌 자신의 정보를 기록할 수 있는 서비스가 창출될 것이고, 이를 즐기게 될 것이다. 이런 사용자 중심의 서비스 창출하기 위한 그림 3-3와 같이 이동 RFID 네트워크 모델이 제시되며, 이를 통하여 이동통신사업자들을 통한 무수히 많은 서비스가 가능하며, 이에 따른 기존의 EPC 네트워크와의

연동 또한 필수적이다.

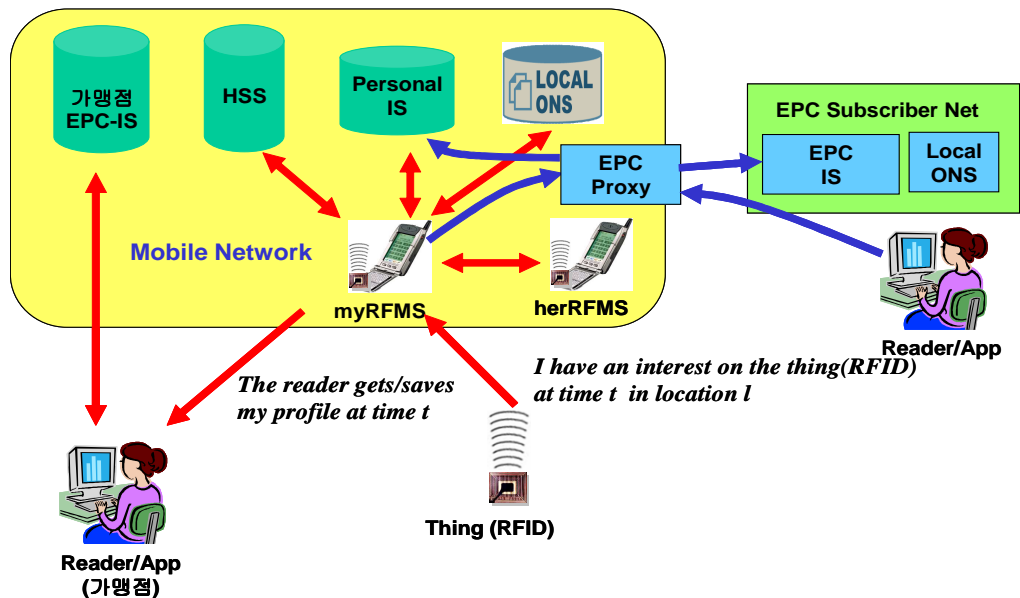


그림 3-3 이동 RFID 네트워크 Components

그림 3-3에서 보듯이 이동 네트워크의 형태 및 구성 요소는 기존 EPC 네트워크와 매우 비슷하다. 단, 단순 사물 및 동식물이 아닌 사용자의 정보를 담당해야 할 기존의 IS와는 조금 다른 형태의 Personal-IS(개념적 의미)와 사용자가 개인적으로 휴대하게 될 RFMS가 새로운 요소로 이용될 것이다. 자신의 RFMS를 기준으로 태그 측면에서는, 기존의 리더가 내 RFMS를 알고 이에 대한 정보와 시간 정보를 함께 기록된다. 또한 리더라는 측면에서는, 원하는 물품 및 기타 등등의 태그에 대한 정보를 시간뿐만 아니라 위치 정보까지 저장함으로써 무수히 많은 응용서비스로 연계될 수 있다. 그림 3-3에서 보듯이 기존의 EPC와의 자료 공유 및 교환을 위해서는, 양단간의 특성 때문에 프로토콜 및 콘텐츠를 바꾸어 줄 수 있는 EPC-Proxy가 요구된다.

## 제2절 Mobile RFID Network Type-2에서 Information Model

### 1. RFMS: Mobile Station with RFID reader/tag

리더기의 기능과 태그가 부착된 개인의 Mobile Station을 통하여 RFID의 서비스를 이용할 수 있다. 자신의 휴대폰 번호는 자연히 RFMS의 구별 단위의 하나로 사용 될 것이며, 리더인 동시에 하나의 태그의 역할을 하기 때문에, 응용 가능한 무수히 많은 서비스가 창출될 것이다. 특징은 다음과 같다.

- Phone number : A MS belongs to a person
- Location information : A phone may know where the things are
- RFID reader : A phone identifies things
- RFID tag : A phone is a thing, too

### 2. Mobile RFID Relational Model

자신의 RFMS를 기준으로 리더기의 측면에서는 일반적인 상품 또는 동식물 및 다른 사람의 RFMS 태그를 읽을 수 있다. 반대로, 태그의 측면에서는 기존 고정된 리더기와 다른 사람의 RFMS의 리더기로 부터 읽혀질 수 있다. 리더기와 태그의 측면으로 관계될 수 있는 경우의 수는 리더 측면에서는 2가지와 태그 측면에서 2가지로 총 4가지의 Relation Model을 가질 수 있다.

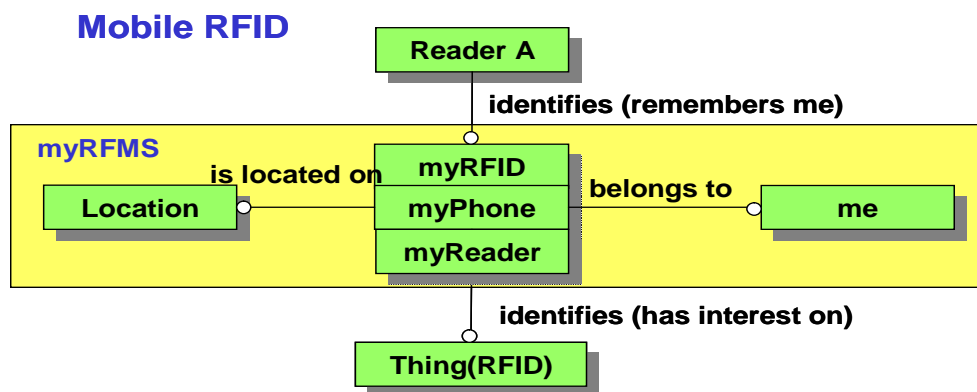


그림 3-4 RFMS Relation Model

### 3. Personal-IS Repository Data Model

Personal-IS는 상품 또는 동식물의 일반적인 태그 정보를 담는 것이 아닌, 한 사람 개인의 기본 정보로 시작하여 이용한 서비스 상품, 목록, 경로 등을 저장함으로써 추후, 자신이 저장한 정보를 토대로한 응용 서비스 이용을 가능하게 한다. 기본적으로 Personal-IS에는 자신의 기본 정보를 바탕으로 위에서 언급한 상품에 대한 목록 및 정보, 서비스 이용 경로 등에 대한 정보를 저장하기 위해 태그 정보뿐만 아니라, 시간, 장소에 대한 정보도 저장된다.

자신의 휴대폰은 자신만의 전화번호로써 자신 식별이 가능해진다. 따라서 위치 및 시간 정보를 이용 및 응용하여 RFMS를 통한 많은 서비스를 창출할 수 있다. 그림 3-4에서와 같이 자신의 RFMS를 통하여 수많은 사람들 중 ‘나’만을 식별이 가능해짐에 따라 개인의 정보 공간을 이용할 수 있으며, RFMS를 통하여 다른 RFMS 또는 시설물 또는 물품에 부착되어 있는 태그로부터 정보를 수집하여 원하는 서비스를 이용할 수 있다.

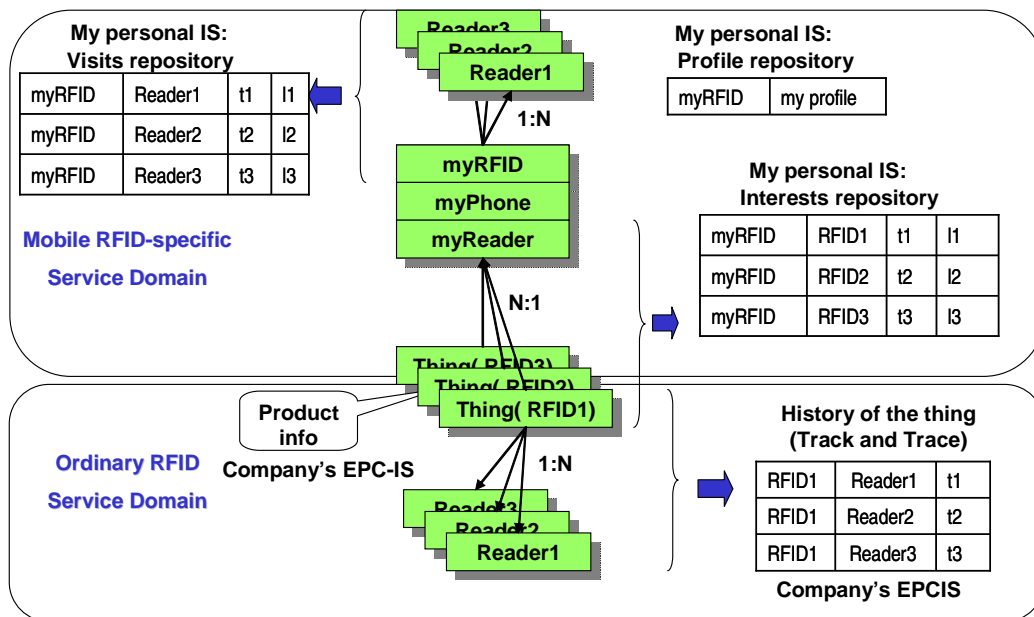


그림 3-5 Personal-IS Repository Data Model

RFMS는 이동통신 사업자가 제공하는 서비스를 통하여 자신이 RFMS를 통한 서비스 내역 및 정보이용을 가능하게 해준다. 그림 3-5은 사물이 아닌 사용자 측면에서 서비스 이용 시 요구되는 저장정보 명세를 간략히 도시화한 그림이다.

가령 한 사용자가 다른 사용자의 RFMS를 인식하여 저장하게 됨으로써 그 상대방과 언제 어디서 어떤 정보 및 서비스를 이용했는지에 대한 기록이 가능해진다. 따라서 친구 및 연인과의 기록이 가능해짐으로써, 자신이 어떤 정보 및 사람과의 행위 등의 상세 정보가 모두 저장 가능해진다.

## 제3절 Personal-IS: 개인과 객체간 관계형 IS

### 1. 서비스 개요 및 요구사항

Personal-IS 서비스는 이동통신 사업자만이 할 수 있는 서비스 창출로써, 개인마다 자신이 원하는 정보를 저장할 수 있는 역할을 해준다. RFMS는 사물(thing)이 아니라 개인(person)을 나타내며 인간이 주체나 객체가 될 수 있는 정보를 저장하고 이용할 수 있게 한다. 즉, RFMS에 부착된 RFID tag는 MS(Mobile Station)를 소유한 개인을 식별할 수 있으며, RFMS에 내장된 리더로 물체에 부착된 tag를 인식하는 일은 RFMS 소유주가 그 물건에 관심을 표명하였다는 의미로 해석할 수 있다. 이와 같은 Personal-IS 서비스는 Mobile Network의 사용자가 자신의 RFMS를 통하여 기존의 EPC Network의 서비스인 IS, DS를 이용할 수 있으며, 반대로 RFMS가 태그라는 측면에서, EPC Network도 반대로 이용할 수 있다. Personal-IS 서비스의 요구사항은 다음과 같다.

- Personal-IS Repository
- Object naming service for accessing Personal- IS
- Personal-IS Accessing Application
- EPC Proxy
- RFID middleware, A set of RFID applications In RFMS

### 2. Personal-IS의 기본 Service Feature

Personal-IS를 위한 기본 서비스 요소는 다음과 같다.

- Profile Repository : 개인의 기본정보를 의미함. 이에 개인의 기초 정보, 관심사, 취미, 이상형 등의 정보를 기록하여 이를 기반으로 다른 서비스에 응용
- Interest Repository : 원하는 물품 및 동식물의 태그를 인식하여 저장함으로써 차후

상세 정보 이용 및 주문 등의 서비스로 연계가능

- Visit Repository : 원하는 장소 및 경로의 정보를 저장함으로써 즐겨 찾는 곳, 단골, 추억의 의미를 부여할 수 있는 서비스로 연계가능

### 3. Profile Repository를 이용한 Service Feature

#### 가. RegisterProfile Service Feature

그림 3-6은 자신의 RFMS를 이용하여 자신의 기본정보(신상 명세, 취미, 좋아하는 것 등)를 저장하는 것을 나타낸다.

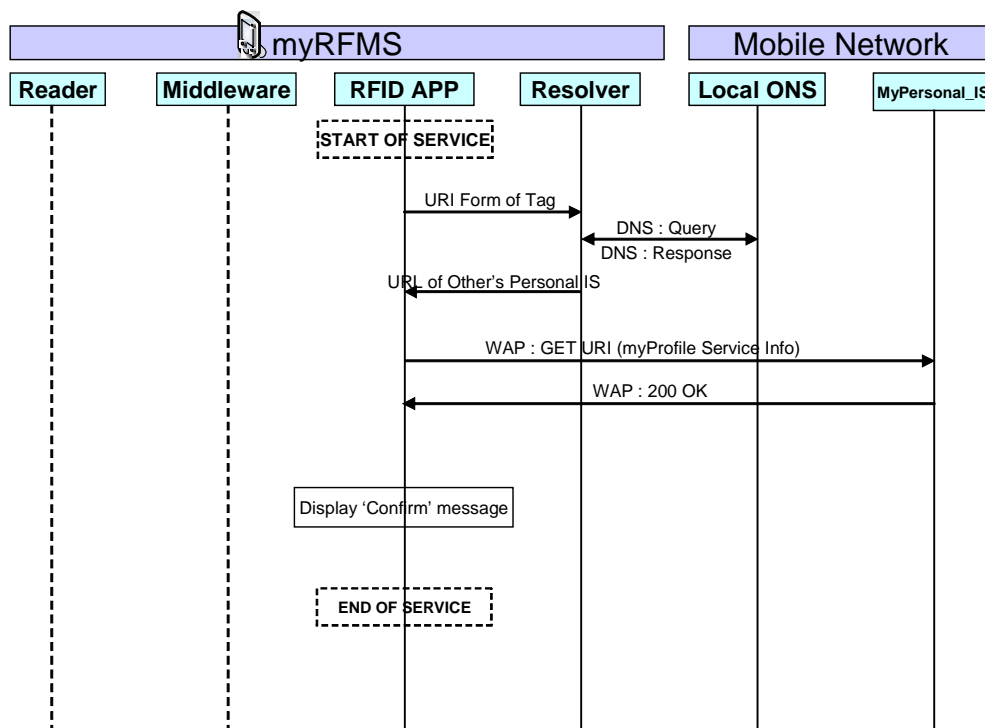


그림 3-6 RegisterProfile Service Feature

- ① 한 가입자가 자신의 RFMS로 기본 정보 및 RFMS의 옵션 설정을 결정하고 입력버튼을 누른다.
- ② 자신의 Personal-IS에 저장하기 위해서 Resolver를 거쳐 Local ONS를 통해 Personal-IS의 위치 정보를 얻는다.
- ③ 가입자가 입력한 정보를 해당 Personal-IS에 저장한다

④ 가입자는 확인 메시지를 보고 서비스를 종료한다.

## 나. GetProfile Service Feature

그림 3-7는 같은 이동 RFID 네트워크내의 다른 가입자에 대한 정보를 알고자 할 때를 도식화한 것이다.

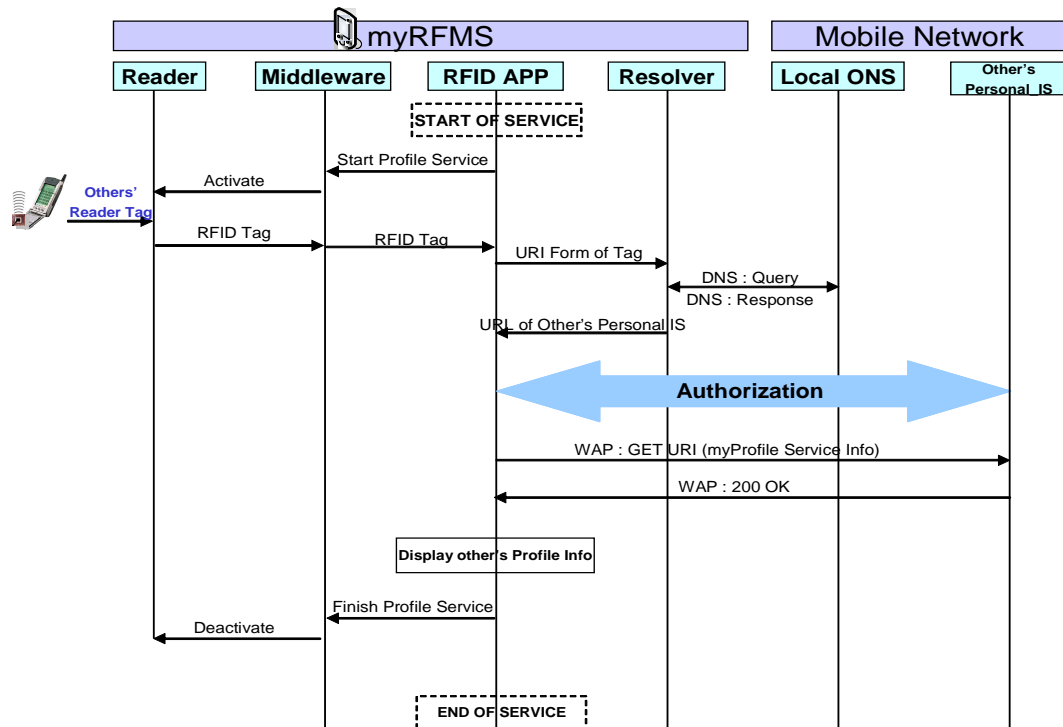


그림 3-7 GetProfile Service Feature

- ① 한 가입자가 같은 이동 RFID 네트워크내의 RFID tag를 가진 타인에 대한 정보를 알고자 할 때 가입자의 RFMS의 리더는 타인의 Tag 정보를 읽어 들여 이를 Application에게 알려준다.
- ② 타인의 Tag를 인식한 RFID의 Application는 Resolver를 통해 이동 RFID 네트워크의 ONS system에게 해당 정보를 가지고 있는 Personal-IS의 위치정보를 요청하고 응답을 반환 받는다.
- ③ 인증 과정을 통해 접근이 가능하다고 판단되면 RFMS의 Application는 타인의 정보를 해당 Personal-IS에게 요청하고 이에 대한 응답을 받아 화면에 출력함으로써 모든 과정이 종료된다.





- ④ EPC-IS는 이에 대한 응답을 Proxy를 거쳐 HTTP 기반에서 WAP 기반에 맞게 HTML/XML에서 WML 형식으로 변환하여 해당 제품에 대한 정보를 RFMS의 Application에게 전송하다.
- ⑤ RFMS의 Application은 제품에 대한 정보를 저장하기 위해 가입자의 해당 Personal-IS의 위치를 알기 위해 Resolver를 통해 이동 RFID 네트워크의 ONS system에 질의하여 응답을 반환 받는다.
- ⑥ RFMS의 Application은 이렇게 받은 위치정보를 이용하여 해당 제품에 대한 정보와 시간, 장소 등을 저장하고 응답을 받으면 모든 과정을 종료한다.

## 나. ReviewInterests Service Feature

그림 3-9은 가입자가 예전에 등록했던 물품들에 대한 정보를 다시 한번 보길 원할 때의 상황을 나타낸다.

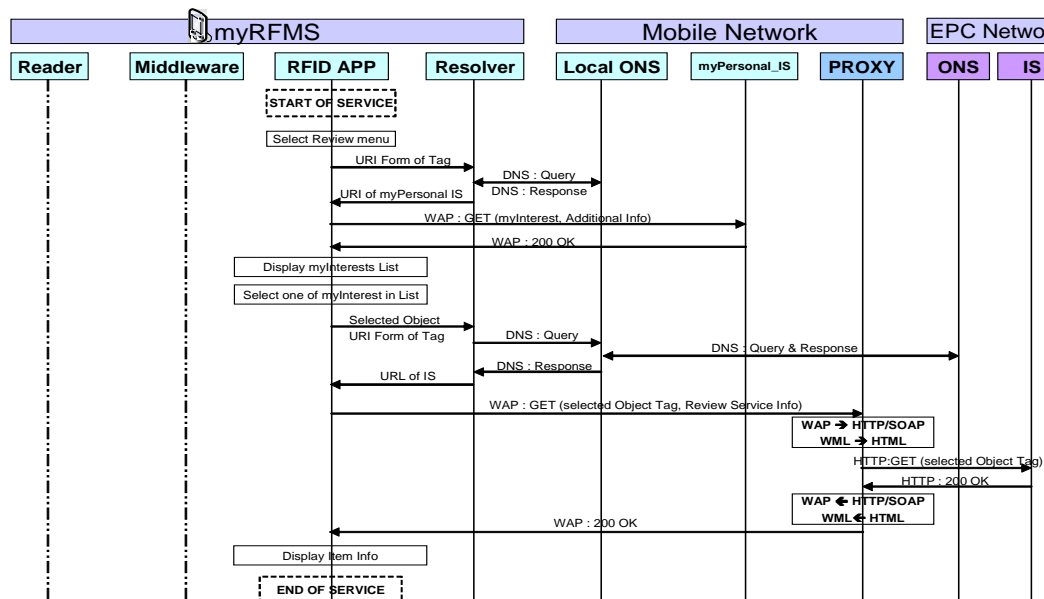


그림 3-9 ReviewInterests Service Feature

- ① 가입자는 자신의 Personal-IS를 통하여 저장된 목록 정보를 살펴본다
- ② 목록에서 원하는 물품을 선택하고 지정한 물품의 정보에 대한 질의 요청을 한다

- ③ 선택한 물품의 태그 정보를 이용하여 Local ONS를 통하여 EPC-IS의 위치 정보를 알아온다.
- ④ 얻은 위치정보를 가지고 EPC Proxy를 통해서 EPC-IS로부터 원하는 정보에 대한 질의요청을 한다.
- ⑤ 다시 EPC Proxy로부터 프로토콜 및 컨텐츠 변환을 거쳐 가입자의 RFMS 화면에 선택한 물의 정보를 표시한다.
- ⑥ 가입자는 물품의 정보를 확인하고 서비스 종료하거나, 기타 다른 응용된 서비스로 연계될 수 있다.

