TETRA 기반의 전력 IT Backbone Network 접속을 위한 Gateway 플랫폼 구조

천대원*, 송병권**, 정태의*
*서경대학교 컴퓨터과학과
**서경대학교 정보통신공학과
chundalbong@gmail.com* bksong@skuniv.ac.kr** tejeong@skuniv.ac.kr*

Gateway Platform Structure based on the TETRA for the Connection of Electric Power IT Backbone Network

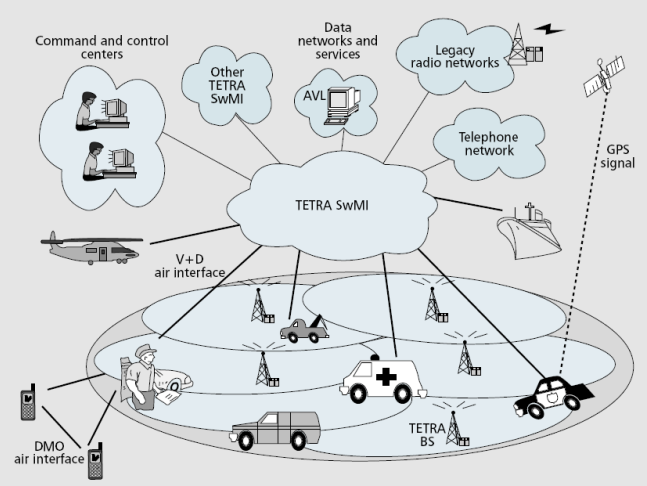
Dae-Won Chun*, Byeong-Kwon Song**, Tae-Eui Jeong*
*Dept of Computer Science, Seokyeong University
**Dept of Information and Communications Engineering, Seokyeong University

요 약

본 논문에서는 대표적인 산업용 프로토콜인 DNP(Distributed Network Protocol) 3.0과 TETRA(Terrestrial Trunked Radio)의 Gateway System Architecture를 제안한다. TETRA는 재난망 무선통신 기술로써 여러 가지 타입의 무선 인터페이스를 제공한다. 이러한 TETRA 기술을 이용해서 원거리에 있는 각 산업용 장비들의 감시·제어를 보다 효율적으로 수행 할 수 있다. 하지만 아직 TETRA의 국내 표준이 이루어지지 않은 상태이기 때문에 본 논문에서는 다양한 산업용 장비와 인터페이스를 할 수 있는 표준을 제공하고자 한다.

1. 서론

TETRA(Terrestrial Trunked Radio)는 국가 재난망 무선통신기술로써, 각 기관의 통신망은 평상시 고유의 업무를 수행하다가 재난재해 발생 시 에는 통합지휘 무선통신망의 기능을 수행하는 System을 말하는 것으로써 다음과 같은 여러 가지 타입의 Air Interface를 지원한다. 음성과 데이터를 보낼 수 있는 V+D(Voice plus Data) Air Interface, 패킷 데이터를 전송할 수 있는 PDO(Packet Data Optimized) Air Interface 그리고 두 단말기 간에 직접 통신을 할 수 있는 DMO(Direct Mode Operation) Air Interface이다. 이러한 다양한 타입의 인터페이스는 긴급한 상황을 알리기 위해 [그림1]과 같이 여러 가지 방식으로 사용된다[1]. 그림에서 TETRA는 자신의 망에서 뿐만 아니라, WLAN과 같은 다른 프로토콜과도 인터페이스를 할 수 있다. 이러한 특징을 이용해서 DNP(Distributed Network Protocol) 3.0, IEC61850, Modbus와 같이 주로 산업용 장비의 감시·제어를 위한 프로토콜과 인터페이스 함으로써, 원격지에 있는 장비에 특정 명령을 내리거나, 장비로부터 필요한 정보를 얻어오는 등의 동작을 수행할 수 있다. 하지만 아직 TETRA와 산업용 프로토콜 간의 인터페이스가 정의 되어 있지 않기 때문에 이러한 기능을 이용할 수 없다. 본 논문에서는 대표적인 산업용 프로토콜인 DNP 3.0과 TETRA간 Gateway를 설계하고, TETRA Interface API를 정의함으로써 차후 TETRA를 이용한 산업용 장비의