## 5. Visits repository를 이용한 Service Feature

### 가. RegisterVisit Service Flow

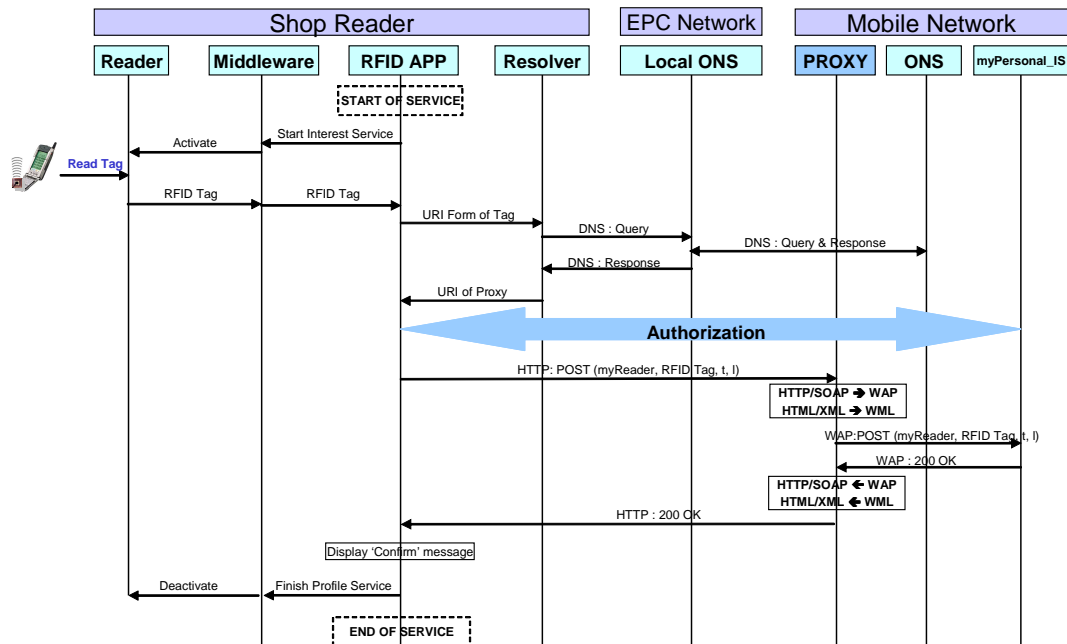


그림 3-10 RegisterVisit Service Flow

- ① 그림 3-10 은 상점에서 고객이 들어왔을 때 해당 고객에 대한 정보를 얻고자 하는 과정을 도식화 한 것 이다.
- ② 신분 증명용 RFID Tag를 소지한 고객이 상점에 들어왔을 때 상점은 고객에 대한 정보를 알고자 Tag정보를 읽어 들인다.

- ③ 상점의 Application은 해당 고객의 정보를 알고자 EPC 네트워크 내의 ONS system을 거쳐 이동 RFID 네트워크의 ONS system으로 접근하여 고객의 정보가 저장된 Personal-IS 의 위치 정보를 반환 받는다. 이 과정에서 이동 RFID 네트워크에 속한 Proxy의 위치정보 역시 전송 받는다.
- ④ 인증과정을 통해 해당 상점이 고객의 정보를 받아 볼 수 있음이 확인되면 상점의 RFID Application은 Proxy를 거쳐 WAP 기반에 맞게 HTML/XML 에서 WML 형식으로 변환하여 Personal-IS에게 고객정보를 요청한다.
- ⑤ 요청을 받은 Personal-IS는 Proxy를 통해 HTTP기반에 맞게 WML에서 HTML/XML 형식으로 변환하여 상점의 Application에게 응답을 반환하고 모든 과정이 종료된다.

## **나. ReviewVisits Service Feature**

Review Service Feature는 그림 3-10 에서 등록된 자주 찾는 장소에 대해 다시 정보를 확인하고자 할 때 이루어지는 서비스이다. Review Service Feature의 그림은 위의 서비스 ReviewInterest()의 상황과 흡사하며 이루어지는 동작 역시 같다고 볼 수 있다. 단, 사용자의 화면상의 서비스 목록, 선택과정과 확인 과정의 표현상(Display)에 차이는 있을 수 있으나, 내부에서 진행되는 일련의 처리과정은 같다.

## 제4장 EPC Network와 Mobile RFID Network간 서비스 연동

### 제1절 통상적인 EPC 네트워크 서비스

EPCglobal에서 제시하는 EPC 네트워크 서비스는 그 범위가 EPC 네트워크로 제한되며, 기존 사용되던 DNS를 확장하여 ONS로 활용하고자 하는 방안을 가지고 있다. RFID Code는 DNS 위임체계와 동일한 방법으로 RFID Code 위임체계가 구성되며, RFID Resolver를 통하여 원하는 RFID Code와 매핑되는 정보를 EPC-IS를 통해 서비스 받을 수 있다.[1][2][3][4][6]

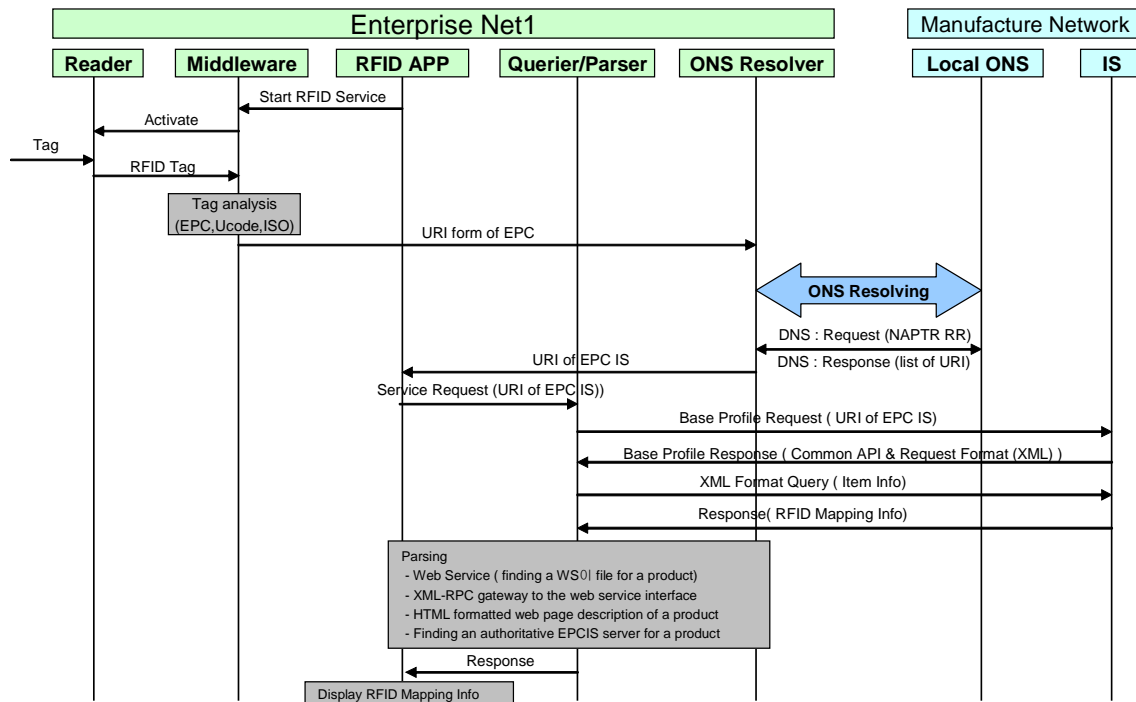


그림 4-1 EPC Service

기존 인터넷망에서 사용되는 프로토콜 및 콘텐츠를 그대로 휴대폰으로 적용시키는 것은 불가능 하다. 따라서, 두 망간의 효율적이고 원활한 정보의 교환과 공유가 이루어지기 위해

서는 EPC-IS Conversion 서비스가 필요하다.

## 제2절 이동통신망에서 정보검색 서비스 구조

### 1. 무선인터넷 플랫폼

WAP(wireless application protocol)은 모토로라, 노키아, 에릭슨, Phone.Com등이 주축이 되어 결성하고, 전 세계 200여 개 업체가 회원으로 참여하고 있는 WAP 포럼이 무선 인터넷을 위하여 정의한 프로토콜이다. 그림 4-2와 같이 단말기에 탑재된 WAP 브라우저가 WAP 게이트웨이를 통해 WML/WML Script로 구성된 콘텐츠에 접근하게 된다. 그 중간 과정으로 WAP 게이트웨이가 인터넷망과의 연결을 담당한다. WAP 게이트웨이와 단말기는 WDP/WTP/WSP로 구성된 WAP 프로토콜을 통해 데이터통신을 한다.

WAP 게이트웨이는 인터넷의 데이터 통신을 위해 정의된 HTTP 프로토콜과 무선 네트워크의 WSP 프로토콜 변환 기능을 함으로써 인터넷의 콘텐츠 서버의 데이터를 단말기에 보여줄 수 있다. 이것은 단말기가 인터넷에서 제공하는 서비스를 받기위한 단방향 흐름이며, 그 역방향은 제공해 주지 않는다.

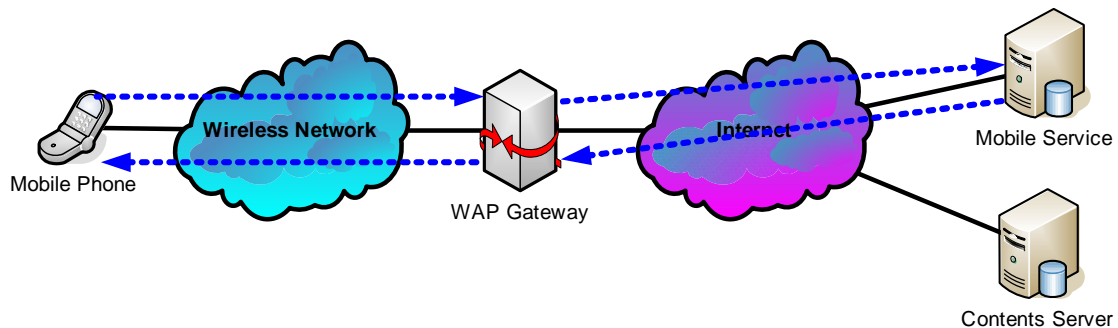


그림 4-2 통상적인 이동통신망에서 정보검색 서비스 구조

### 2. WSP(Wireless Session Protocol) 개요

WSP는 인터넷의 HTTP프로토콜의 기능을 Mobile환경에 맞게 설계한 프로토콜이다. HTTP는 TCP/IP위에 동작되며, WSP는 WTP(Wireless Transfer Protocol)위에서 동작한다. 그러나 WAP 2.0에서는 하부 레이어를 TCP/IP로 설계하는 아키텍처가 제시되었으며, 그림 4-3과 같

은 전형적인 WAP 프로토콜 레이어 보다는 TCP/IP 레이어 위에 세션을 관리하는 WSP만을 사용하는 레이어 환경으로 바뀌고 있다.[23]

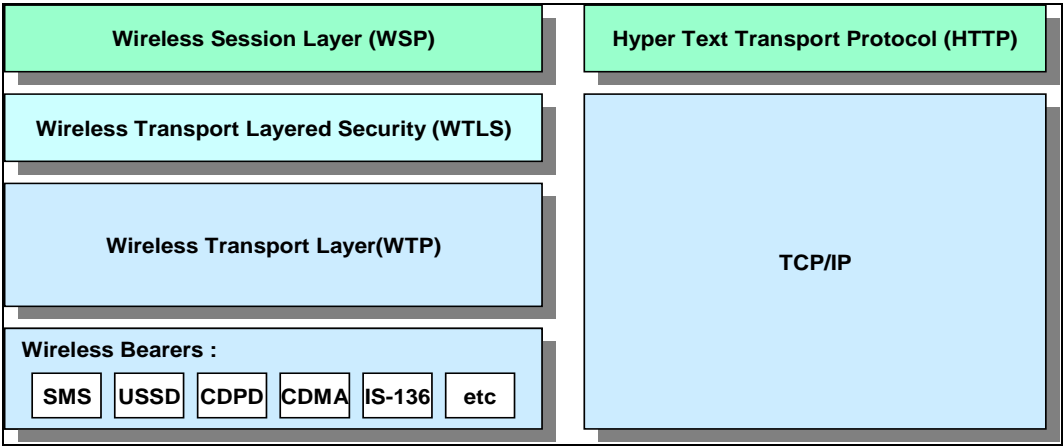


그림 4-3 WAP 과 인터넷

WSP는 무선대역폭을 효과적으로 사용하기 위해 개발되었다. 인터넷에서의 HTTP 프로토콜에서 지원하는 모든 메소드를 Binary Format으로 지원하며, Mobile환경만을 위한 기능들도 정의하고 있다. WSP는 크게 두 가지 Connection Mode와 Connectionless Mode로 설계가 가능하다. 두 가지 Mode의 차이점은 Connection Mode는 TCP의 PCB(Process Control Block)처럼 세션간의 Session State Table을 통해 연결 신뢰성 및 Idle상태의 관리한다. 이에 반해 Connectionless Mode는 세션간의 어떠한 Management를 하지 않기 때문에 UDP와 같은 기능을 한다. 본 논문에서는 WSP를 Connectionless Mode로 설계하였다.[9]

WSP는 Mobile 환경에 적합하게 Data Unit Structure와 Encoding 방법을 규정하고 있다. 데이터 포맷을 표현하는데 있어 표 4-1과 같은 Data Type이 존재한다.

표 4-1 Primitive Data Types

Data Type	Definition
bit	1 bit of data
octet	8 bits of opaque data
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
uintvar	Variable length unsigned integer

WSP Message들의 데이터 타입은 대부분 uintvar 타입을 사용한다. 이 타입은 1byte의 데이터를 “Continue bit(1bit) + Payload(7bit)”로 표시한다. 만약 0x87A5(1000 0111 10101 0101)로 어떤 데이터가 인코딩 되었다면 그림 4-4와 같이 표현된다.[9]

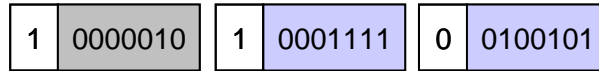


그림 4-4 uintvar data format

## 가. Protocol Data Unit Structure

### 1) PDU Common Fields

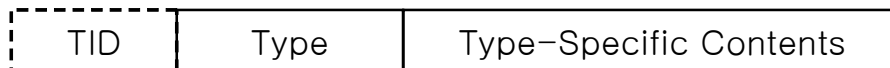


그림 4-5 WSP PDU

그림 4-5은 WSP의 PDU를 나타내며, 필드에 대한 기능은 다음과 같다.

- TID 필드는 Connectionless Mode에서 WSP Request/Response 패킷을 구분하기 위해 사용한다.
- Type 필드는 WSP 패킷의 종류를 나타낸다. 주로 사용되는 Type은 HTTP에서 데이터의 요청과 응답시 사용되는 GET과 Reply 메서드를 많이 사용하며, 그림 4-6의 PDU Type과 같이 HTTP의 모든 메서드들을 WSP에서 지원한다.

Name	Assigned Number
<i>Reserved</i>	0x00
Connect	0x01
ConnectReply	0x02
Redirect	0x03
Reply	0x04
Disconnect	0x05
Push	0x06
ConfirmedPush	0x07
Suspend	0x08
Resume	0x09
<i>Unassigned</i>	0x10-0x3F
Get	0x40
Options (Get PDU)	0x41
Head (Get PDU)	0x42
Delete (Get PDU)	0x43
Trace (Get PDU)	0x44
<i>Unassigned (Get PDU)</i>	0x45-0x4F
<i>Extended Method (Get PDU)</i>	0x50-0x5F
Post	0x60
Put (Post PDU)	0x61
<i>Unassigned (Post PDU)</i>	0x62-0x6F
<i>Extended Method (Post PDU)</i>	0x70-0x7F
Data Fragment PDU	0x80
<i>Reserved</i>	0x81-0xFF

그림 4-6 PDU Type Assignments

## 나. Main PDU 타입

### 1) GET Method PDU

Name	Type	Source
URILen	uintvar	Length of the URI field
URI	URILen octets	S-MethodInvoke.req::Request URI
Headers	Multiple octets	S-MethodInvoke.req::Request Headers



## 2) Reply Method PDU

Name	Type	Source
Status	uint8	S-MethodResult.req::Status
HeadersLen	uintvar	Length of the ContentType and Headers fields combined
ContentType	Multiple octets	S-MethodResult.req::Response Headers
Headers	(HeadersLen-length of ContentType) octets	S-MethodResult.req::Response Headers
Data	Multiple octets	S-MethodResult.req::Response Body

### 다. Header Encoding

WSP는 HTTP/1.1 헤더필드와 호환되게 각 Header Name과 Header Value들에 대한 매핑코드들을 WSP 스펙에 정의하고 있다. Well-Known Header Name 과 Well-Known Header Value은 그림 4-7와 같이 Binary vlaues로 매핑코드가 정의되어 있으며, 그 이외의 Header Name과 Header Value들은 0x00(NULL)로 처리함으로써 불필요한 정보를 인코딩하지 않는다.[9]

HTTP/1.1 header:	Accept: application/vnd.wap.wmlc
Encoded header:	
0x80	-- Well-known field name "Accept" coded as a short integer
0x94	-- Well-known media "application/vnd.wap.wmlc" coded as a short integer

그림 4-7 Header Encoding

## 제3절 서비스 연동을 위한 요구사항

RFID Network에서의 사용자는 EPC Network의 서비스 이용할 수 있고, EPC Network의 사용자는 RFID Network의 서비스인 IS, DS를 이용할 수 있다. 이를 위하여, 양단간의 프로토콜 및 콘텐츠를 변환시켜 주는 EPC Proxy가 필요하고, 양단의 Local ONS는 EPC net에서 mobile operator의 Local ONS로 Personal-IS에 대한 resolution request가 올 때는 해당 EPC Proxy의 URI로 변환할 수 있어야 하고, Personal-IS 등록시, 이에 대한 URI 뿐 아니라 EPC Proxy로 redirect하도록 EPC Proxy의 URI에 대한 NAPTR RR를 작성하여 Local ONS 저장해 두어야 한다.

## 제4절 EPC-Proxy: Contents와 Protocol의 변환

EPC-Proxy는 이동 RFID 네트워크와 EPC 네트워크의 매개체로써, 두 네트워크의 서비스 호환 및 상호운영에 필요한 시스템이다.

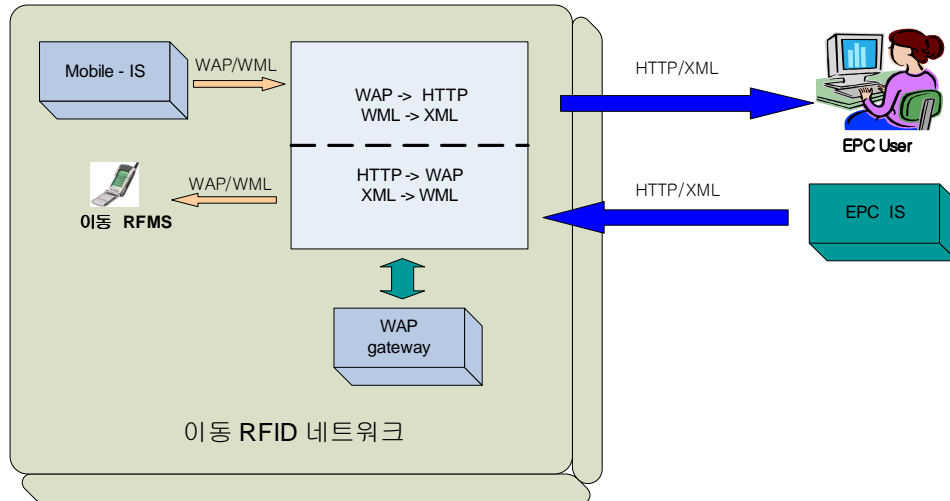


그림 4-8 EPC-Proxy의 기능

그림 4-8은 이동 RFID 네트워크와 EPC 네트워크 사이의 프로토콜 변환과 실 데이터 포맷의 변환을 통한 각 End-User가 어떤 네트워크를 통해 서비스를 요청하거나 원하는 서비스를 받을 수 있는 것을 보여주고 있다. Personal-IS를 이용하기 위해서는 WAP 프로토콜로 서비스 요청이 이루어져야 하며, EPC-IS를 이용하기 위해서는 HTTP프로토콜로 서비스 요청이 이루어져야 한다. 이런 동작이 가능하기 위해서 중간 매개체 역할을 하는 EPC-Proxy는 어느 네트워크의 누가 서비스 요청을 했는지, 어디로 서비스요청을 하는지에 대한 연결상태 정보를 저장하며, 프로토콜 변환과 데이터포맷의 변환과정을 수행해야 한다. EPC-Proxy의 가장 큰 역할은 EPC 네트워크의 구성과는 별개로 이동 RFID 네트워크를 구축하여 EPC 네트워크가 제공하는 서비스를 이동 RFID 네트워크 End-User에게 제공할 수 있으며, 그 역 과정도 가능하게 해준다는 것이다. EPC-Proxy를 이용하기 위해서 이동 RFID 네트워크가 관할하는 RFID 태그에 대해서 ONS의 Zone 파일에는 하나의 태그 클래스당 2개씩의 NAPTR RR이 존재해야 한다.

## 제5절 ONS를 이용한 Configuration Hiding

### 1. ONS Zone 파일 구성

표 4-2은 이동 RFID 네트워크의 Local ONS의 Zone파일 구성방법을 나타낸다. EPC User가 이동 RFID 네트워크의 Personal-IS로 서비스 요청 시, EPC-Proxy를 통해 서비스가 요청 되어 하며 원하는 데이터를 전송 받아야 한다. EPC User Application이 Personal-IS의 URL을 알기 위해 ONS리졸빙을 거치면, 표 4-2의 서비스 필드가 “epc+epcis”인 NAPTR RR을 리턴 받는다. “rfms+...”은 EPC 네트워크에서 정의한 서비스타입이 아니므로 리졸빙 절차 시 NAPTR RR에 포함되지 않고 “epc+...”로 정의된 서비스의 NAPTR RR만이 반환된다.

표 4-2 이동 RFID 네트워크의 Local ONS의 Zone파일 구성

Order	Pref	Flags	Service	Regexp	Replacement
0	0	u	epc+epcis	URL_EPC-Proxy/ <b>DEST=URL_Personal-IS</b>	.
0	0	u	rfms+epcis	URL_Personal-IS	.

어플리케이션은 리턴받은 NAPTR RR의 Regexp필드에 정의된 EPC-Proxy의 URL로 연결요청을 하며, 이동 RFID 네트워크의 Personal-IS로 연결할 수 있도록 “DEST=URL\_Personal-IS”를 파라미터로 전달한다. EPC-Proxy는 전달받은 파라미터를 통해 Personal-IS에 서비스요청과 EPC 네트워크의 서비스요청 어플리케이션에 Personal-IS의 서비스를 제공해준다. 서비스 필드가 “rfms+...”으로 정의된 NAPTR RR은 이동 RFID 네트워크의 이동 RFMS가 이동 RFID 네트워크의 Personal-IS의 서비스를 이용할 때 리졸빙 결과로 리턴되는 NAPTR RR이다. EPC-Proxy를 거치지 않고 서비스 되어 하는 서비스이므로, 바로 Personal-IS의 URL만을 제공한다.

### 2. Mobile RFID 네트워크의 어플리케이션 알고리즘

그림 4-9은 이동 RFID 네트워크의 Application의 동작 알고리즘으로, 이러한 알고리즘이 필요한 이유는 EPC 네트워크 Application은 원하는 태그정보에 대한 리졸빙 결과 서비스 필드가 “rfms+...”로 되어진 NAPTR RR은 인식되지 않는 서비스이므로 리턴받지 않지만, 이동 RFID 네트워크의 Application은 리졸빙 결과 하나의 RFID 태그에 서비스타입이 “rfms+...”, “epc+...” 인 NAPTR RR을 리턴 받는다.

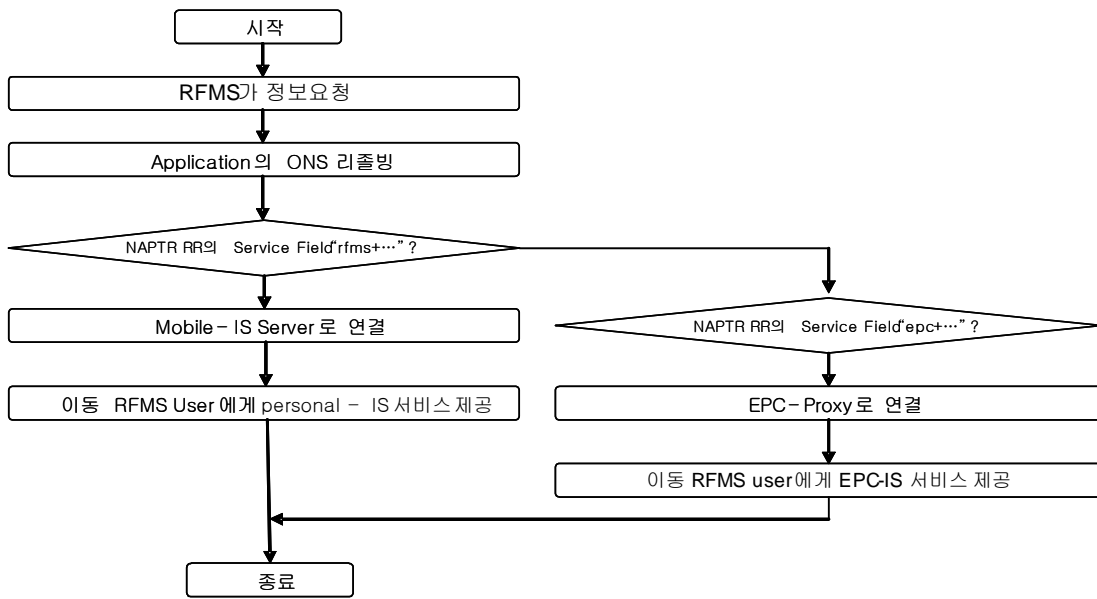


그림 4-9 이동 RFID 네트워크의 Application 알고리즘

두 RR중 “rfms+...”의 NAPTR RR을 선택할수 있게끔 알고리즘이 구성되어야 한다. 이와 같이 이동 RFID 네트워크가 관리하는 RFID 태그에 대해서 ONS의 존파일 관리하고, EPC-Proxy를 거치면 두 네트워크의 프로토콜과 데이터 포맷의 제약없이 두 네트워크의 서비스 호환이 가능하다.

## 제6절 서비스 연동 시나리오

### 1. 이동 RFID 네트워크에서 EPC 네트워크로 정보흐름

그림 4-10는 리더기가 장착된 핸드폰을 이용하여 EPC 네트워크의 정보를 얻고자 할 때의 과정(예: 야외에서 맘에 드는 제품을 보았을 때 해당 제품의 사이즈 및 소재 등의 제품 정보를 알고자 할 때)을 도식화 한 것이다.

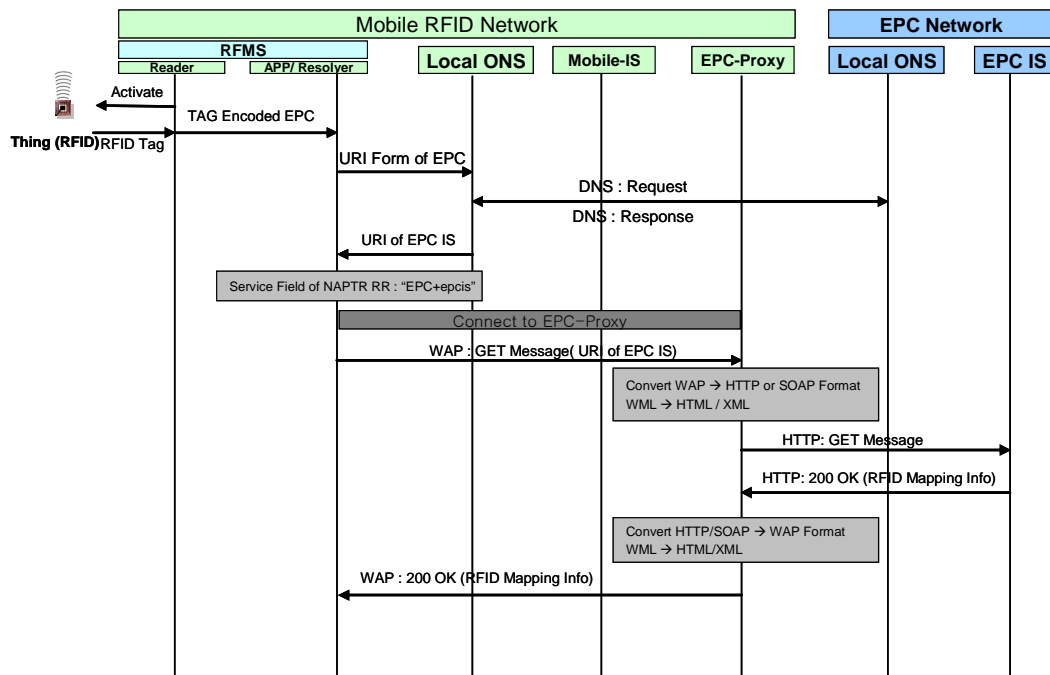


그림 4-10 이동 RFID 네트워크 -> EPC 네트워크

- ① 리더기는 해당 Tag의 정보를 읽어 들여 이를 Resolver에게 전송한다.
- ② 이렇게 읽어 들인 tag정보를 이용하여 원하는 정보가 어디 있는지 알기 위해 RFMS는 해당 Local ONS에게 해당 URI형태로 태그정보를 전송하고, Local ONS는 EPC 네트워크의 ONS에 접근하여 태그 정보를 가지고 있는 EPC-IS가 어디에 있는지 위치 정보를 질의한다.
- ③ EPC 네트워크 내의 ONS 시스템은 요청에 대해 해당 EPC-IS 내의 위치를 이동 네트워크의 Local ONS에게 반환하고 Local ONS는 이를 다시 RFMS의 Resolver에게 반환한다.
- ④ 위 과정을 통해 얻은 EPC-IS의 URL을 통해 EPC-Proxy로 연결요청을 한 후 WAP 프로토콜의 SetVar() 메서드를 사용하여 EPC-IS의 URL을 전달해 준다.
- ⑤ WAP을 기반으로 하는 이동 RFID 네트워크 환경과 HTTP/SOAP 기반의 EPC 네트워크 환경의 연동을 위해 Proxy를 두어 이동 망의 Local ONS로부터 들어온 내용을 HTTP/SOAP 형식에 맞게 HTML/XML로 바꾸어 원하는 EPC - IS로 요청 Message를 전송한다.

- ⑥ 해당 요청에 대한 응답을 EPC-IS는 EPC-Proxy로 전송하여 WAP 환경에 맞게 WML로 변환하고, EPC-Proxy는 해당 정보를 RFMS의 Application에게 전송한다.

## 2. EPC 네트워크에서 이동 네트워크로 정보흐름

그림 4-11은 EPC 네트워크에 연결된 하나의 단말이 이동 RFID 네트워크로부터 원하고자 하는 정보를 획득하는 과정(예: 가정에서 자신이 RFMS를 이용한 결제 내역을 알고자 할 때)을 도식화 한 것이다.

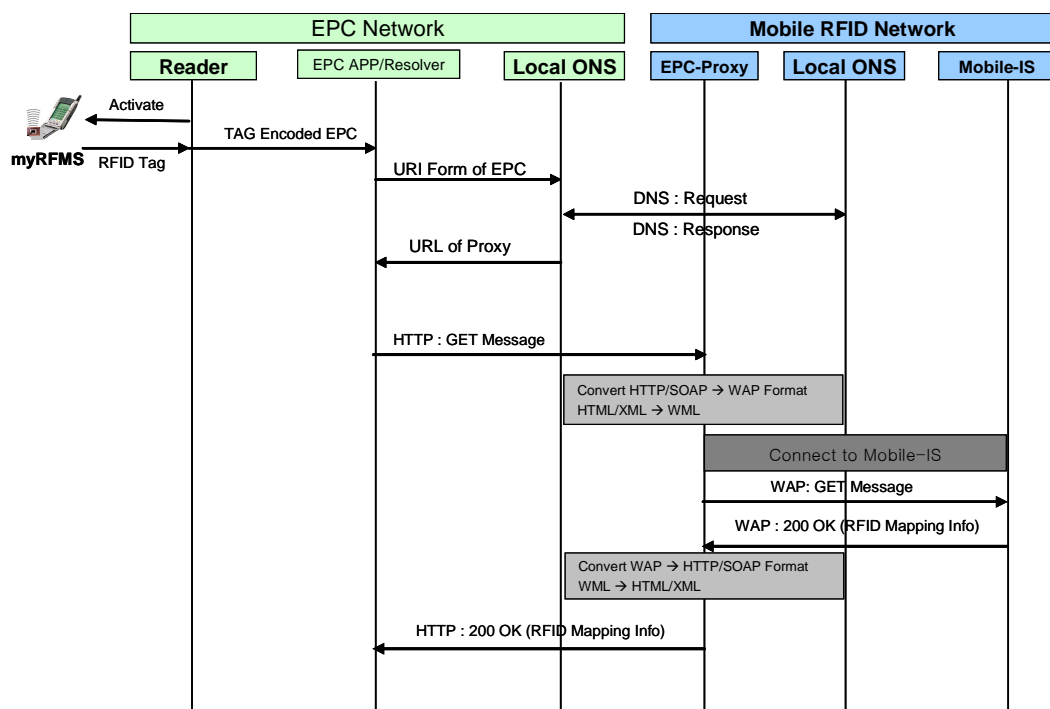


그림 4-11 EPC 네트워크 → 이동 RFID 네트워크

EPC 네트워크에 연결된 리더기를 이용하여 해당 Tag 정보를 얻어오면 리더기는 이를 해당 EPC Resolver에게 전달한다. 전달 받은 URI폼의 Tag 정보를 이용하여 ONS의 위임체계에 따라 리졸빙을 한다.

이동 RFID 네트워크의 Local ONS는 제공하고자 하는 정보를 가진 Personal-IS의 URL을 EPC 네트워크의 Local ONS에게 반환하고, EPC 네트워크의 Local ONS는 이를 다시 Resolver에게 전송한다. Resolver가 전달받은 NAPTR RR은 EPC-Proxy의 URL과 HTTP 파라미터 전달방

식으로 구성된 텍스트 문자열을 가지고 있다. 어플리케이션에서 EPC-Proxy로 정보 요청시 파라미터로 전달되는 Personal-IS의 URL을 통해 EPC-Proxy는 해당 Personal-IS로 연결을 수행한다.

- ① 요청 message를 받은 EPC-Proxy는 해당 내용을 HTTP 환경에 맞게 변환하여 HTML/XML 언어로 해당 연결된 Personal-IS에게 요청 Message를 전송한다.
- ② 이동 네트워크 내의 Personal-IS는 해당 요청에 대한 응답을 EPC-Proxy에게 전송하고 EPC-Proxy는 이를 WAP 환경에 맞게 WML 언어로 변환하여 원하는 Application에게 전송한다.

## 제5장 EPC Network와 서비스 연동 가능한 Mobile RFID 네트워크 구성요소 구현

### 제1절 ONS Resolver 설계 및 구현

#### 1. DNS를 이용한 Object Name Resolution

본 논문에서 구현되는 RFID Client는 리더기가 읽어 들인 해당 EPC에 대해 tag conversion 형식에 맞게 URI로 변경되어 입력이 이루어진 상황을 가정하고 설계되었다. 본 Application은 입력 받은 URI를 이용하여 해당 EPC에 매칭되는 EPC-IS의 리스트를 출력한다. 이중 사용자가 EPC-IS의 URL을 선택함으로써 원하고자 하는 data의 종류를 결정하여 그에 따른 해당 정보를 출력한다.[1][2]

##### 가. DNS 메시지 형식

도메인 프로토콜 내부의 모든 통신은 메시지라고 하는 단일 형식에 담겨 전송된다. ONS Resolving을 위해 사용되는 DNS 메시지의 형식은 그림 5-1과 같이 5개의 부분으로 이루어져 있다. 이 중, 일부는 경우에 따라 비어있을 수도 있다.

헤더 부분은 메시지의 다른 부분들 중 어떤 것들이 존재하는 지를 나타내는 필드를 가지고 있다. Question 부분은 네임 서버에게 질문을 하는 필드를 갖고 있다. 이 필드들은 질의 종류(QTYPE), 질의 클래스(QCLASS), 및 질의 도메인 네임(QNAME)으로 구성된다. 마지막 세

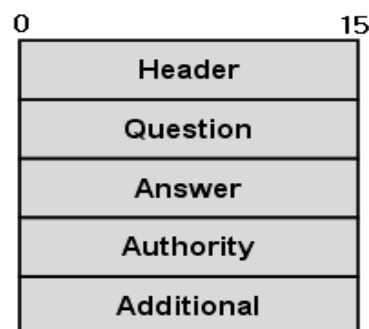


그림 5-1 DNS 메시지 포맷



부분들은 형식이 같은 부분으로 연결된 리소스 레코드(RR)들을 갖고 있다. Answer 부분은 질의에 대한 응답하는 리소스 레코드들을 갖고 있고 Authority 부분은 권한을 갖고 있는 네임 서버를 가리키는 리소스 레코드를 가지고 있다. 마지막 Additional 부분은 질의에 대한 대답은 아니지만 질의와 밀접한 관계가 있는 리소스 레코드를 가지고 있다.[7][8]

## 1) Header Section

DNS메시지의 헤더 부분은 메시지의 메시지가 질의인지 응답인지, 표준 질의 인지 또는 기타 다른 opcode인지를 나타낸다. 헤더 부분의 포맷은 그림 5-2와 같다.[19]

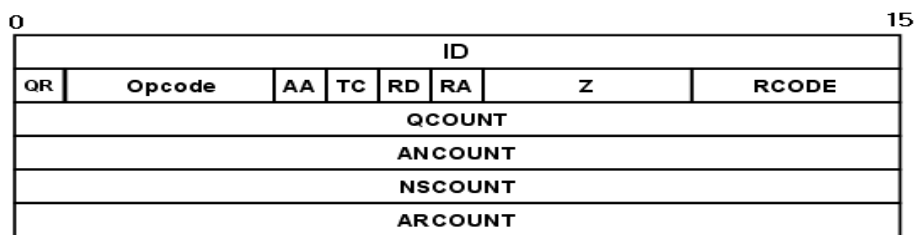


그림 5-2 DNS 메시지 Header section 포맷

헤더의 각 필드가 나타내는 의미는 다음과 같다.

- ID : 임의의 질의를 생성하는 프로그램에 의해 할당되는 16비트의 인식자이다. 질의에 대한 응답 속에 이 인식자가 복사되어, 질의를 보낸 측이 어떤 질의에 대한 응답인지를 식별한다.
- QR(Query or Response) : 메시지가 질의(0)인지 응답(1)인지를 나타낸다.
- Opcode : 메시지 내의 질의의 종류를 나타내는 필드이다.
- AA(Authoritative Answer) : 응답하는 네임 서버가 질문 부분의 도메인 네임에 대한 권한을 갖고 있으면 이 비트가 설정된다.
- TC(Truncation) : 전송 채널이 허용하는 크기보다 메시지가 커서 메시지가 잘려졌음을 나타낸다.
- RD(Recurision Desired) : 이 비트는 질의 내에서 설정되며 응답으로 복사된다. 만약 RD가 설정되어 있으며 네임서버에게 재귀적 질의를 처리하도록 지시한다.
- Z : 나중에 위해 예약되어 있는 부분으로 질의와 응답에서 모두 0 으로 설정된다.

- RCODE(Response Code) : 4비트 필드로서 응답의 일부분이다.
- QDCOUNT : 질문 부분(Question Section) 내에 있는 항목들의 개수를 나타낸다.
- ANCOUNT : 질문 부분(Question Section) 내에 있는 리소스 레코드의 개수를 나타낸다.
- NSCOUNT : 권한 부분(Authority Section) 내에 있는 네임서버 리소스 레코드의 개수를 나타낸다.
- ARCOUNT : 기타 부분(Additional Section) 내에 있는 리소스 레코드의 개수를 나타낸다.

## 2) Question Section

DNS메시지의 질문 부분(Question Section)은 질의 내의 ‘질문’ 즉, 무엇을 물어보는지 정의하는 인자들은 운반하는데 이용된다. 질문 부분의 형식은 그림 5-3과 같다.[8][19]

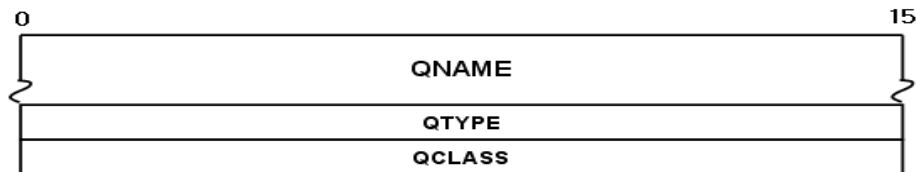


그림 5-3 DNS 메시지 Question section 포맷

질문 부분의 각 필드가 나타내는 의미는 다음과 같다

- QNAME : 레이블들의 나열로 표현된 도메인 네임으로서, 각 레이블은 하나의 길이 옥텟과 그 숫자만큼의 옥텟들로 구성되어 있다.
- QTYPE : 질의의 종류를 나타낸다.
- QCLASS : 질의의 클래스를 나타낸다.