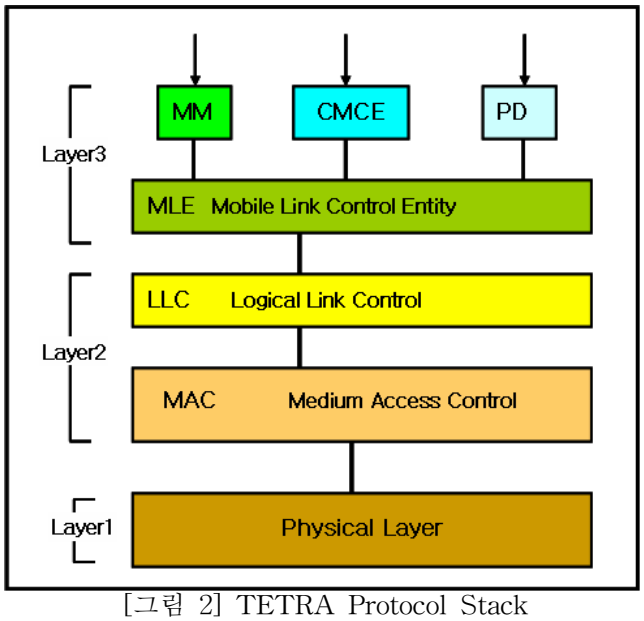
[그림 1] TETRA 기반의 Telecommunication

※본 연구는 전력산업연구개발 위탁 연구비로 수행되었음

감시·제어를 용이하게 하는 것을 목표로 한다.

2. TETRA

TETRA는 [그림 2]와 같이 총 3개의 Layer로 구성 되어 있다. 2계층은 다시 세부적으로 LLC와 MAC으로 나뉘어지고, 3계층 역시 크게 3.1-MLE와 3.2-SNDCP, CMCE, MM으로 나뉘어진다. 각 Layer의 Sub-Entity들이 수행하는 기능을 살펴보면, 먼저 SNDCP(SubNetwork-Dependent Convergence Protocol) Entity는 여러 프로토콜 간의 통신을 할 수 있게 해주는 다리 역할을 하는 계층으로써 NSAPI(Network Service Access Point Identifier)라는 Parameter를 이용하여 프로토콜을 구분한다. NSAPI는 총 2byte가 할당되어 있으며, 총 16개의 프로토콜을 구분할 수 있다. 예를들면 IPv4=1, IPv6=2 이와 같은 방식으로 어떤 프로토콜에 번호를 부여한다[2]. MM(Mobile Management) Entity는 이동성을 관리하고[3], CMCE(Circuit Mode Control Entity)는 통화와 통화품질 감시에 관련된 기능을 처리하는 Entity이다[4]. 실제로 Data의 전송과 관련 있는 Entity는 SNDCP의 SN-DATA Primitive와 SN-UNITDATA 그리고 CMCE 내부의 Sub-Entity인 SDS의 SDS-DATA Primitive에서만 수행할 수 있다.