## 3) Answer, Authority 및 Additional Section

DNS메시지의 대답(Answer), 권한(Authority) 및 기타(Additional)부분은 모두 동일한 형식을 가지고 있으며, 형식은 그림 5-4와 같다.[8][19]

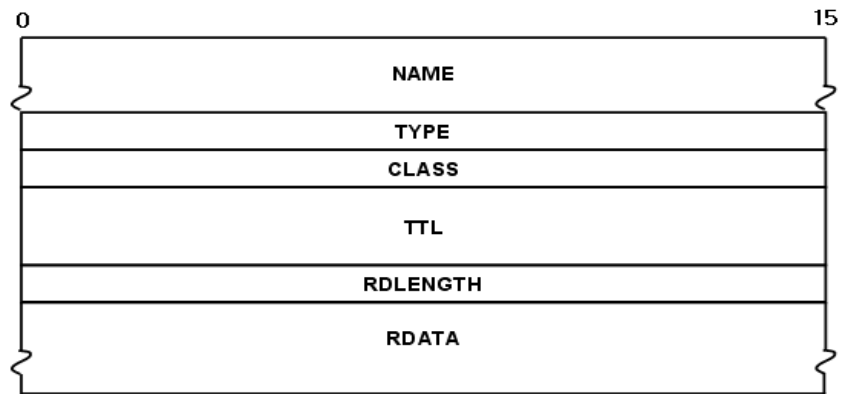


그림 5-4 DNS 메시지 Answer, Authority 및 Additional Section 포맷

각 필드의 의미는 다음과 같다.

- NAME : 리소스 레코드에 관계된 도메인 네임
- TYPE : 리소스 레코드 종류 코드 중의 하나를 갖고 있는 필드로 RDATA 필드에 있는 데이터의 의미를 나타낸다.
- CLASS : RDATA 필드에 있는 데이터의 클래스를 나타낸다.
- TTL : 리소스 레코드가 캐시에 남아있을 초 단위의 시간 간격을 나타낸다.
- RDLENGTH : RDATA 필드의 옥텟들의 길이를 나타낸다.
- RDATA : 리소스 레코드를 설명하는 가변 길이의 문자열이다. 이 정보 형식은 리소스 레코드의 TYPE 및 CLASS에 따라 달라진다.

## 나. NAPTR Resource Record

NAPTR RR 과정은 rewriting과정을 통해 DNS 데이터부분의 변경과 재위임을 통해 보다 편리하고 정확한 응답을 받기 위해 정의 되었다. 결론적으로 데이터는 정규 표현식을 이용하여 다량의 정보가 오히려 작은 DNS패킷에서 코드화되어 높은 효율성과 간결성을 가지게 되었다.

NAPTR RR type 으로 들어오는 응답은 일반적인 DNS 메시지 형식과 동일하다. 다만 그 데이터의 저장된 내부 값이 구분 되어 있다는 점이 다를 뿐이다.[20]

그림 5-5 은 DNS의 응답으로 들어온 값의 header 의 필드 와 그 내용 값을 보여준다. ID 부분은 전송하는 query의 ID와 같은 값으로 보내진다. 만약 query패킷의 값과 다른 값이

라면 해당 NAPTR RR은 다른 query에 대한 응답이다. QR필드는 NAPTR RR이 응답의 한 종류이므로 1로 세팅된다. QDCOUNT역시 query의 개수만큼 세팅되어 전송된다. ANCOUNT는 query에 대한 응답의 개수를 값이 세팅된다. 만약 해당 질의에 대한 응답이 존재하지 않는다면 ‘0’으로 세팅되어지며 Answer section이 전송되지 않는다. NSCOUNT는 query에 대한 응답을 가지고 있던 DNS 네임서버의 개수를 보여주며 Authority name server section의 내용의 개수를 의미한다. ARCOUNT는 추가 정보에 대한 개수를 의미한다.[20]

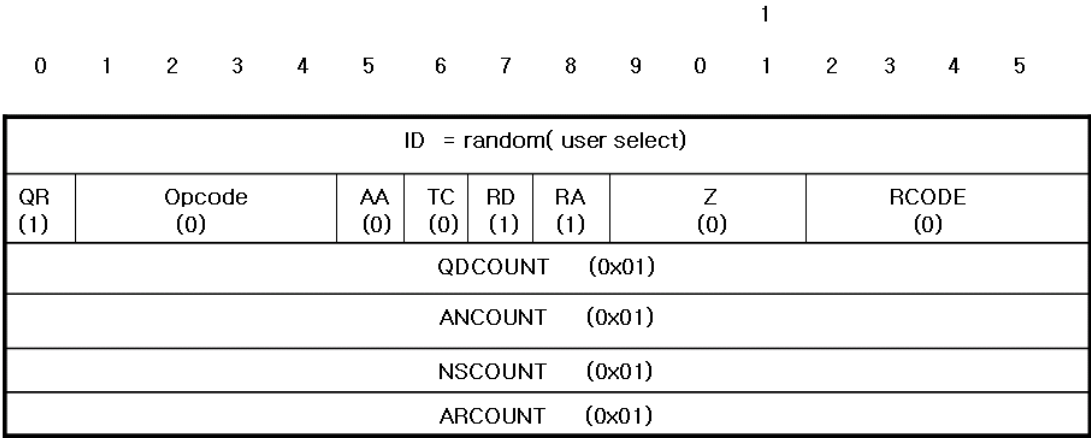


그림 5-5 NAPTR RR header

그림 5-6은 DNS 응답 시의 answer section 부분의 필드 정의와 입력 값을 보여준다.

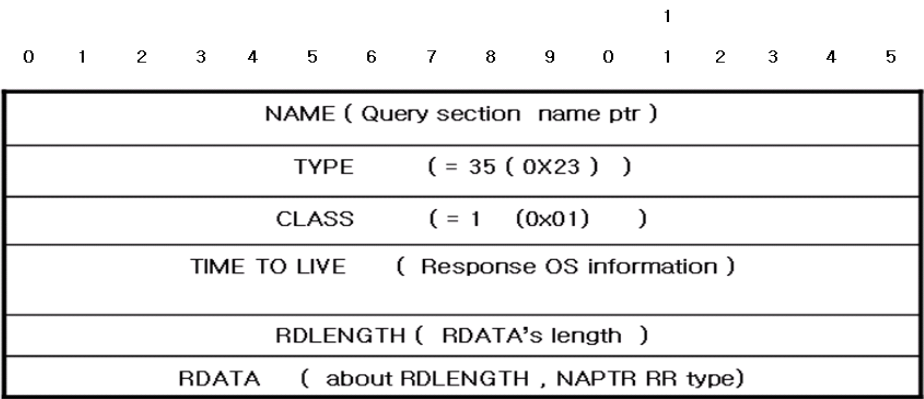


그림 5-6 NAPTR Answer section

그림 5-7은 RDATA 내부에서의 데이터값 필드와 그 내용을 보여준다. ORDER는 각각의 DNS 응답의 우선순위를 의미한다. Preference는 같은 동일한 order에 대한 우선순위를 결정해 주기 위해 사용된다. FLAG는 해당 응답의 type을 알려준다 FLAG는 레코드 내의 다른 필드의 해석에 영향을 주기 때문에, 만약 레코드에 알 수 없는 FLAG 값이 있는 경우에는 반드시 레코드 전체를 건너뛰어야 한다. 이 필드는 ORDER 값보다 우선한다. FLAG는 단일 문자로 표시되며 문자 집합은 [A-Z, 0-9]가 된다. FLAG 필드는 다음과 같은 4개의 값들을 갖는다.

- “S”- 다음 lookup 이 DNS SRV(DNS specifying the location of services) 레코드임을 의미한다.
- “A”- 다음 lookup 이 A, AAAA, A6 레코드임을 의미한다.
- “U”- 다음 과정은 DNS lookup 이 아니라 Regexp 필드의 결과임을 나타낸다.
- “P”- 이후의 과정이 프로토콜에 의존적인 것이므로 더 이상 DNS 질의를 하지 않아야 한다는 것을 의미한다.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
ORDER ( = setting value : zone file 's order field )															
Preference ( = setting value : zone file's pref field )															
FLAG ( = setting value : zone file's flag field )															
SERVICES( = setting value : zone file's services field )															
REGXP( = setting value : zone file's regxp field )															
REPLACEMENT( = setting value : zone file's repl field )															

그림 5-7 NAPTR RR 의 RDATR 의 각 필드들

## 2. ONS Resolver 구현

ONS Resolving은 그림 5-8와 같이 URI형식의 입력된 RFID Code 값을 Domain Name포맷으로 변경하는 과정을 거친다. 이렇게 변경된 데이터의 정보를 DNS 시스템으로 질의하기 위해 알맞은 헤더를 추가하여 ONS에게 질의한다. 이 헤더 내용은 DNS spec 에 맞게 DNS 헤더의 필드에 필요한 ID, Flag들, 그리고 해당 데이터에 포함된 내용의 개수 들에 대한 부분과 질의 형식일 때 사용되는 question section의 QNAME(질의문), QTYPE(질의 타입), QCLASS(질의 방식)이 포함된다.[1][2][3][4][19][20]

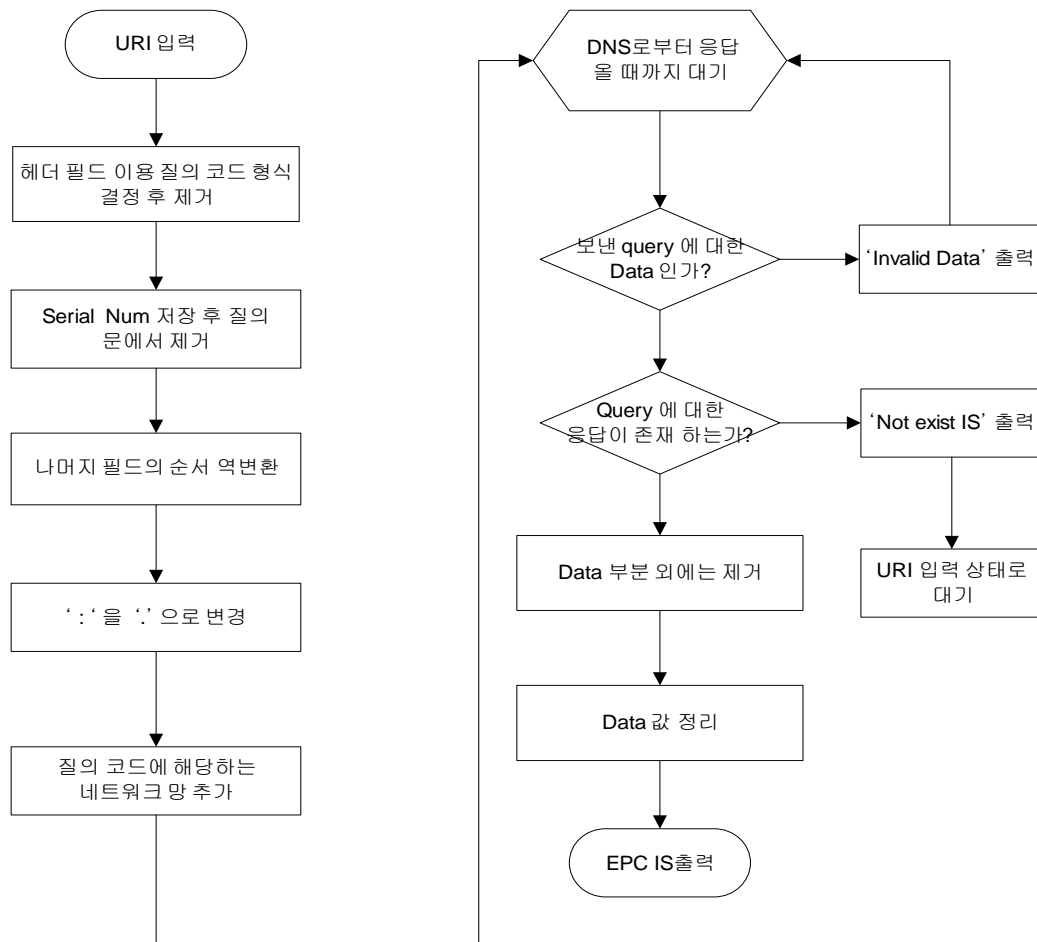


그림 5-8 URI형식의 RFID Code 변환

이렇게 전송된 DNS query에 대한 응답이 들어올 경우 그림 5-9과 같이 이를 event처리 하여 해당 데이터 값을 저장하여 그 내용에 대한 확인을 하고 내용이 DNS Response가 올바르다면, 데이터 내용 중에서 EPC-IS 에 관련된 응답 부분만을 정리하여 출력한다. 이 과정에서 각 EPC-IS들의 우선 순위(order 와 Preference를 통하여)를 확인 하는 것이 가능하다.

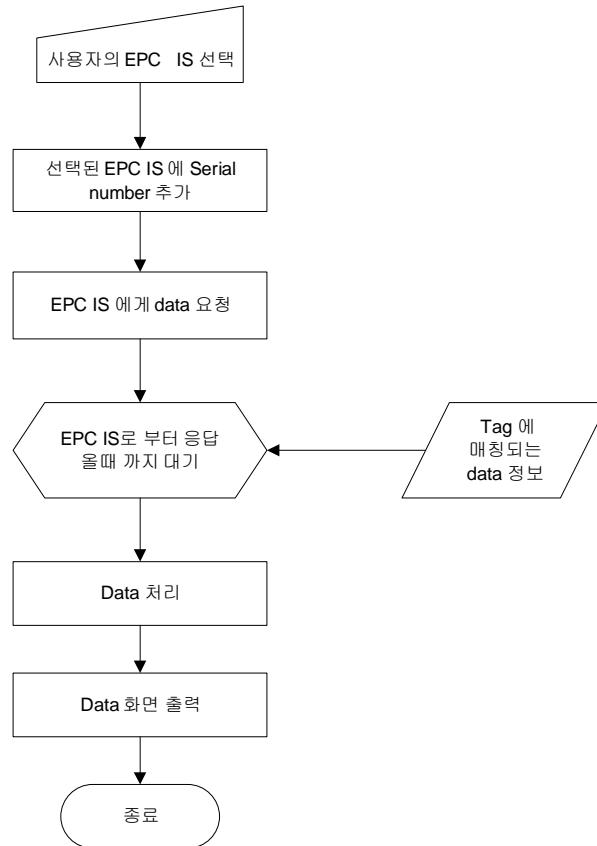


그림 5-9 EPC-IS 처리 과정

출력된 EPC-IS 중 하나를 사용자가 선택하게 되면 최초의 Tag로부터 얻은 Serial Number를 추가하여 WSP Client와 HTTP 웹브라우저로 RFID Code에 해당하는 정보를 요청한다.

본 논문에서는 Client역할을 하는 WAP Client와 EPC 네트워크의 유저 프로그램으로 웹브라우저를 사용하여 결과를 보여주는 두 가지 프로그램에 DNS Resolving모듈이 추가된다. DNS Resolving모듈은 C/MFC로 구현하였다.

## 가. Mobile Client(RFMS)의 Resolver 모듈

C로 구현된 ONS Resolver는 그림 5-8와 같이 URI형식의 RFID Code 값을 DN(Domain Name) 형식으로 변경하는 과정을 거쳐 DNS query메시지를 작성한 후 query메시지를 ONS에게 전송하게 된다. RFID Code를 Domain Name 형식으로 변환 하는 함수는 그림 5-10, 그림 5-11과 같다.

```
void URItToDN(char * rURI)
{
    char* num;
    char ch;
    char buffer[512]="";
    char tmp[512]="";
    char * sub_string = "urn:epc: ";
    char * sub_string_rfms = "urn:rfm: ";
    int type = -1 ;
    size_t total_length;
    size_t sub_length;
    int strLen;
    char * conv_string;
    char conv_string1[512]="";
    int i=0,j=0;
    /* check epc code */
    total_length = strlen(rURI);
    sub_length = strlen(sub_string);
    strncpy(tmp,rURI,sub_length);

    if( strcmp(sub_string,tmp)==0)
    {
        type =0;
    }
    else if( strcmp(sub_string_rfms,tmp)==0)
    {
        type =1;
    }
    else
    {
        printf("this is Not RFID URI.\n");
        exit(0);
    }

    conv_string=(char *)malloc( (total_length)-(sub_length) );
    strcpy(conv_string,&rURI[sub_length]);

    total_length=strlen(conv_string);
    num = strchr(conv_string,'. ');
    strcpy(serialNum,&num[1]);
    strLen=total_length-strlen(serialNum)-1;
    strncpy(conv_string1,conv_string,strLen);
    strcat(conv_string1,".xml");
}
```

그림 5-10 ONS Resolver URItToDN()

그림 5-10의 URItToDN() 함수를 통해 DN 형식으로 변경된 RFID Code는 그림 5-11의



mkQname()함수를 이용하여 DNS query의 question section중 QNAME부분을 만들게 된다.

```
void mkQname(char * dns)
{
    int dnslen;
    char tokenLen;
    char* tmpbuf;
    char* token;
    char* ptr;

    ptr = (char*)dnsquery;
    dnslen = strlen(dns);
    tmpbuf = (char*)malloc(sizeof(char) * dnslen);
    strcpy(tmpbuf, dns);
    token = strtok(tmpbuf, ".");
    tokenLen = strlen(token);

    /* Make DNS Query type */
    while(token != NULL)
    {
        debug(" token :%s len : %u", token, tokenLen);
        memcpy(ptr, &tokenLen, 1);
        strcat(dnsquery, token);
        ptr = (char*)(dnsquery + strlen(dnsquery));
        token = strtok(NULL, ".");
        if(token != NULL)
            tokenLen = strlen(token);
    }
    tokenLen = 0;
    memcpy(ptr, &tokenLen, 1);
    dnsQlen = strlen(dnsquery)+1;
} ? end mkQname ?
```

그림 5-11 ONS Resolver mkQname()

그림 5-12의 mkQueryPkt() 함수는 그림 5-2에서 설명한 DNS query의 header section과 question section의 QNAME을 제외한 나머지 부분을 만들어 완전한 DNS query메시지를 만들어 ONS에게 전송하게 된다.

```

void mkQueryPkt(void)
{
    char* query;
    unsigned short* tmptr;
    int hdrlen;
    dnsQhdr* qhdr;

    hdrlen = sizeof(dnsQhdr);
    debug("hdrlen : %d", hdrlen);
    qhdr = (dnsQhdr*)dnspacket;

    /* set header option 1 */
    qhdr->id = htons(ID); // 5383 , 2bytes 0x0716
    qhdr->flag = htons(0x0100); // 1 , 2bytes
    qhdr->qdcount = htons(0x0001); // 256, 2bytes
    qhdr->ancount = 0;
    qhdr->nscount = 0;
    qhdr->arcount = 0;

    query = (char*)(dnspacket+hdrlen);
    memcpy(query, dnsquery, dnsQlen);
    tmptr = (unsigned short*)(dnspacket + hdrlen + dnsQlen);
    *tmptr = htons(0x0023); /* Query Type 8960, 2bytes, 0x0023 */
    tmptr = tmptr + 1; // shift 2bytes
    *tmptr = htons(0x0001); /* Query Class 256, 2bytes, 0x0001 */

} ? end mkQueryPkt ?

```

그림 5-12 ONS Resolver mkQueryPkt()

## 나. EPC Client의 Resolver 모듈

EPC Client는 MFC를 이용하여 Dialog형식의 UI로 표현되며 Window 환경에서 동작이 가능하다. 마우스를 이용한 진행이 가능하며 질의에 대한 응답의 처리를 위해 이벤트 방식을 이용하여 각각의 응답을 처리 하였다. EPC client의 기본적인 동작 과정은 ONS resolver와 동일하다. 입력된 URI에대한 변환이 필요하며 이 과정을 통해 Domain Name형식으로 저장되어 DNS query에 알맞은 type으로 한번 더 변환되어 전송된다.

그림 5-13은 epc client에서 입력받은 URI(“urn:epc:id:sgtin.1.24.400”)에서 헤더(“epc:urn:”)를 제거하고 serial번호(400)를 저장 후, 삭제하는 일련의 과정을 보여준다. 입력 받은 URI의 헤더를 검사하여 client에서 제공하는 방식의 태그 정보가 아니라면 client는 종료된다.

```

// EPC code 확인
int total_length = URI.GetLength();
int sub_length = sub_string.GetLength();
tmp=URI.Mid(0,sub_length);

CString sub_string = "urn:epc:"; // EPC code 읽을 확인 하후기 뒤라 비교 값
CString sub_string_rfms = "urn:rftm:"; // RFMS code 확인 비교값

if( (tmp.Compare(sub_string)==0 ) )
{
    type=0;
}
else if((tmp.Compare(sub_string_rfms)==0 ) )
{
    type=1;
}
else
{
    AfxMessageBox("It is not URI");
    // 종료 방법 입력 필요
}
// remove header ( "urn:epc:" and "urn:rftm"
conv_string= new char[ (total_length) - (sub_length) ];
conv_string= URI.Right( (total_length) - (sub_length) );

//remove serial Number Field.
number=conv_string.ReverseFind('.');
SerialNum=conv_string.Mid(number+1,total_length);
conv_string=conv_string.Mid(0,number);
SerialNum=SerialNum+".xml";

```

그림 5-13 epc client 에서 URI converting 과정 (1)

```

//Invert the order & conversion ':' to '.'
number=conv_string.GetLength();
for( int i = number-1 ; i>1 ; i-- )
{
    ch=conv_string[i];
    if( ch == ':' )
    {
        buffer += conv_string.Right(conv_string.GetLength() - ( i + 1 ) );
        buffer += '.';
        conv_string=conv_string.Mid(0,i);
    }
    else if ( ch == '.' )
    {
        buffer += conv_string.Right(conv_string.GetLength() - ( i + 1 ) );
        buffer += ':';
        conv_string=conv_string.Mid(0,i);
    }
}
buffer +=conv_string;
// add network field
if ( type==0)
{
    buffer += ".hufs.ac.kr";
}
else if(type==1)
{
    buffer += ".icc.ac.kr";
}

```

그림 5-14 epc client 에서 URI converting 과정 (2)

그림 5-14는 남아있는 URI에서 ‘:’를 ‘.’으로 변경하고 각 필드의 위치를 역순으로 배치한 후, 태그에 따라 알맞은 네트워크망을 선택하여 추가 시켜 domain name으로 만들어주는 과정을 보여준다. 네트워크망의 선택은 헤더 부분을 검사하여 얻은 정보를 이용한다. 이후 전체 길이의 값은 저장되어 응답시에 사용 된다.

```
// DNS packet 입력
memset(m_tempQuery, 0x00, total_len);

memcpy(m_tempQuery + 0x00, &m_QueryHDID, 2);           // 2 "0715"
memcpy(m_tempQuery + 0x02, &m_QueryHDFlag, 2);         // 2 "0100"
memcpy(m_tempQuery + 0x04, &m_QueryHDQdcount, 2);       // 2 "0001"

memcpy(m_tempQuery + 0x0c, dnsQuery, len);               // len " dns type domain" //

memcpy(m_tempQuery + 0x0c + len, &m_QueryQtype, 2);      // 2 " 23( naptr type) "
memcpy(m_tempQuery + 0x0c + len + 0x02, &m_QueryQclass, 2); // 2 "0001"

// dns = m_QueryMsg;
m_QueryLength=len;
```

그림 5-15 DNS query packet 입력

그림 5-15 은 DNS type으로 만들어진 query를 이용하여 DNS query packet을 만들어 전송할 때 DNS header section과 question sectiond을 세팅하는 과정을 보여준다. Header의 설정은 그림 5-2 를 question section의 설정은 그림 5-3 참조하여 작성하였다. 각 field의 값은 사전 정의하여 해당 값을 입력하였다.[1][8][20]

## 제2절 Mobile Client 구현

WSP Client는 그림 5-16과 같이 입력된 RFID Code를 가지고 ONS Resolving후 WSP Request를 만드는 부분과 WSP Reply를 받은 후 처리과정으로 나뉜다. WSP Request는 Well-Known HTTP Header Name과 Well-Known HTTP Header Value로 구성되며, 그 구성은 HTTP Code Table을 참조한다. HTTP Code Table은 Well-Know Header에 대해 Binary Code값을 가지고 있다.

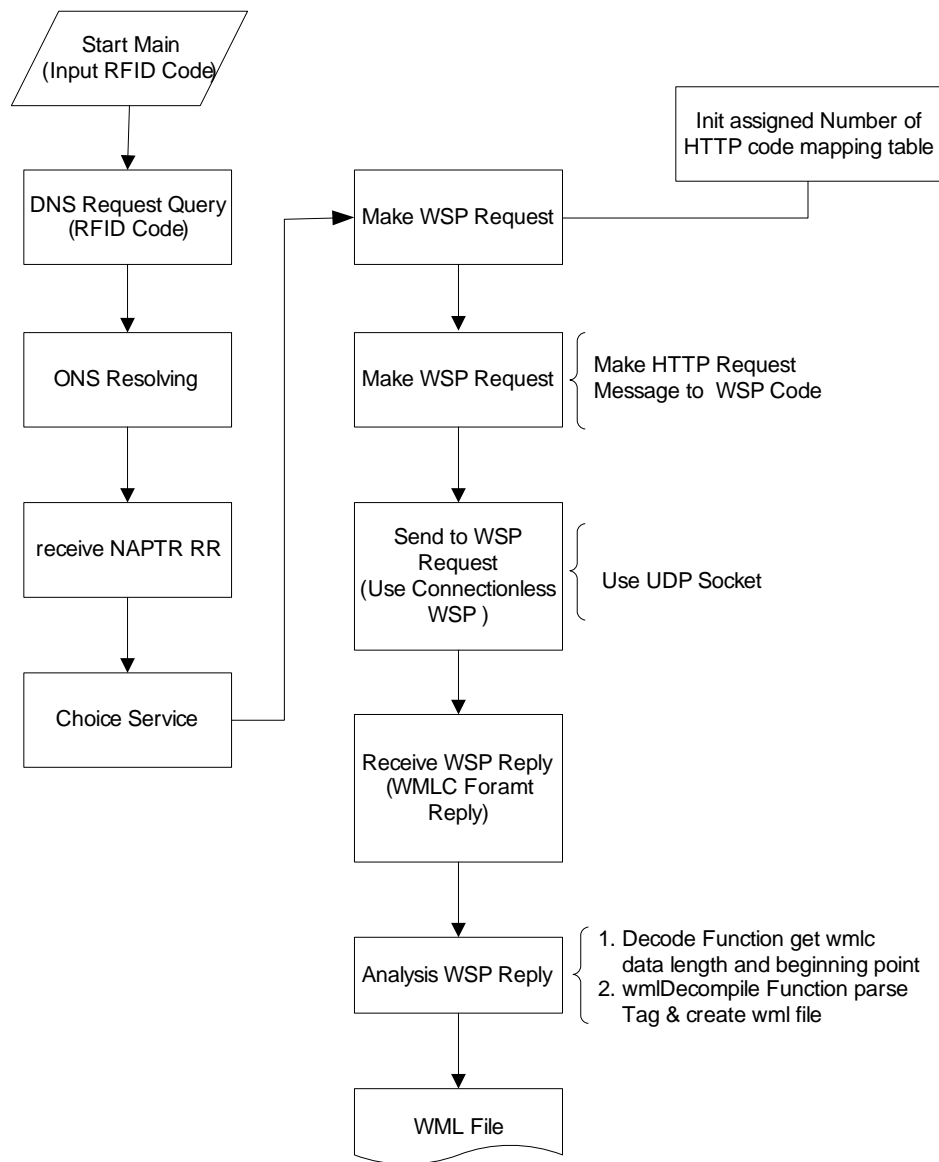


그림 5-16 WSP Client Flow

WSP Client가 WSP Reply를 받으면 WMLC로 구성된 데이터를 WML 포맷으로 변환하여 웹 브라우저에 보여준다.

그림 5-17은 WSP Reply가 도착하면 header 부분과 Data부분을 구분하여 WMLC 포맷을 WML 파일로 저장하기 위해 wmlc\_ptr에 Data부분의 인덱스를 저장하는 Decode() 부분이다.

```

int decode(const char *wsp_reply, int wsp_reply_len, unsigned short *tid, int *wmlc_ptr)
{
    int i=0, p=0;
    int status;
    unsigned int headerLen, contentLen=0, sizeLen, wmlcSize;
    unsigned char val, contentLenStr[8];
    char server[64];

    /* Check wsp reply and get wmlc data */

    /* Get tid of this package */
    *tid = wsp_reply[p++];

    /* Check method */
    if(strcasecmp("REPLY", Method[wsp_reply[p++]])!=0) {
        printf("Wrong method: %0x\n", wsp_reply[p-1]);
        return WRONG;
    }

    /* Check status */
    if((status = wsp_reply[p++]) != 0x20) {
        /* printf("status = %0x\n", status); */
        if (status == 0x44)
            printf("File not found!\n");
        else
            return WRONG;
    }
}

```

그림 5-17 WSP Client Decode()

### 제3절 Personal IS 구현

WSP Server는 EPC-IS의 기능을 하는 것으로써 WML데이터를 가지고 있으며, WSP Request를 전송 받으면, 해당 path에서 RIFD Code에 매핑된 정보를 WSP Reply를 통해 보내준다. 그림 5-18과 같이 WSP Server는 WML 데이터를 Mobile 환경을 고려하여 WMLC Binary Code로 만들어 Client에게 전송한다. 그림 5-18은 WSP Server의 순서도를 나타낸다. 그림 5-19는 WSP Server가 WSP Request를 받으면 헤더를 만들고, 데이터를 WMLC로 만들기 위해 wmlCompiler()를 호출하는 과정이다.

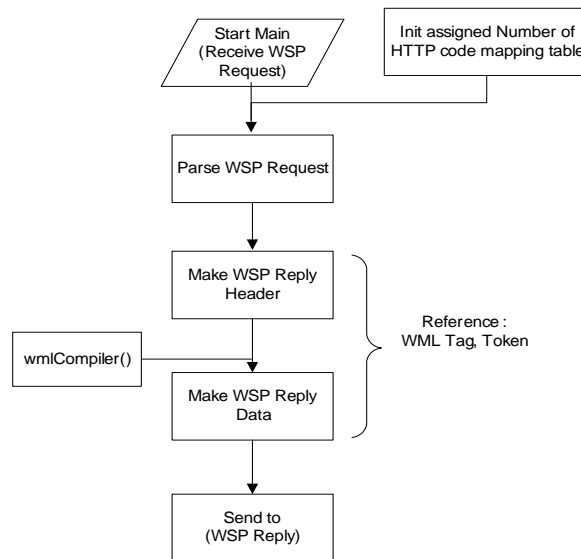


그림 5-18 WSP Server Flow

```

wsp_reply[cur++] = 0x80+search(header_field, HeaderField_en, HEADERFIELD_ENMAX);
strcpy(header_value, "application/vnd.wap.wmlc"); //strcpy(header_value, "text/vnd.wap.wml");
|
wsp_reply[cur++] = 0x80+search(header_value, FileType_en, FILETYPE_ENMAX);

strcpy(header_field, "Content-Language");
wsp_reply[cur++] = 0x80+search(header_field, HeaderField_en, HEADERFIELD_ENMAX);
strcpy(header_value, "en");
wsp_reply[cur++] = 0x80+search(header_value, Language_en, LANGUAGE_ENMAX);

printf("\n wsp_reply making... header \n");
/* calculate header length */
wsp_reply[p_headerlen] = cur - p_headerlen - 1;

printf("Thread%d: Encode done...\n\n", pthread_self());

/*file output*/
dataLen = read_file(filebuf, file);
p = filebuf;

/* transfer data */
wsp_reply[cur++] = 8;

if((wmlcSize = wmlCompiler(p, dataLen, wsp_reply+cur+8))<0) //p is data start pointer
    return WRONGTYPE;
sprintf(wsp_reply+cur, "%d", wmlcSize);
printf("Thread%d: WML compiler done...wmlcszie = %d\n\n", pthread_self(), wmlcSize);

```

그림 5-19 Make WSP Reply

## 제4절 EPC-Proxy 설계 및 구현

### 1. EPC-Proxy 개요

EPC Proxy는 기능적으로 인터넷에서 제공하는 서비스를 Mobile 단말기에 서비스하기 위해 만들어진 WAP-Gateway의 확장된 형태이다. WAP-Gateway는 인터넷에서 제공하는 Content를 Mobile 단말기에 제공하기 위해 WAP Protocol과 TCP/IP Protocol을 변환하는 Protocol Conversion기능과 무선대역폭을 최대한 활용하기 위해서 WML 데이터를 WMLC로 압축하는 기능을 가진다. 데이터의 요청은 항상 Mobile 단말기에서 이루어지며, 각종 다양한 Content Service는 웹서버 혹은 wap서버에서 제공된다. 현재의 WAP-Gateway의 기능으로는 EPC 네트워크와 Mobile RFID 네트워크 사이에서 인터넷을 사용하여 Mobile RFID 네트워크에서 제공하는 서비스를 이용할 수 없다. EPC-Proxy는 이것을 가능하게 하며 부가적으로 다음과 같은 장점을 가진다.

- EPC 네트워크의 구조와 관계없이 기존의 Mobile 네트워크의 구조변경 없이 Mobile RFID 네트워크를 구축하여 서비스를 제공할 수 있다.
- 두 네트워크의 Access Point를 제공함으로써, 각종 소프트웨어 기능 모듈의 추가 및 삭제가 용이하다.
- 인터넷을 통해 Mobile RFID 네트워크 서비스를 제공할 수 있다.

### 2. EPC-Proxy Flow Diagram

EPC-Proxy는 아래와 같이 두 가지 정보흐름을 지원한다.

- EPC 유저가 Mobile RFID 네트워크 제공하는 서비스를 요청하는 경우
- RFMS를 통해 EPC 네트워크가 제공하는 서비스를 요청하는 경우

위와 같은 두 가지 흐름을 위해서 프로그램은 기능적으로 Client/Server의 모델을 가져야 하며, WSP 와 HTTP 프로토콜의 Conversion이 가능해야 한다.



## 가. EPC 유저가 Mobile RFID 네트워크로 정보요청시 흐름도

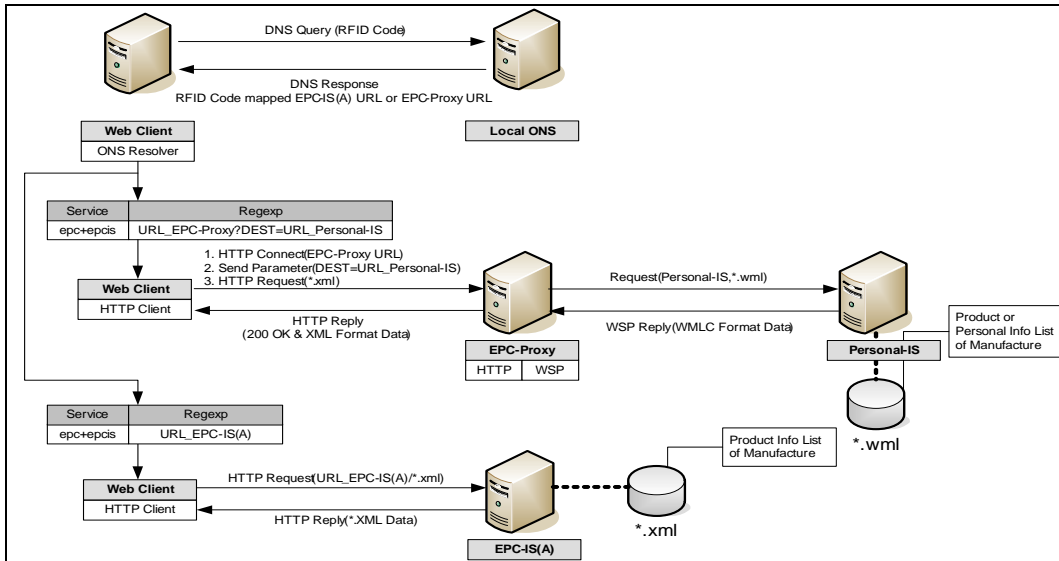


그림 5-20 Mobile RFID 네트워크의 정보이용

## 나. RFMS를 통해 EPC 네트워크로 정보요청시 흐름도

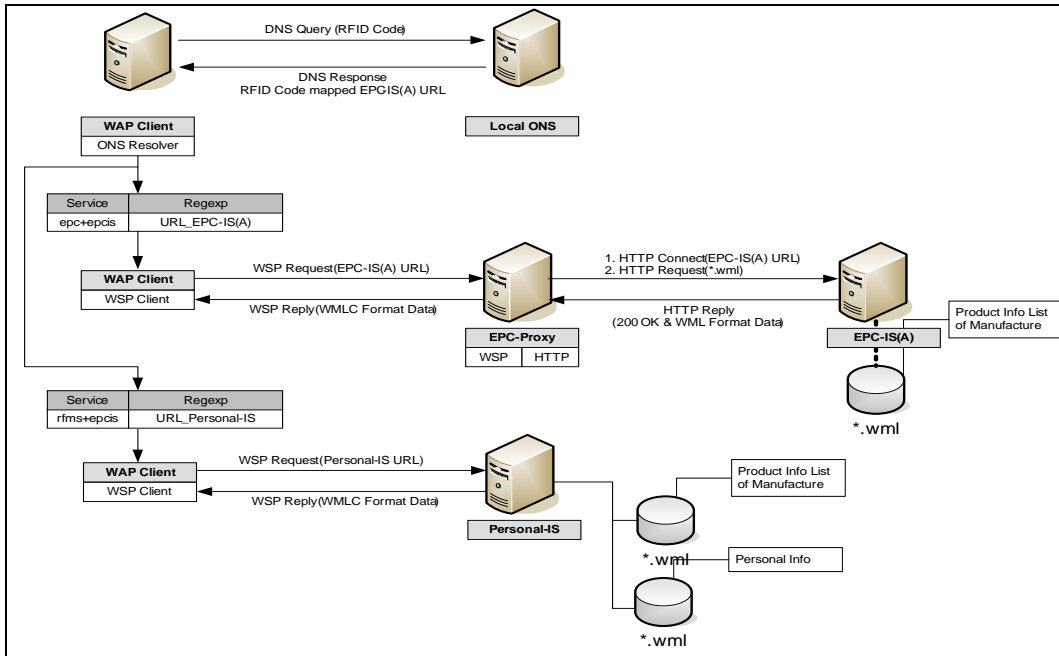


그림 5-21 EPC 네트워크의 정보이용

#### 다. Program Sequence Chart

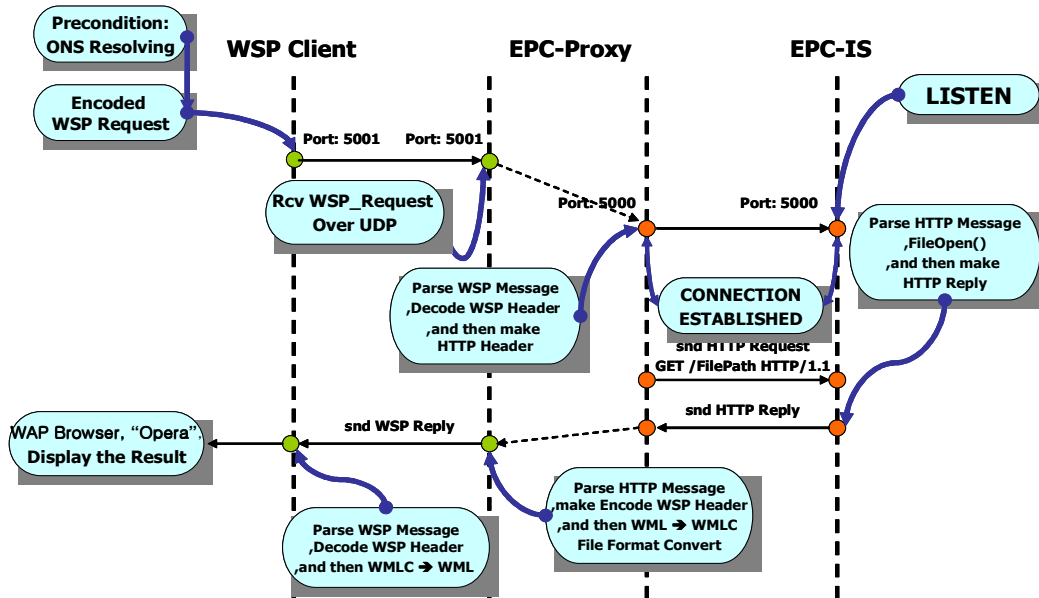


그림 5-22 Program Sequence Chart

그림 5-22 는 WSP Client에서 ONS Resolving을 거쳐 EPC-IS의 데이터를 서비스 받기 위한 일련의 행동과정의 순서를 나타낸 것이다.

### 3. EPC-Proxy 구현

EPC-Proxy는 WSP Message를 HTTP Message로 Protocol Conversion 후에 EPC 네트워크에 데이터를 요청하는 쓰레드와 HTTP Message를 WSP Message로 Protocol Conversion하는 쓰레드로 구현된다. EPC-Proxy의 주 기능은 Protocol Conversion 뿐 아니라 ONS 리졸빙을 통해 얻은 상대방의 URL을 파싱하는 역할을 제공하며, 상대 네트워크의 Access Point로써 동작하는 것이다.