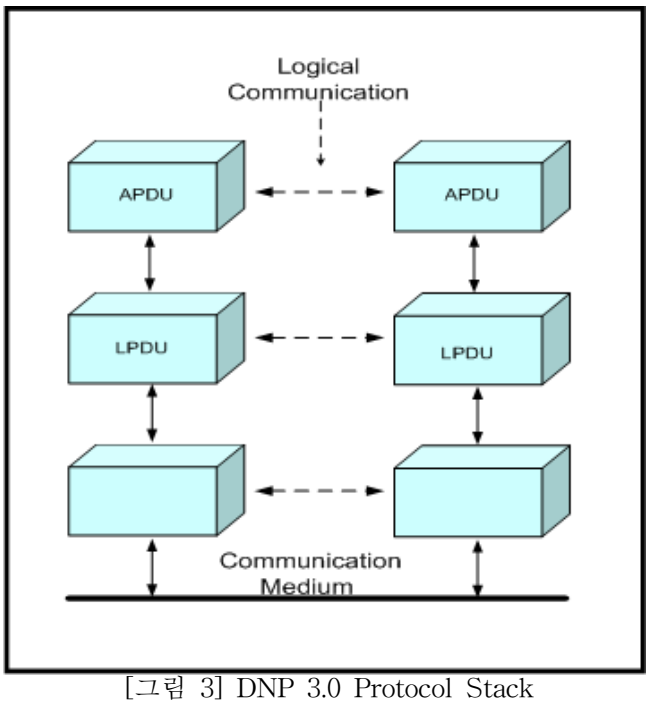


[그림 2] TETRA Protocol Stack

본 논문은 TETRA 3계층의 User와 Provider(L2)로부터 수신한 Data에만 중점을 두고 있기 때문에 세부적인 사항은 TETRA 표준문서에서 습득할 것을 요한다.