## 가. Mobile RFID 네트워크에서 EPC 네트워크로 정보요청

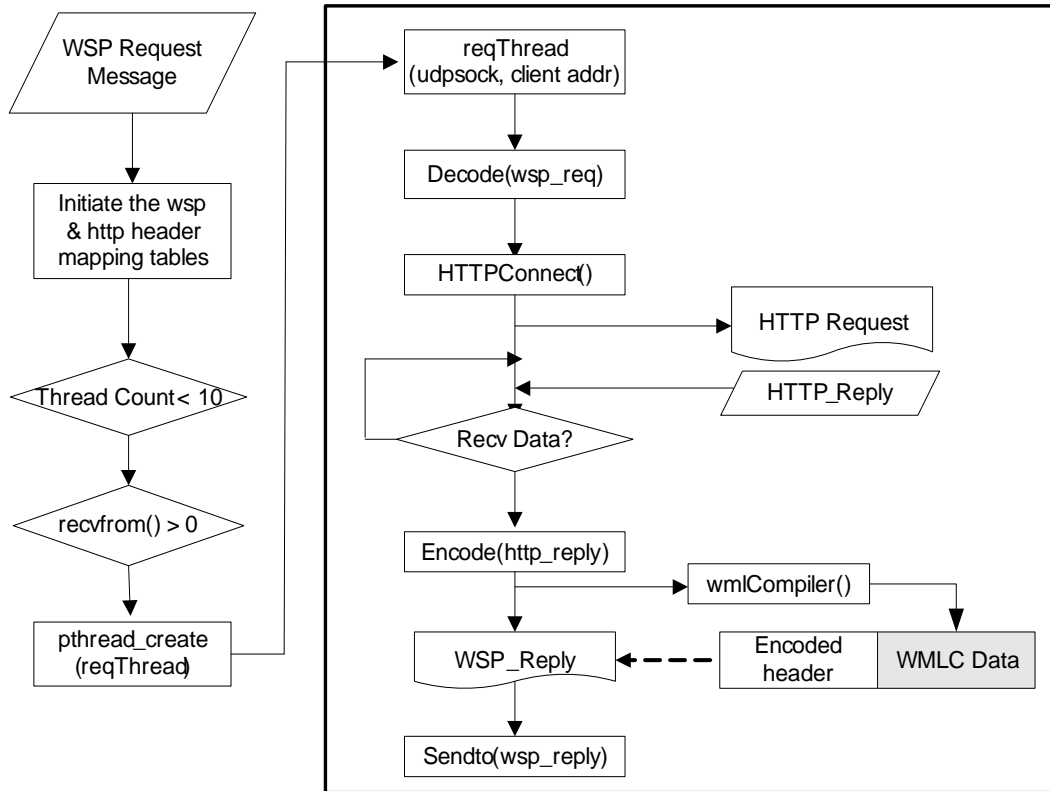


그림 5-23 EPC 네트워크로 정보요청

그림 5-23는 EPC-Proxy가 RFMS로부터 EPC 네트워크의 EPC-IS로 요청되는 정보를 처리되는 과정이다. WSP Request는 Binary 포맷으로 만들어져 EPC-Proxy로 전달되며, 전달된 WSP Request를 HTTP Request로 변환한다.

EPC-IS는 HTTP Request의 RFID 코드를 보고 HTTP Reply를 통해 EPC-Proxy에게 해당 데이터를 리턴한다. EPC-Proxy는 HTTP Reply를 통해 HTTP Header부분을 인코딩 과정을 통해 WSP Header를 만드며, HTTP Reply의 Data는(\*.WML) WMLC 파일 포맷으로 만들어 “WSP Header + WMLC”의 형태로 WSP Client에게 전송한다.

표 5-1는 WSP Client에서 EPC-IS로 RFID Serial Number에 맵핑된 정보요청시 EPC-Proxy에서 처리하는 주요 Method List와 그 기능을 나열한 것이다.

표 5-1 Main Method List

Method	Operation	Return Value
Decode	WSP Request에서 요청할 서버이름, 서버포트를 추출한다. WSP Request를 HTTP Code String table을 참조하여 HTTP Request를 작성한다.	Error 발생시 Error Code를 리턴한다.
HTTPConnect	서버이름과 서버포트를 이용하여 HTTP 서버와 Connect한다. GET Method를 사용하여 RFID Serial Number에 해당하는 정보를 요청한다. HTTP 서버로부터 HTTP Reply를 기다린다.	Error 발생시 Error Code를 리턴한다.
Encode	HTTP Reply를 받은 후 HTTP Reply Header를 WSP Code String Table을 참조하여 WSP Header를 만든다. HTTP Reply의 Data 부분의 시작 포인터를 계산하여 wmlCompiler()에게 알려준다.	Error 발생시 Error Code를 리턴한다.
wmlCompiler	xmlParserFile()을 통해 xml file Parsing후에 correct xml file임을 판단한 후, 각 xml tag를 encode tag로 교체한다.	Error 발생시 Error Code를 리턴한다.

그림 5-24은 HTTP Code Table로써 HTTP와 WSP간 메시지 Encoding과 Decoding시에 사용된다.

```

void table_init()
{
    /* initialize Method table */
    strcpy(Method[0x01], "CONNECT"); strcpy(Method[0x04], "REPLY");
    strcpy(Method[0x05], "DISCONNECT"); strcpy(Method[0x06], "PUSH");
    strcpy(Method[0x08], "SUSPEND"); strcpy(Method[0x09], "RESUME");
    strcpy(Method[0x40], "GET"); strcpy(Method[0x60], "POST");
    strcpy(Method[0x61], "PUT");

    /* initialize HeaderField table */
    strcpy(HeaderField[0x00], "Accept"); strcpy(HeaderField[0x01], "Accept-Charset");
    strcpy(HeaderField[0x02], "Accept-Encoding"); strcpy(HeaderField[0x03], "Accept-Language");
    strcpy(HeaderField[0x04], "Accept-Range"); strcpy(HeaderField[0x08], "Content-Encoding");
    strcpy(HeaderField[0x0C], "Content-Language"); strcpy(HeaderField[0x0D], "Content-Length");
    strcpy(HeaderField[0x10], "Content-Range"); strcpy(HeaderField[0x11], "Content-Type");
    strcpy(HeaderField[0x12], "Date"); strcpy(HeaderField[0x1F], "Proxy-Authenticate");
    strcpy(HeaderField[0x20], "Proxy-Authorization"); strcpy(HeaderField[0x26], "Server");
    strcpy(HeaderField[0x29], "User-Agent");

    /* initialize FileType table */
    strcpy(FileType[0x01], "text/*"); strcpy(FileType[0x02], "text/html");
    strcpy(FileType[0x03], "text/plain"); strcpy(FileType[0x08], "text/vnd.wap.wml");
    strcpy(FileType[0x09], "text/vnd.wap.wmlscript"); strcpy(FileType[0x12], "application/x-www-form-urlencoded");
    strcpy(FileType[0x14], "application/vnd.wap.wmlc"); strcpy(FileType[0x15], "application/vnd.wap.wmlscript");
    strcpy(FileType[0x16], "application/vnd.wap.wta-eventc"); strcpy(FileType[0x17], "application/vnd.wap.uaprof");
    strcpy(FileType[0x18], "application/vnd.wap.wtls-ca-certificate"); strcpy(FileType[0x19], "application/vnd.wap.wtls-user-certificate");
    strcpy(FileType[0x1D], "image/gif"); strcpy(FileType[0x1E], "image/jpeg");
    strcpy(FileType[0x21], "image/vnd.wap.wbmp"); strcpy(FileType[0x27], "application/xml");
    strcpy(FileType[0x28], "text/xml"); strcpy(FileType[0x29], "application/vnd.wap.wbxm");

    /* initialize Language table */
    strcpy(Language[0x19], "en"); strcpy(Language[0x22], "fr");
    strcpy(Language[0x36], "ja"); strcpy(Language[0x8B], "la");
    strcpy(Language[0x41], "zh");

    /* initialize character set table */
    strcpy(Charset[0x04], "iso-8859-1");
}
? end table_init ?

```

그림 5-24 WSP와 HTTP Header의 매핑 테이블

WSP Client에서 EPC-IS의 정보를 이용하기 위해 WSP Request를 보내면 EPC-Proxy는 그림

5-25와 같이 WSP Request의 Encoding된 Request를 HTTP Code Table을 참조하여 HTTP Request로 변환한다.

```

/* Get headers */
while(wsp_request[p]) {
    val = wsp_request[p++];

    /* File type header */
    if(!strcmp(HeaderField[val-0x80], "Accept")) {
        strcat(http_request, "Accept: ");

        val = wsp_request[p++];
        strcat(http_request, FileType[val-0x80]);

        while(wsp_request[p]==wsp_request[p-2]) {
            val = wsp_request[p+1];
            strcat(http_request, ", ");
            strcat(http_request, FileType[val-0x80]);
            p+=2;
        }
        strcat(http_request, "\n");
    }

    /* Character set header */
    if(!strcmp(HeaderField[val-0x80], "Accept-Charset")) {
        strcat(http_request, "Accept-Charset: ");
        /* p = addHeaderValue(wsp_request, p, Charset, http_request);*/

        val = wsp_request[p++];
        strcat(http_request, Charset[val-0x80]);

        while(wsp_request[p]==wsp_request[p-2]) {
            val = wsp_request[p+1];
            strcat(http_request, ", ");
            strcat(http_request, Charset[val-0x80]);
            p+=2;
        }
        strcat(http_request, "\n");
    }
}

```

그림 5-25 WSP Request를 HTTP Request로 변환

그림 5-26는 Encode()의 일부분으로써, Method\_en(HTTP Code Mapping Table)을 검색하여 “REPLY” Method에 해당하는 코드를 가지고 WSP Reply Header를 만드는 과정을 보여준다.

```

/* Add tid to WSP reply */
wsp_reply[cur++] = (unsigned char)tid;

/* Add encoding of method to WSP reply */
wsp_reply[cur++] = search("REPLY", Method_en, METHOD_ENMAX);

p = index(http_reply, '\n');

/* Get first line of http reply */
strncpy(line, http_reply, p-http_reply);
/* Ignore protocol name */
protocol = strtok(line, "\t");
if(strcmp(protocol, "HTTP/1.1"))
    return WRONGREPLY;

```

그림 5-26 WSP Reply Message

그림 5-27은 Enocde/Decode()에서 사용되며, 공통적으로 원하는 매핑 테이블의 포인터와 원하는 검색데이터를 넣어 binary Search를 사용하여 매핑코드를 찾아낸다.

```

/* Use binary search method to get the WSPcode of a HTTPcode */
unsigned char search(char *s, Map *table, int tablesize)
{
    int begin, end, middle, r;

    begin=0;
    end = tablesize-1;

    while(begin <= end) {
        middle = (end+begin)/2;
        r = strcasestr(s, table[middle].HTTPcode);
        if(r < 0)
            end = middle-1;
        else if(r > 0)
            begin = middle+1;
        else
            return table[middle].WSPcode;
    }
    printf("Search func can't find %s\n", s);
    return 0;
} ? end search ?

```

그림 5-27 HTTP, WSP Mapping Code Serach Method

그림 5-28은 wmlCompiler()의 기능으로써, WML Tag를 WML Tag Encoding Table에서 검색하여 각 Tag를 이진코드로 변환하는 기능을 한다.

```

void compileTags(xmlNodePtr node, char *out, int *p)
{
    int tagVal, tagNum = -1;

    while(node) {
        if(strcmp("text", node->name) == 0) {
            outputString(node->content, out, p);
        } else {
            if((tagNum = getTagIndex(node->name)) < 0) {
                fprintf(stderr, "Parse error: unknown tag %s\n", node->name);
                crashBurn();
            }

            tagVal = tagNum;

            if(node->properties)
                tagVal |= WMLTC_ATTRIBUTES;

            if(node->chilids)
                tagVal |= WMLTC_CONTENT;

            out[(*p)++] = tagVal;

            if(node->properties)
                compileAttributes(node->properties, out, p);

            if(node->chilids)
                compileTags(node->chilids, out, p);

            if(tagHasClosure(tagNum))
                out[(*p)++] = WMLTC_END;
        } ? end else ?

        node = node->next;
    } ? end while node ?
} ? end compileTags ?

```

그림 5-28 WML Tag Compiler

WSP Reply의 WML 데이터를 이진코드로 변환하여 전송함으로써, 부족한 무선대역폭을 효과적으로 사용할 수 있다.

WML Tag는 그림 5-29와 같이 각 태그마다 그림 5-30의 매핑 코드가 존재하며 wmlCompiler()는 이 매핑 테이블을 가지고 WSP Reply의 데이터영역을 Encoding 하여 WSP Client에게 전달한다.

```
#define WML_TAGS_MIN WMLT_A
#define WML_TAGS_MAX WMLT_WML
static char wml_tags[(WML_TAGS_MAX - WML_TAGS_MIN) + 1] = {
/* 0x1D */ "td",
/* 0x1E */ "tr",
/* 0x1F */ "table",
/* 0x20 */ "p",
/* 0x21 */ "postfield",
/* 0x22 */ "anchor",
/* 0x23 */ "access",
/* 0x24 */ "b",
/* 0x25 */ "big",
/* 0x26 */ "br",
/* 0x27 */ "card",
/* 0x28 */ "do",
/* 0x29 */ "em",
/* 0x2A */ "fieldset",
/* 0x2B */ "go",
/* 0x2C */ "head",
/* 0x2D */ "i",
/* 0x2E */ "img",
/* 0x2F */ "input",
/* 0x30 */ "meta",
/* 0x31 */ "noop",
/* 0x32 */ "prev",
/* 0x33 */ "onevent",
/* 0x34 */ "optgroup",
/* 0x35 */ "option",
/* 0x36 */ "refresh",
/* 0x37 */ "select",
/* 0x38 */ "small",
/* 0x39 */ "strong",
/* 0x3A */ NULL,
/* 0x3B */ "template",
/* 0x3C */ "timer",
/* 0x3D */ "u",
/* 0x3E */ "setvar",
/* 0x3F */ "wml",
};
#define WML_TAG_DESC(a) (((a>=WML_TAGS_MIN)&&(a<=WML_TAGS_MAX))?wml_tags[(a-WML_TAGS_MIN)]:NULL)
```

그림 5-29 WML Tag

```
/* Tag tokens */

#define WMLT_A 0x1C
#define WMLT_ANCHOR 0x22
#define WMLT_ACCESS 0x23
#define WMLT_B 0x24
#define WMLT_BIG 0x25
#define WMLT_BR 0x26
#define WMLT_CARD 0x27
#define WMLT_DO 0x28
#define WMLT_EM 0x29
#define WMLT_FIELDSET 0x2A
#define WMLT_GO 0x2B
#define WMLT_HEAD 0x2C
#define WMLT_I 0x2D
#define WMLT_IMG 0x2E
#define WMLT_INPUT 0x2F
#define WMLT_META 0x30
#define WMLT_NOOP 0x31
#define WMLT_P 0x20
#define WMLT_POSTFIELD 0x21
#define WMLT_PREV 0x32
#define WMLT_ONEVENT 0x33
#define WMLT_OPTGROUP 0x34
#define WMLT_OPTION 0x35
#define WMLT_REFRESH 0x36
#define WMLT_SELECT 0x37
#define WMLT_SETVAR 0x3E
#define WMLT_SMALL 0x38
#define WMLT_STRONG 0x39
#define WMLT_TABLE 0x1F
#define WMLT_TD 0x1D
#define WMLT_TEMPLATE 0x3B
#define WMLT_TIMER 0x3C
#define WMLT_TR 0x1E
#define WMLT_U 0x3D
#define WMLT_WML 0x3F
```

그림 5-30 WML Tag Token

본 논문에서는 WML 문서를 WMLC로 인코딩하기 위해 각 네스팅된 WML 엘리먼트들을 핸들링

할 수 있는 libxml2 라이브러리를 사용하였다. Libxml2 라이브러리는 표 5-2와 같이 XML 문서를 핸들링 할 수 있는 함수들을 정의하고 있으며, XML 문서가 로딩되면 그림 5-31의 xmlDoc structure의 root아래 그림 5-32와 같은 child structure의 트리구조로 가지고 있어 네스팅된 WML 또는 XML 태그를 쉽게 파싱할 수 있다.

```
typedef struct _xmlDoc xmlDoc;
typedef xmlDoc *xmlDocPtr;
struct _xmlDoc {
#ifdef XML_WITHOUT_CORBA
    void *_private; /* for Corba, must be first ! */
    void *vepv; /* for Corba, must be next ! */
#endif
    xmlElementType type; /* XML_DOCUMENT_NODE, must be second ! */
    char *name; /* name/filename/URI of the document */
    const xmlChar *version; /* the XML version string */
    const xmlChar *encoding; /* encoding, if any */
    int compression; /* level of zlib compression */
    int standalone; /* standalone document (no external refs) */
    struct _xmlDtd *intSubset; /* the document internal subset */
    struct _xmlDtd *extSubset; /* the document external subset */
    struct _xmlNs *oldNs; /* Global namespace, the old way */
    struct _xmlNode *root; /* the document tree */
    void *ids; /* Hash table for ID attributes if any */
    void *refs; /* Hash table for IDREFs attributes if any */
};
```

그림 5-31 xmlDoc Structure

```
struct _xmlNode {
#ifdef XML_WITHOUT_CORBA
    void *_private; /* for Corba, must be first ! */
    void *vepv; /* for Corba, must be next ! */
#endif
    xmlElementType type; /* type number in the DTD, must be third ! */
    struct _xmlDoc *doc; /* the containing document */
    struct _xmlNode *parent; /* child->parent link */
    struct _xmlNode *next; /* next sibling link */
    struct _xmlNode *prev; /* previous sibling link */
    struct _xmlNode *childs; /* parent->childs link */
    struct _xmlNode *last; /* last child link */
    struct _xmlAttr *properties; /* properties list */
    const xmlChar *name; /* the name of the node, or the entity */
    xmlNs *ns; /* pointer to the associated namespace */
    xmlNs *nsDef; /* namespace definitions on this node */
#ifdef XML_USE_BUFFER_CONTENT
    xmlChar *content; /* the content */
#else
    xmlBufferPtr content; /* the content in a buffer */
#endif
} ? end _xmlNode ? ;
```

그림 5-32 xmlNode Structure

xmlNodePtr는 표 5-2 와 같은 포인터를 사용하여 문서를 파싱하며, 그 예는 그림 5-33와 같다.



표 5-2 xmlNodePtr의 기능

구 분	기 능
xmlNodePtr→children xmlNodePtr→last xmlNodePtr→parent	Node의 첫번째 child Node의 마지막 child 현재 Node의 부모 노드
xmlNodePtr→next xmlNodePtr→prev	다음 같은 수준의 노드 이전 같은 수준의 노드
xmlNodePtr→doc	자신(Node)을 포함하고 있는 document의 xmlDocPtr 포인터

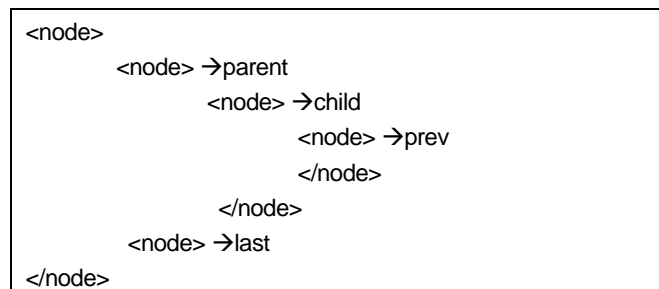


그림 5-33 네스팅된 문서구조

#### 나. EPC 네트워크에서 Mobile RFID 네트워크로 정보요청

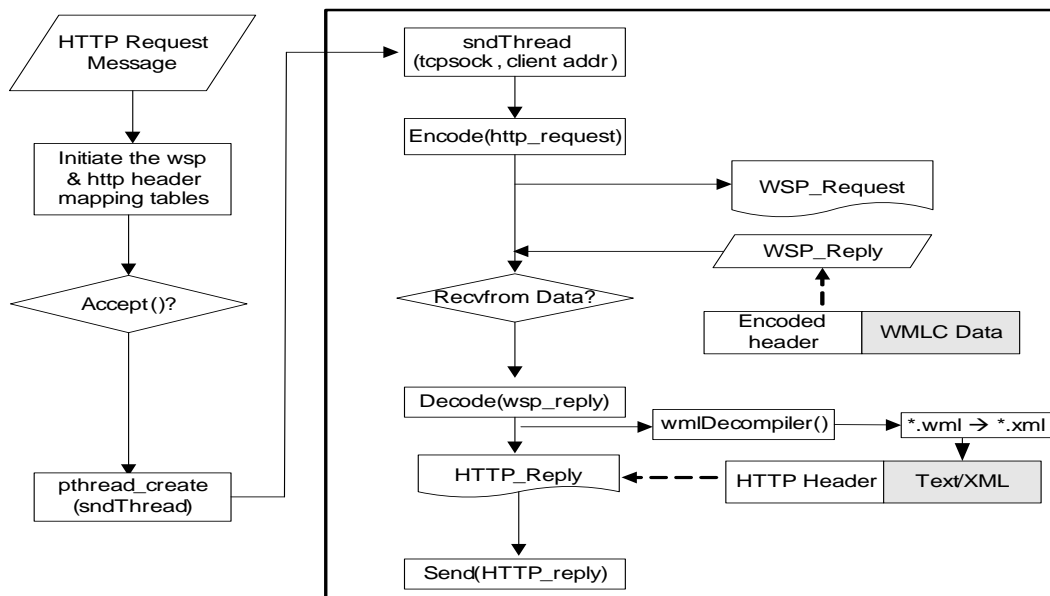


그림 5-34 Mobile RFID 네트워크로 정보요청

그림 5-34은 EPC 네트워크 유저가 Mobile RFID의 Personal-IS로 정보를 요청할 때 EPC-Proxy가 처리하는 과정이다. EPC 네트워크 유저는 ONS를 통해 “URL\_EPC-Proxy/URL\_Personal-IS”를 리턴 받으며, URL\_EPC-Proxy를 통해 HTTP Connect를 요청하며, EPC-Proxy는 URL\_Personal-IS를 통해 Personal-IS로 RFID 맵핑 데이터를 요청한다. EPC 유저는 이런 중간과정을 인식하지 못하며 서비스 받게 된다.

표 5-3은 EPC 유저가 Mobile RFID 네트워크의 Personal-IS로 RFID Code에 해당하는 정보를 전송하기 위한 처리 과정 중 중요함수들과 그 기능을 나타낸 것이다. RMFS가 EPC-IS에 데이터를 요청하는 것과 반대되는 과정으로써, 다른 점은 사용자가 EPC-Proxy의 경유를 모르게 하기 위해 ONS에는 “URL\_EPC-Proxy/URL\_Personal-IS”의 NAPTR RR을 관리하며, EPC-Proxy는 이 Path를 보고 Personal-IS의 URL을 알아내야 한다.