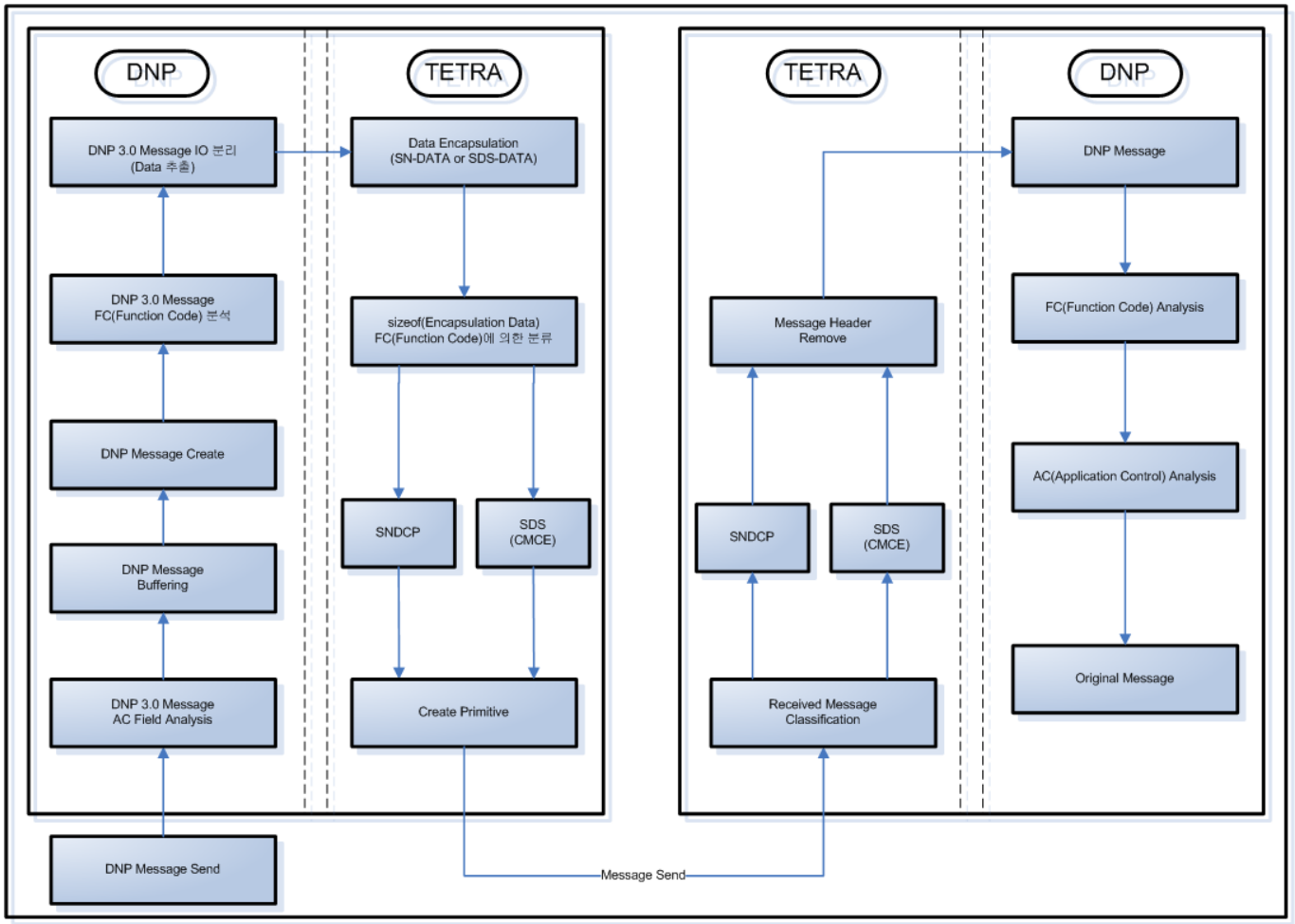
3. DNP 3.0

DNP 3.0은 장비를 감시·제어하기 위해 사용되는 산업용 프로토콜로써 Master와 Slave의 개념을 적용한다. Master는 Slave를 제어하기 위해 주로 Read, Write, Restart와 같은 명령을 보내고, 명령에 대한 Response를 기다린다. 반대로 Slave는 Master로부터 특정 명령을 받았을 경우, 이에 대한 처리를 한 후 Master에게 응답을 보낸다. 하지만 Slave는 Master의 명령에 의한 응답뿐 아니라, 장치에 특정 Event가 발생했을 경우, Unsolicited Response를 사용해서 Master에게 Message를 전송할 수 있다. Master는 또다른 Master와 연결될 수 있고, 하나의 Master는 여러 개의 Slave를 관리한다.



[그림 3] DNP 3.0 Protocol Stack

DNP 3.0은 [그림 3]과 같이 총 3개의 Layer로 구성되어 있지만, IEC Document 8702-5-1에 의해 정의된 Frame Size(최대 16byte)보다 더 큰 메시지(256byte)를 전송하기 위해 2계층과 3계층 사이에 Psudo Transport Layer를 사용할 수 있다[5]. 먼저 3계층이 수행하는 일은 사용자로부터 받은 메시지가 어떤 종류의 메시지(혹은 명령)인지를 구분해 주는 기능과 User로부터 큰 Data를 받았을 때, 이를 Segment하고, Reassemble하는 역할, 원격 장치의 특정 영역의 Data를 Read하거나 Write할 때, 범위를 지정해주는 기능을 수행한다. 2계층은 실질적인 Data의 통신을 담당하는 계층으로써 Error Check나 Data의 전송 방향을 제어하는 기능을 수행한다.