표 5-3 Main Method List

Method	Operation	Return Value
Main (Encode)	HTTP Request Parsing parserURL()을 통해 Hidding된 Person-IS URL을 알아낸다. HTTP Request를 WSP Request로 변환한다. WSP Request를 WSP Server(WAP Server)에게로 전송한다.	Error 발생시 Error Code를 리턴 한다.
Decode	WSP Reply를 대기한다. HTTP Code Mapping table을 참조하여 WSP Reply Header의 이진코드를 HTTP Reply format으로 변환한다. WSP Reply의 데이터영역의 WMLC Data를 WML파일로 변환 후 임시파일로 저장한다. 임시파일로 저장된 WML 파일을 XML 파일로 변환 후 HTTP Reply의 Data부분에 실어 HTTP Client에게 전송한다.	Error 발생시 Error Code를 리턴 한다.
wmlDecompiler	Decode()에서 호출되며, WMLC 파일을 WML Tag Mapping Table을 참조하여 WML 파일로 만든다.	Error 발생시 Error Code를 리턴 한다.

그림 5-35은 Encoding된 WSP Request를 보여준다. WSP Request는 HTTP Request와 같이 Header::=Header value형식을 지원한다.

```

/* Initialize WSP request */
wsp_req[1] = 0x40;

/* http:// */
wsp_req[3] = 0x68;
wsp_req[4] = 0x74;
wsp_req[5] = 0x74;
wsp_req[6] = 0x70;
wsp_req[7] = 0x3a;
wsp_req[8] = 0x2f;
wsp_req[9] = 0x2f;

/* URL */
/* buf = Get /DEST=PersonalIS URL/RFIDCODE_Page.xml HTTP/1.1 */
paresURL(buf,wspname,filename);
/* buf = pis.hufs.ac.kr/234.wml */
/* wspname = pis.hufs.ac.kr */
/* filename = /234.wml */
printf("\n pass paresURL() \n");

for(i=0; i<strlen(filename); i++)
    wsp_req[10+i] = filename[i]; //7# http://

/* URL Length */
wsp_req[2] = i+7;
//wsp_req[2] = i;

/* Accept character set: iso-8859-1 */
wsp_req[10+i++] = 0x81;
wsp_req[10+i++] = 0x84;
/* Accept type: application/vnd.wap.wmlc, text/html */
wsp_req[10+i++] = 0x80;
wsp_req[10+i++] = 0x94;
wsp_req[10+i++] = 0x80;
wsp_req[10+i++] = 0x82;

```

그림 5-35 WSP Request

그림 5-36는 EPC 유저가 Person-IS의 정보를 이용하기 위해 ONS Resolving을 통해 가져온 데이터를 해석하는 작업을 나타낸다. EPC-Proxy를 감추기 위해 “URL-EPC-Proxy/URL\_Personal-IS”의 형태로 전달받으므로, HTTP Request Message를 해석해야 한다.

```

void paresURL(char * buf, char * wspname, char * filename)
{
    /* buf = GET /DEST=PersonalIS URL/RFIDCODE_Page.xml HTTP/1.1 */
    int index = 0;
    int strlength = 0;
    char tmpbuf[32];
    char newbuf[32];
    int position;
    //char wspname[32];
    //char filename[20];
    char tmpname[32];

    for(index = 0 ; index < strlen(buf) ; index++)
    {
        if( buf[index] == '\n' )
        {
            printf("stop here\n");
            break;
        }
    }
    strncpy(tmpbuf,buf,index - 1);
    index = 0;
    strcpy(newbuf,tmpbuf+10);
    printf("\n1=%s\n",newbuf);/*Person- IS URL/234.xml HTTP/1.1" || 10 is "GET /DEST="*/
    strcat(newbuf,"\0");

    strlength = strlen(newbuf);
    strcpy(tmpbuf,"");

    for(index = 0 ; index < strlen(newbuf); index++)
    {
        if( newbuf[index] == ' ' )
        {
            newbuf[index] = '\0';
            printf("\n%d\n",index);
            break;
        }
    }

    strcpy(tmpbuf,newbuf);
    printf("\n2=%s\n",tmpbuf);/*Person- IS URL/234.xml"*/
}

```

그림 5-36 Person-IS URL 해석

## 제6장 Mobile RFID 네트워크와 EPC 네트워크간 서비스 연동 시험

### 제1절 서비스 연동환경

Mobile RFID 네트워크에서 개인정보서비스와 EPC 네트워크의 서비스연동을 테스트 하기 위해서는 먼저 ONS Delegation Chain을 구성하여 RFID Code에 대한 연계성이 필요하며, EPC Client와 WAP Client에서 Resolving 절차의 테스트를 거쳐야한다. 최종 테스트에서는 WAP Client가 EPC 네트워크의 EPC-IS로부터 RFID Code에 매핑되는 정보를 전송받고, 그 반대과정으로 EPC 유저가(EPC Client) Mobile RFID 네트워크의 Personal-IS로부터 RFID Code에 매핑되는 정보를 받는 것을 확인한다.

#### 1. ONS Delegation 구성

ONS는 그림 6-1과 같은 위임구조를 가지며, 논리적으로 IS가 위치한 네트워크에 따라 “hufs” 도메인은 EPC 네트워크를, “icc” 도메인은 Mobile RFID 네트워크를 나타낸다.

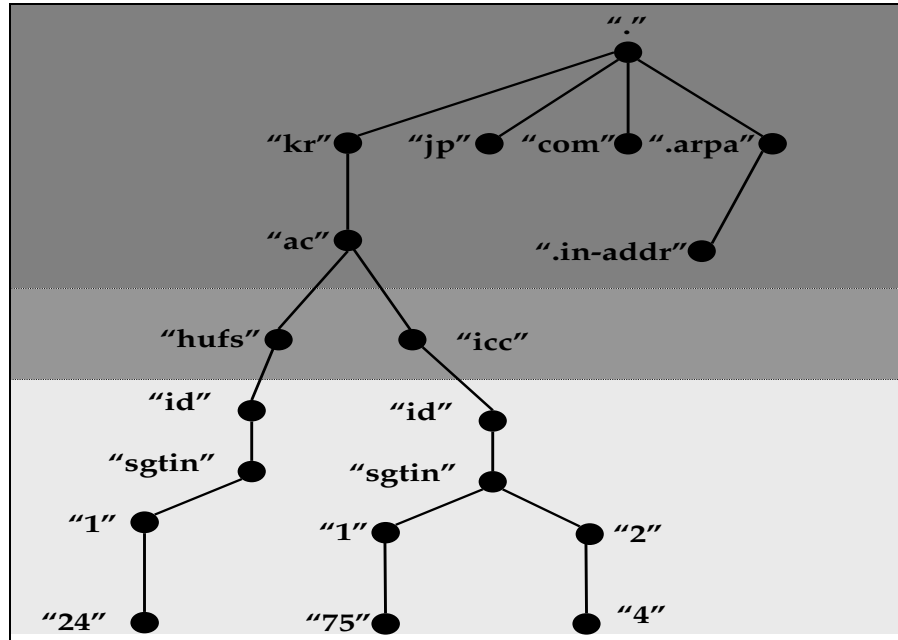


그림 6-1 Testbed ONS Delegation Chain

그림 6-2는 Root ONS기능을 한다. 서브로 두 DNS를 정의 하고 있으며 NS 레코드를 사용하여 hufs 도메인과 icc도메인의 권한을 위임한다.[8][12][20]

\$TTL 86400				
@	IN	SOA	ac.kr. mclab.ac.kr. (	
			2004092401	;serial
			21600	;Refresh
			1800	;Retry
			1209600	;expire
			86400	;mininum
			)	
	IN	NS	ac.kr.	
ac.kr.	IN	A	192.168.5.5	
hufs	IN	NS	hufs.ac.kr.	
icc	IN	NS	icc.ac.kr.	
hufs.ac.kr.	IN	A	192.168.5.3	
icc.ac.kr.	IN	A	192.168.7.2	

그림 6-2 Root DNS 설정

그림 6-3과 그림 6-4는 icc도메인과 hufs도메인의 Zone파일 구성을 보여준다. ONS는 기본적으로 DNS의 구성과 다르지 않음을 보여준다. 단지 ONS는 NAPTR RR로 관리되며, EPC 네트워크는 현재 NAPTR RR의 서비스 필드를 사용하여 서비스 종류를 넓혀가고 있다.

\$TTL 86400				
@	IN	SOA	icc.ac.kr. admin.hufs.ac.kr. (	
			1997022712	; Serial
			28800	; Refresh
			14400	; Retry
			3600000	; Expire
			86400	; Minimum
			)	
localhost	IN	A	127.0.0.1	
icc.ac.kr.	IN	NS	icc.ac.kr.	
icc.ac.kr.	IN	A	192.168.7.2	
pis.icc.ac.kr.	IN	A	192.168.7.2	
epcproxy.icc.ac.kr.	IN	A	192.168.7.3	
75.1.sgtin.id.icc.ac.kr.	IN	NAPTR	11 11 "u" "epc+epcis" "!"^.*\$ <a href="http://epcproxy.icc.ac.kr:5001/DEST-pis.icc.ac.kr/product/">http://epcproxy.icc.ac.kr:5001/DEST-pis.icc.ac.kr/product/</a> /"	.
4.2.sgtin.id.icc.ac.kr.	IN	NAPTR	8 9 "u" "epc+epcis" "!"^.*\$ <a href="http://epcproxy.icc.ac.kr:5001/=pis.icc.ac.kr/people/">http://epcproxy.icc.ac.kr:5001/=pis.icc.ac.kr/people/</a> /"	.
84.1.sgtin.id.icc.ac.kr.	IN	NAPTR	10 10 "u" "rfms+epcis" "!"^.*\$ <a href="http://pis.icc.ac.kr/product/">http://pis.icc.ac.kr/product/</a> /"	.

그림 6-3 "icc"도메인의 Zone파일 구성

```

$TTL      86400
@         IN      SOA      hufs.ac.kr.      admin.hufs.ac.kr.      (
                                1997022712 ; Serial
                                28800      ; Refresh
                                14400      ; Retry
                                3600000    ; Expire
                                86400 )    ; Minimum

localhost      IN      A      127.0.0.1

hufs.ac.kr.    IN      NS      hufs.ac.kr.
hufs.ac.kr.    IN      A      192.168.5.3

epcis.hufs.ac.kr.    IN      A      192.168.5.3
rfid.hufs.ac.kr.    IN      A      192.168.4.4

24.1.sgtin.id.hufs.ac.kr. IN NAPTR 10 10 "u" "epc+epcis" "!^.*$http://epcis.hufs.ac.kr/product//"

```

그림 6-4 "hufs"도메인의 Zone파일 구성

## 2. Testbed 구성

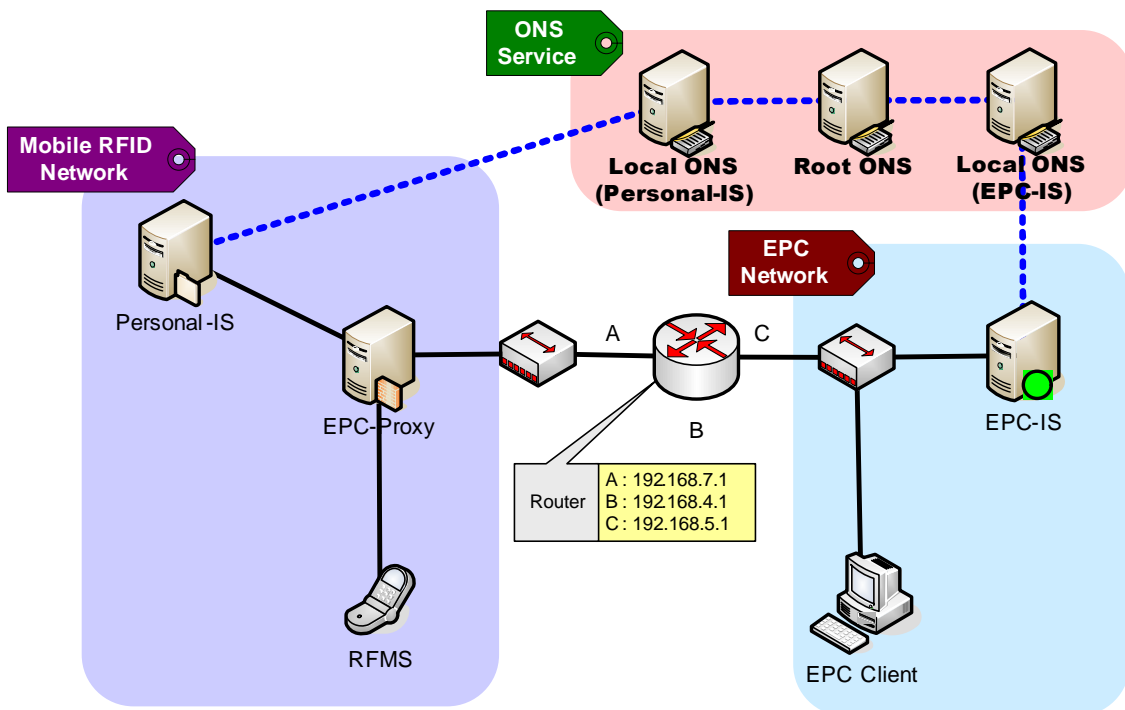


그림 6-5 테스트베드 구성

그림 6-5의 testbed는 Mobile RFID 네트워크와 EPC 네트워크로 분리되며, 실선은 물리적 연결을 나타내며, ONS의 점선은 Delegation Chain을 나타낸다. 두 네트워크는 EPC-Proxy를 통해 HTTP →WSP, WSP→HTTP의 프로토콜 Converting과 Data Conversion을 통해 RFID Code에 매핑되는 정보를 전달 받을 수 있다

## 제2절 서비스 연동 시험

### 1. RFID Code 체계 설계

RFID Code 입력값은 “urn:epc|rfm:id:sgtin:<CodeType>.<2stCodeType>.<Serial Number>”로써 정의한다.

- Serial Number : 물품 혹은 개인정보의 최종 데이터 id가 된다.
- 2stCodeType : CodeType의 소분류로써 CodeType이 2(물품), 2stCodeType은 24 (가전제품)으로 표시한다.
- CodeType : 데이터의 분류 체계중 최상위 클래스로써 물품정보,개인정보 등의 대분류를 뜻한다.
- Prefix : sgtin.id.hufs.ac.kr.
- epc|rfm : EPC Code 또는 RFMS Code 분류를 뜻한다.

RFID Code에 대한 정보 요청시 Serial Number를 제외한 나머지 코드 체계가 ONS 위임체계를 따라 최종 저장소(EPC-IS 또는 Personal-IS)를 찾아낸다.

## 2. Mobile RFID 네트워크에서 EPC 네트워크의 서비스 이용

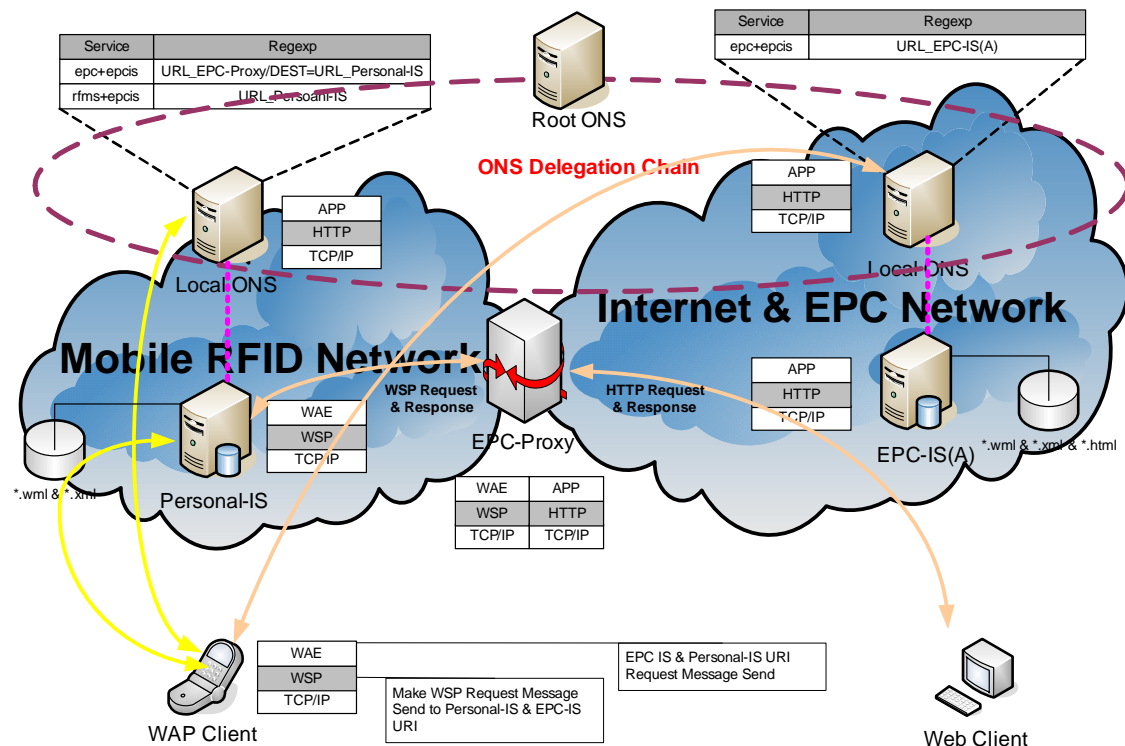
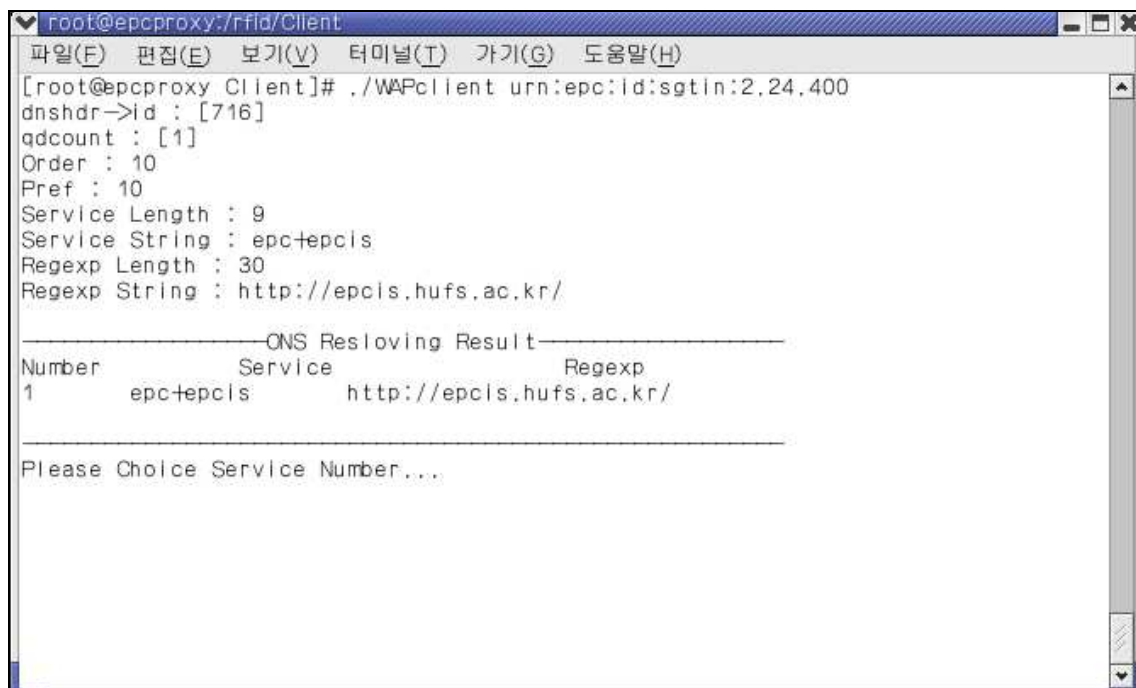


그림 6-6 Mobile RFID 네트워크에서 EPC 네트워크의 정보이용 흐름

그림 6-6 과 같이 Mobile RFID 네트워크에서 EPC 네트워크의 EPC-IS를 사용하는 것을 테스트 하였다. 그림 6-7 은 WAP Client에서 RFID Code를 입력받아 ONS Resolving을 통해 NAPTR RR를 리턴 받은 결과를 보여준다. 그림 6-8은 EPC-Proxy의 의 결과화면이며 Request를 받고, EPC-IS로부터 HTTP Response를 받은 후 그 데이터를 WMLC로 인코딩한 과정을 보여준다. 그림 6-9 처럼 EPC-IS는 HTTP Request 메시지를 파싱하여 해당 데이터를 EPC-Proxy에게 보내준다.





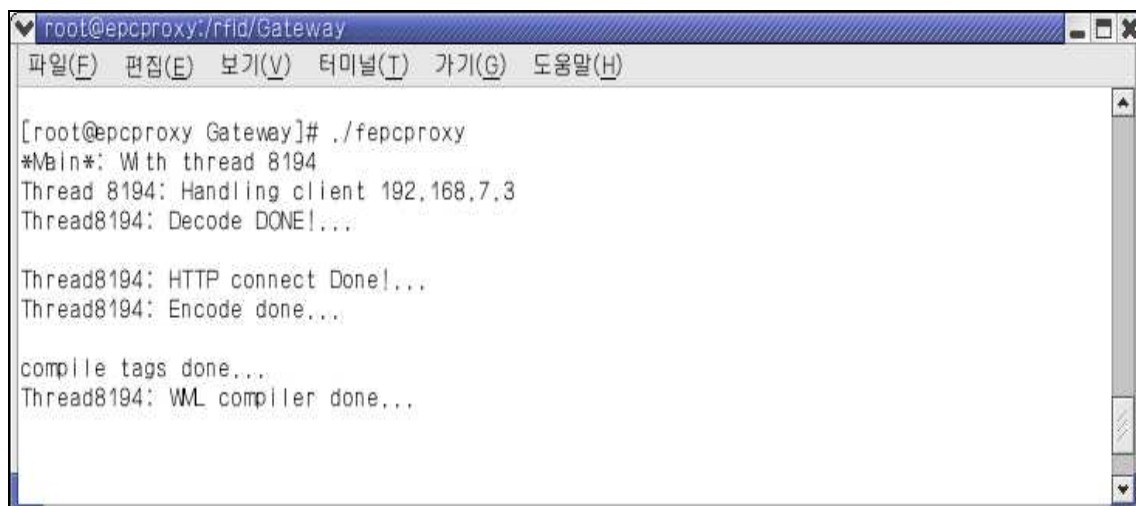
A terminal window titled 'root@epcproxy:/rfid/Client'. The menu bar includes '파일(F)', '편집(E)', '보기(V)', '터미널(T)', '가기(G)', and '도움말(H)'. The command prompt shows '[root@epcproxy Client]# ./WAPclient urn:epc:id:sgtin:2.24.400'. The output displays DNS header information: 'dnshdr->id : [716]', 'qdcount : [1]', 'Order : 10', 'Pref : 10', 'Service Length : 9', 'Service String : epc+epcis', 'Regexp Length : 30', and 'Regexp String : http://epcis.hufs.ac.kr/'. Below this is a section titled 'DNS Resolving Result' containing a table with columns 'Number', 'Service', and 'Regexp'. The table has one row: '1', 'epc+epcis', and 'http://epcis.hufs.ac.kr/'. The prompt 'Please Choice Service Number....' is at the bottom.

```
root@epcproxy:/rfid/Client
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
[root@epcproxy Client]# ./WAPclient urn:epc:id:sgtin:2.24.400
dnshdr->id : [716]
qdcount : [1]
Order : 10
Pref : 10
Service Length : 9
Service String : epc+epcis
Regexp Length : 30
Regexp String : http://epcis.hufs.ac.kr/

-----DNS Resolving Result-----
Number      Service      Regexp
1           epc+epcis    http://epcis.hufs.ac.kr/

Please Choice Service Number....
```

그림 6-7 WAP Client(Resolving 결과화면)



A terminal window titled 'root@epcproxy:/rfid/Gateway'. The menu bar includes '파일(F)', '편집(E)', '보기(V)', '터미널(T)', '가기(G)', and '도움말(H)'. The command prompt shows '[root@epcproxy Gateway]# ./fepcproxy'. The output shows logs for thread 8194: '\*Main\*: Wth thread 8194', 'Thread 8194: Handling client 192.168.7.3', 'Thread8194: Decode DONE!...', 'Thread8194: HTTP connect Done!...', 'Thread8194: Encode done...', 'compile tags done...', and 'Thread8194: WML compiler done...'.

```
root@epcproxy:/rfid/Gateway
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
[root@epcproxy Gateway]# ./fepcproxy
*Main*: Wth thread 8194
Thread 8194: Handling client 192.168.7.3
Thread8194: Decode DONE!...

Thread8194: HTTP connect Done!...
Thread8194: Encode done...

compile tags done...
Thread8194: WML compiler done...
```

그림 6-8 EPC-Proxy 결과화면

```
root@epcis:/home/mclab/epcis/server
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
[root@epcis server]# ./epcis
Handling TCP client 192.168.7.3

GET /400.wml HTTP/1.1
Accept-Charset: iso-8859-1
Accept: application/vnd.wap.wmlc, text/html
Host: epcis.hufs.ac.kr

HTTP/1.1 200 OK
Server: (HttpServer)
Content-Length: 814
Content-Type: text/vnd.wap.wml

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM/DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="no1" title="Card 1">
    <do type="accept" label="OK">
      <go href="#no2">
        </go>
      </do>
    <p>Name : han min kyu</p>
    <p>Age : 19761021</p>
    <p>Fields of work : IT</p>
    <p>Interest : program</p>
  </card>
  <card id="no2" title="Card 2">
    <do type="accept" label="OK">
      <go href="#no3">
        </go>
      </do>
    <p>List of Person</p>
    <p>Name : Lee byung Hee</p>
    <p>Age : 19791006</p>
    <p>Fields of work : IT</p>
    <p>Interest : reading</p>
  </card>
  <card id="no3" title="Card 3">
    <p>List of Person</p>
    <p>Name : back il woo</p>
    <p>Age : 19791008</p>
    <p>Fields of work : art</p>
  </card>
</wml>
```

그림 6-9 EPC-IS 결과화면

그림 6-10 는 WAP Client가 EPC-IS에게 요청한 결과를 웹 브라우저에 출력하고 있다. 이 결과는 EPC-Proxy가 WML을 인코딩하여 WMLC로 보낸 데이터를 WAPC Client가 WML로 재작성하여 보여준 것이다.



그림 6-10 WAP Client(WSP Request 결과화면)

### 3. EPC 네트워크에서 Mobile 네트워크의 서비스 이용

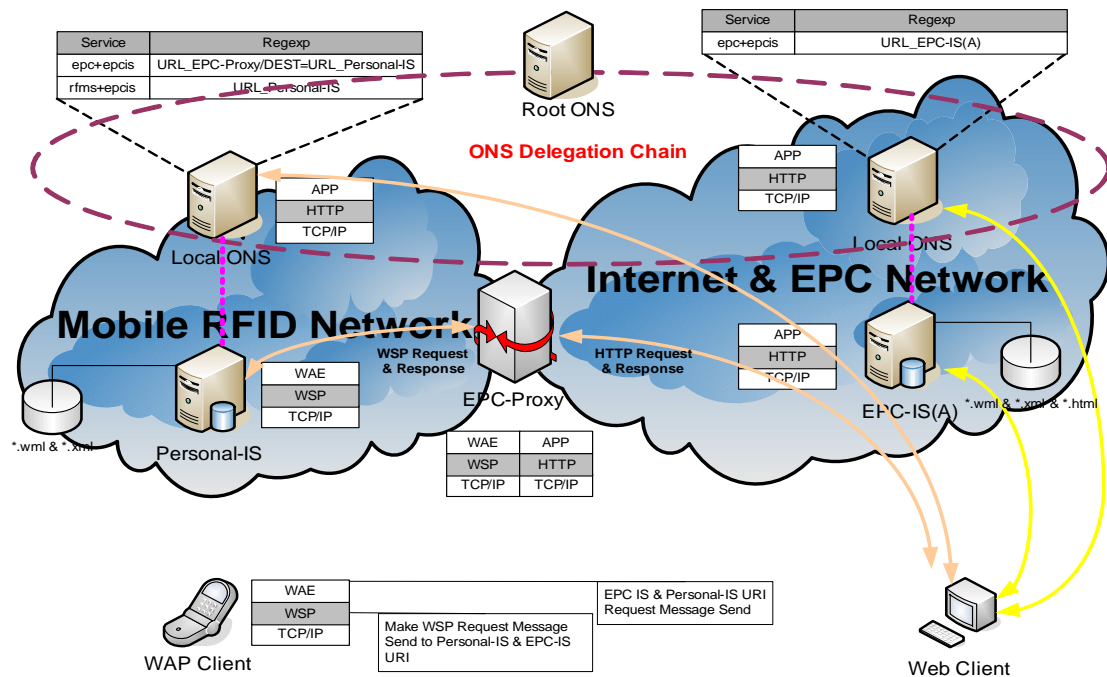


그림 6-11 EPC 네트워크에서 Mobile RFID 네트워크의 정보이용 흐름

그림 6-11 과 같이 EPC 네트워크에서 Mobile RFID 네트워크의 Personal-IS를 사용하는 것을 테스트 하였다. 그림 6-12은 EPC Client에서 RFID Code를 입력받아 ONS Resolving을 통해 NAPTR RR를 리턴 받고 Request를 보낸 상태를 보여준다.

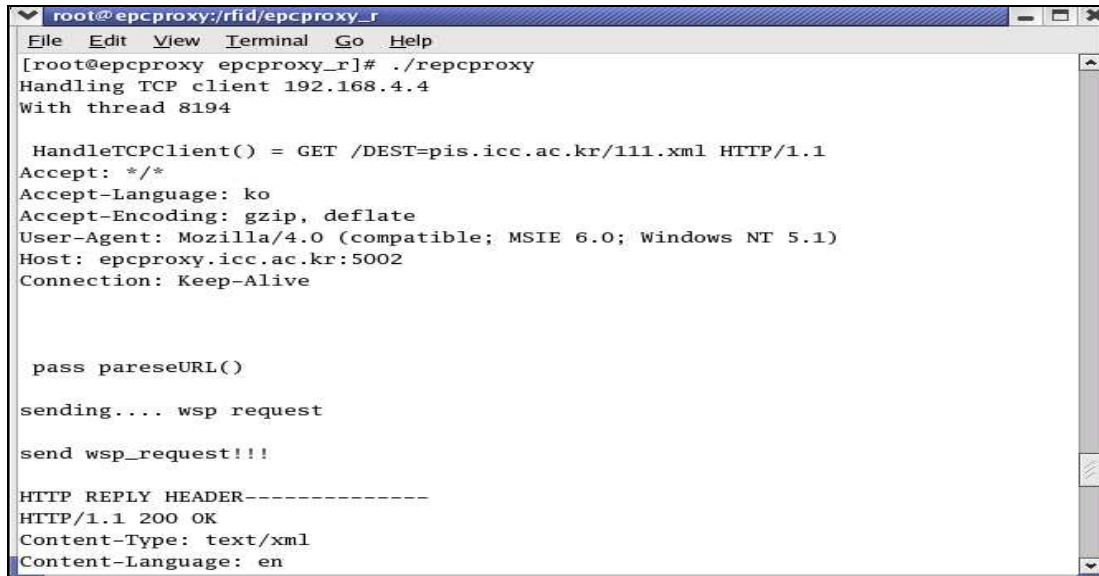
The screenshot shows a window titled "CODE INPUT" with a text field for "RFID CODE" containing "urn:rfm:id:sgtin:2.4.111". Below it, a section titled "NAPTR RR Return Value" displays a table:

Order	Pref	Service	URL
98	9	epc+epcis	http://epcproxy.icc.ac.kr:5002/DEST=pis.icc.ac.kr/

그림 6-12 EPC Client(ONS Resolving) 결과화면

그림 6-13은 EPC Client로부터의 요청을 WSP Request로 만들어주며, 그 결과 Personal-IS

에서 WSP\_Reply(WMLC 데이터)를 받은 후 EPC Client에게 HTTP\_Reply 메시지를 전송하는 과정이다. 그림 6-14는 Personal-IS가 WML데이터를 WMLC로 컴파일 후 EPC-Proxy에게 WSP\_Reply 메시지를 보내는 과정을 보여준다.



```

root@epcproxy:/rfid/epcproxy_r
File Edit View Terminal Go Help
[root@epcproxy epcproxy_r]# ./repcproxy
Handling TCP client 192.168.4.4
With thread 8194

  HandleTCPClient() = GET /DEST=pis.icc.ac.kr/111.xml HTTP/1.1
Accept: */*
Accept-Language: ko
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: epcproxy.icc.ac.kr:5002
Connection: Keep-Alive

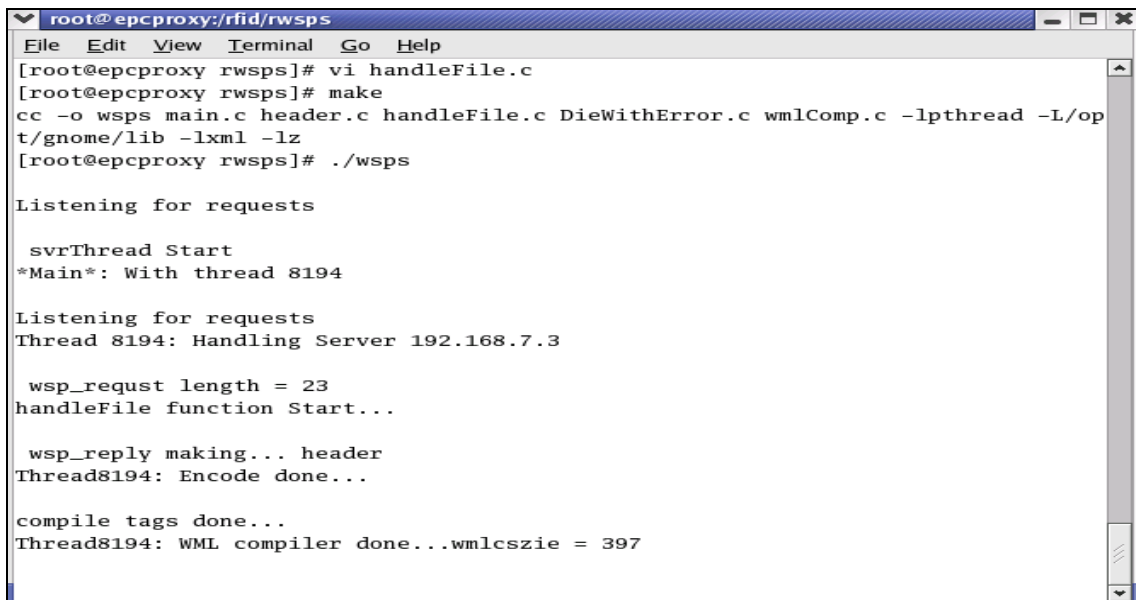
  pass parseURL()

sending.... wsp request

send wsp_request!!!

HTTP REPLY HEADER-----
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Language: en
  
```

그림 6-13 EPC-Proxy 결과화면



```

root@epcproxy:/rfid/rwsp
File Edit View Terminal Go Help
[root@epcproxy rwsp]# vi handleFile.c
[root@epcproxy rwsp]# make
cc -o wsp main.c header.c handleFile.c DieWithError.c wmlComp.c -lpthread -L/opt/gnome/lib -lxml -lz
[root@epcproxy rwsp]# ./wsp

Listening for requests

  svrThread Start
*Main*: With thread 8194

Listening for requests
Thread 8194: Handling Server 192.168.7.3

  wsp_request length = 23
handleFile function Start...

  wsp_reply making... header
Thread8194: Encode done...

compile tags done...
Thread8194: WML compiler done...wmlcszie = 397
  
```

그림 6-14 Person-IS 결과화면

그림 6-15은 EPC Client에게 보내진 최종 결과로써, Personal-IS의 저장되어 있는 RFID와 맵핑된 정보를 웹 브라우저에 보여준 것이다.



그림 6-15 EPC Client Request 결과화면

## 제7장 결론 및 향후 개선 방향

본 논문은 인터넷 및 Mobile 인프라에서의 RFID 네트워크에 대한 국내외 동향과 서비스에 대해 조사분석하고, 인터넷 인프라에서의 EPC 네트워크 Testbed를 구축하여 EPC 네트워크의 구조와 운영방식 및 서비스에 대해 소개하였다.

Mobile 인프라에서의 Mobile RFID 네트워크 구축은 모바일 사용자가 인터넷 인프라에서 제공되는 서비스들을 이용할 수 있도록 기능적 모델을 타입 별로 구분하고, 이에 따른 서비스 모델을 제시하였다. Personal-IS는 Mobile RFID 네트워크가 단순 사물에 대한 정보인식에서 탈피하여, 사용자 관점에서의 서비스 분류를 통해 다양한 서비스가 가능함을 보였다. 또한, EPC-Proxy를 통한 두 네트워크의 서비스 공유방안 연구를 통하여, 유저가 다양한 서비스를 받을 수 있는 기능적 모델을 제시하였다. 이를 통해, Mobile RFID 네트워크가 인터넷 인프라에서 제시하는 네트워크 구조와 상관없이 인터넷 인프라의 RFID 네트워크가 제공하는 서비스를 받을 수 있는 방법을 제시하고 구현하였다.

구현부분의 Testbed는 간단한 네트워크 구조로서, TCP/IP위에 WSP 프로토콜과 WAP 게이트웨이의 단방향을 양방향으로 전송할 수 있게 확장하고, ONS 위임구조를 구성하여 Mobile RFID 네트워크와 EPC 네트워크의 서비스 연동을 테스트할 수 있도록 구성하였다.

테스트 결과, 두 네트워크의 연동방식은 두 가지의 확장모델이 가능하다. 첫 번째는 EPC-Proxy의 역할을 증대 시키는 것이다. 본 구현에서 EPC-Proxy는 “프로토콜의 양방향 변환”, “WMLC를 WML 포맷으로 변환”, “WML을 WMLC로 포맷 변환”, “ONS Configuration Hidding의 해석”의 기능을 하지만, 많은 기능적 모델이 있을수 있다. 또한 RFMS의 기능을 하는 WAP Client에 포함된 리졸버 기능은 핸드폰의 처리능력 부담을 줄이기 위해 EPC-Proxy에 포함될 수 있으며, EPC-Proxy는 Mobile RFID 네트워크의 LoadBalancing을 위해 caching의 역할을 할 수 있도록 확장이 가능하다.

두 번째 확장모델은 Mobile RFID 네트워크의 Personal-IS의 구조적 확장이다. 현재, Mobile 서비스는 인터넷상에 있는 콘텐츠 서버를 통해 서비스 받고 있다. Personal-IS와 웹상의 콘텐츠 서버와 주기적인 동기메시지 교환으로 콘텐츠의 저장과 수정을 동시에 업데이트하여 각 네트워크를 사용하는 유저에게 빠른 Access Service를 제공하는 것이다.

본 논문에서는 Mobile RFID 네트워크에서 가져야 하는 구조적 모델과 서비스 모델을 제시

하고, ONS Configuration Hidding을 통한 Mobile RFID 네트워크와 EPC 네트워크의 서비스 연동을 구현하여 그 가능성을 보여 주었다. 차후 많은 구조적 모델과 연동 방안제시에 많은 기여가 되었으면 하는 바램이다.

## 참 고 문 헌

- [1] EPCglobal Object Name Service(ONS) 1.0, EPCglobal, 29 November 2004.
- [2] EPCglobal Object Name Information Service(OIS) 1.0, EPCglobal, 8 March 2005
- [3] EPCglobal The EPCglobal Architecture Framework Final Version, EPCglobal, 1 July. 2005.
- [4] VeriSign EPC Network Services 1.5, VeriSign, Aug. 2004.
- [5] Auto-IS Savant Specification 1.0, Version of 01 Sep. 2003.
- [6] EPCglobal, "EPC Information Services(EPCIS) Version 1.0 Specification", Working Draft, Oct. 2004.
- [7] D. Senie, "Encouraging the use of DNS IN-ADDR Mapping", INTERNET-DRAFT, April 2004.
- [8] DNS & BIND, 2nd Edition, by Paul Albiz & Cricket Liu O'Reilly.
- [9] WAP-230-WSP Approved Version 5 July. 2001.
- [10] Roy Fielding, "Hypertext Transfer Protocol - HTTP/1.1", Internet Draft (Text / PostScript), IETF HTTP WG, August 1996.
- [11] 인터넷 주소자원 국제동향 보고서, (재)한국인터넷정보센터, IPv4/6 DNS 지침서, 4. 2004.
- [12] RFID 검색시스템 구축 및 운영지침 v1.0, NIDA, 7 Dec. 2004.
- [13] 고희봉, 오영철, 유승화, "RFID 표준화 동향," Telecommunications Review, 제15권, 2호, pp. 244-256, 4. 2004.
- [14] 조영빈 "모바일 RFID 사업 동향 및 전망 ," Telecommunications Review, 제15권, 2호, pp. 229-243, 4. 2005



- [15] RFID 기술개요 및 국내외 동향분석, 김상태 IITA 8. 2003
- [16] RFID 기술 및 산업동향, 박주상 TTA e-logistics
- [17] <http://www.verisign.com>
- [18] [http://www.epcglobalinc.org/standards\\_technology/Standard\\_Development\\_Process.html](http://www.epcglobalinc.org/standards_technology/Standard_Development_Process.html)
- [19] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", RFC1035, November 1987.
- [20] M.Mealling, "The Naming Authority Pointer (NAPTR) DNS Resource Record", RFC2915, September 2000.
- [21] O'Reilly - HTTP Pocket Reference.chm
- [22] R.Fielding, "Hypertext Transfer Protocol - HTTP/1.1", RFC2068, September 2000.
- [23] <http://www.w3schools.com/wap/tutorial>
- [24] 홍진표, Technical Document "RFID Service Classification on Mobile Network", MCLAB-D-01, May 2005.
- [25] 홍진표, Technical Document "Mobile RFID-specific Service Model and Information Flow", MCLAB-D-02, 홍진표, June 2005.
- [26] 홍진표, Technical Document "The RFID Service Classification on Mobile Network Final Version", MCLAB-S-01, Sept. 2005.