4. TETRA-DNP3.0 Gateway Architecture

[그림 4]는 DNP Application Layer와 TETRA 3계층 상위에서 DNP-TETRA간 메시지를 변환시켜주는 TSCML(TETRA-SCADA Cross Mapping Layer)의 Architecture이다 .TSCML은 DNP의 Master 또는 Slave 로부터 수신한 메시지를 Encapsulation 한 후, TETRA를 통해 전송, 이를 수신한 DNP측에서 Decapsulation하는 일련의 과정을 보여주고 있다. 메시지의 변환 과정을 세부적으로 살펴보면, 먼저 TSCML은 DNP장비로부터 특정 메시지를 수신한다. 수신한 DNP 메시지의 AC(Application Control) header에는 Fragment된 메시지의 순서를 구분할 수 있는 Bit가 포함되어 있다. 처음을 나타내는 FIR bit와 마지막을 나타내는 FIN bit가 존재한다. FIR=1이고 FIN=0일 경우는 Fragment된 메시지의 처음을 의미하고, FIR=1 FIN=1일 경우는 연속되는 메시지, FIR=0 FIN=1일 경우는 Fragment된 메시지의 마지막을 의미한다. TSCML은 Fragment된 메시지를 임시 버퍼에 저장하고 마지막을 나타내는 FIN bit가 Set된 메시지가 도착했을 경우 완전한 하나의 DNP 메시지를 생성한다. 생성된 메시지의 목적을 알아보기 위해 FC(Function Code)를 분석한다. FC의 종류는 크게 8가지로 분류된다. 주로 많이 쓰이는 기능코드들에 대해 살펴보면 Transfer와 관련된 코드로는 Read, Write, Confirm이 있고, Control과 관련된 코드로는 Select, Operation과 같은 코드들이 있다. FC까지 분리가 되면 다음은 실제 Data부분이 남는다. 실제로 Data는 Object Header와 IO로 분리가 되는데 본 Gateway에서는 Object Header를 분석하지 않고, Data부분 전체를 Encapsulation해서 수신측에서 분석을 하도록 설계하였다. 이렇게 얻은 Data를 TETRA로 전송하기 위한 Encapsulation 작업이 필요한데, 2가지 방식의 Encapsulation 방식이 존재한다. SNDCP와 CMCE 내부의 SDS 부개체가 Data를 전송하는 기능을 가지고 있는데, 어떤 방식으로 Encapsulation 할지는 앞에서 분석한 FC와 Data의 Size로써 결정한다. Encapsulation된 Data는 TETRA Primitive를 이용해서 전송이 되며, 이를 수신한 측에서는 Decapsulation 과정을 수행하면 본래의 메시지를 얻을 수 있다. Decapsulation 과정에서는 Data의 Object Header를 분석함으로써 실제로

원하는 결과 값을 산출해야 한다.

5. 구현

프로토콜은 프로세스 모델과 라이브러리 모델 두 가지 방법으로 구현할 수 있는데, 이 모델은 두 가지 방법을 모두 사용해서 구현한다. 하나의 Master에 여러 개의 Slave가 연결되어 있기 때문에 독립된 여러 개의 처리를 위한 Thread를 사용해야 하며, 재전송을 통한 데이터의 신뢰성 있는 전달을 위해 독립된 Timer 프로세스를 사용한다. 그리고 실제적인 Data 전송을 구현하기 위해서 Primitive 함수를 사용한다. [그림 5]는 실제 사용되는 SN-DATA.req와 SN-DATA.ind 함수의 Parameter와 함수의 형태를 보여준다.

Primitive / Parameter	req	ind
NSAPI	M	M
Priority	O	-
N-PDU	M	M

SN_DATA_req(int NSAPI, int Priority, struct N_PDU)

[그림 5] Request, Indication Primitive

6. 결론

현재 국내의 국가 재난망은 여러 분야에 걸쳐서 사용되고 있지만 실질적인 국내의 표준이 아직 확립되어 있지 않기 때문에 상호 연동에 어려움이 있다. 따라서 TETRA에서 사용되는 표준 API의 개발이 시급한 시점이다. 본 논문은 산업용 프로토콜인 DNP와 국가 긴급 재난망인 TETRA에 대하여 간단히 설명하였고, DNP와 TETRA의 Gateway 구현 시 고려해야 할 사항들과 응용 프로그래밍 인터페이스 설계에 대해 언급하였다. 지금 현재는 대표적인 프로토콜인 DNP에 대한 예를 들었지만, 장차 나아가 IEC61850이나 Modbus와 같은 산업용 프로토콜 전반에 걸쳐서 사용될 수 있는 Gateway가 될 수 있도록 연구를 진행하고 있다.

참고 문헌

[1] Apostolis K. Salkintzis, Motorola "Evolving Public Safety Communication Systems by Integrating WLAN and TETRA Networks"

[2]ETSI EN 300 392-2 "Terrestrial Trunked Radio(TETRA) : Voice plus Data(V+D); Part 28 : TETRA Packet Data Protocol

[3]ETSI EN 300 392-1 "Terrestrial Trunked Radio(TETRA) : Layer 3 Air Interface; Part 6 : Mobile Management Service

[4]ETSI EN 300 392-2 "Terrestrial Trunked Radio(TETRA) : Voice plus Data(V+D); Part 14 : CMCE Protocol

[5]DNP User Group "Distributed Network Protocol V3.00 Documentation"