전력 IED에 간단하게 DNP 탑재를 위한 효율적 DNP API 개발

임희택*, 최면송**, 김태완***, 이승재****, 임성정*****
명지대*, 명지대**, 명지대***, 명지대****, 한국전기연구원*****

Development of DNP API for simply applying to power IED

Hee-Taek Lim*, Myeon-Song Choi**, Tea-Wan Kim***, Seung-Jae Lee****, Sung-Jung Rim*****
Myongji University*, Myongji University**, Myongji University***, Myongji University****, KERI*****

Abstract - 현재의 전력산업에서 통신은 필수 항목이 되었다. 따라서 전력장치의 대부분이 통신 기능을 가지고 있으며 전력장치 개발을 위해서는 통신 탑재가 필수이다. 본 논문에서는 전력분야에서 가장 널리 사용되고 있는 DNP(Distributed Network Protocol)를 간편하게 전력장치에 탑재할 수 있는 DNP API를 제안한다. 제안된 DNP API는 Recloser에 DNP를 탑재하여 편리성을 검증하고, Test Harness를 통하여 통신 신뢰도를 검증하였다.

1. 서 론

기존의 아날로그 기기가 주종을 이루던 전력산업은 IT 기술을 융합하여 실시간 통신으로 운전, 제어 및 감시가 가능해 졌다. 이를 바탕으로 전력산업은 계속해서 신기술을 창출해 내고 있으며 전력계통이나 시스템, 장비들은 고도의 지능화를 이루고 있다. 현재는 전력산업과 통신이 매우 밀접한 관계를 가지고 있으며 개폐기의 원격제어, 배전용 변압기의 원격 부하측정 및 열화 진단 등 전력산업의 거의 모든 분야에 통신기술이 접목되어 있다.

전력분야의 장치들을 개발하기 위해서는 통신 기능이 필수 사항이 되었으며 많은 산업체에서 통신 기능을 지원하기 위해 많은 시간과 노력을 들이고 있다.

전력분야에 적용되는 통신 환경은 데이터 전송량이 작지만 빠른 전송이 이루어져야 한다. 또한, 강력한 에러제어기능이 필요하다, 주로 공장이나 변전소, 광범위한 계통의 곳곳에 설치되는 기기는 다양한 전자기 노이즈에 노출되어 있다. 이러한 열악한 통신환경에서 신뢰성 있는 통신기술을 수행하기 위해서 강력한 에러제어 기능을 가져야한다[1].

이러한 전력분야에서 주로 사용되는 통신 프로토콜로는 Modbus, IEC-60870, DNP(Distributed Network Protocol) 3.0, MMS/UCA2.0, IEC-61850등이 있다. 이러한 통신 프로토콜들의 탑재를 위해서는 상당한 통신지식을 필요로 하며 OS의 종류, 컴파일러의 종류에 따라 그 적용이 매우 까다롭다.

본 논문에서는 현재 국제적으로 널리 사용되고 있는 DNP의 편리한 탑재를 위하여 DNP API를 제안한다. DNP API를 사용함으로써 DNP를 처음 접하는 사람들도 해당 기기에 DNP를 쉽게 탑재할 수 있도록 하여 전력분야의 산업체들이 DNP통신기능을 지원하기 위하여 들이는 시간과 노력을 줄일 수 있다.

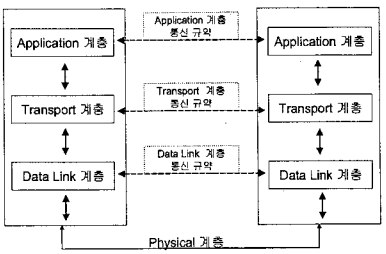
본 논문에서 제안한 DNP API의 검증은 실제 전력분야의 장치인 Recloser에 DNP를 탑재하고 Master / Slave Simulator를 개발함으로써 그 편리성과 효율성, 신뢰성을 검증하였다.

2장에서는 DNP의 구조와 특징, DNP를 탑재 시 불편한 점을 살펴보고, 3장에서는 제안된 DNP API의 구조와 역할, 특징에 대해 설명하고, 4장에서는 사례연구를 통해 제안된 DNP API의 편리성, 효율성, 신뢰성을 검증할 것이다.

2. DNP의 구조와 특징

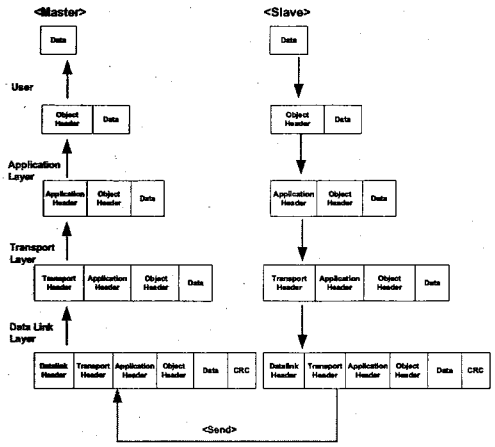
2.1 DNP의 구조

DNP는 IEC에서 정의한 제어통신의 특징인 짧은 응답시간, 잡음에 강한 특성, 저속의 통신네트워크 환경에서의 적응성을 모두 만족한다. 따라서 현재 국제적으로 널리 사용되고 있다. DNP는 OSI 참조 모델과 동일한 7계층은 아니지만 IEC의 표준 권고 안에서 정한 Physical, Data Link, Application 등 3개의 계층에 Pseudo-transport 계층을 추가하여 4계층으로 구성된다.



<그림 1> 프로토콜 모델 체계

Application 계층은 사용자 프로세스로부터 받은 데이터에 Application 계층에서 요구되는 제어정보들을 붙여 Transport 계층으로 보내거나 반대로 Transport계층으로부터 올라온 데이터를 분석하여 분석된 데이터를 사용자 프로세스에 제공하는 역할을 한다. Transport 계층은 큰 데이터를 효율적으로 전송하기 위하여 최대 249Byte 길이로 데이터를 분해해서 Data Link계층으로 보내거나 반대로 분해되어 받은 데이터를 다시 복구하여 Application 계층으로 전송한다. Data Link계층은 시스템간의 실질적인 통신을 담당하여 그 헤더에는 목적지 주소, 발신지 주소, 크기, 제어정보를 가지고 있으며, 에러방지도 이 계층에서 수행된다. 일반적인 경우의 Response시 데이터 처리 과정을 그림 2에 나타내었으며 Request시는 Response의 반대방향으로 이루어진다.



<그림 2> Request에 대한 Response 과정

2.2 DNP탑재 시 문제점

DNP는 매우 큰 프로토콜이므로 해당 장치의 특성과 기능에 상관없이 전체 프로토콜을 모두 적용하여 장치를 개발하는 것이 비효율적이다. 따라서 해당 장치에 맞게 DNP의 구현 범위를 정하는 과정이 필요하다.. Data Object와 Qualifier의 적용범위, Polling 방법, Unsolicited Reporting 등을 개발자가 개발 할 장치의 특성에 맞게 분석하여 정해야 한다.

DNP 탑재 과정에 있어서는 첫째 Physical 계층을 어떻게 구성할 것인가를 정의해야 한다. DNP에는 Physical 계층에 대한 정해진 규약이 없기 때문에 실제 구현에 있어 각 회사마다 자사 제품에 맞는 방식으로 개발하고 있다. 또한 해당 장치의 환경에 따라 Physical 계층에서의 구현이 달라지므로 개발자는 DNP와 해당 장치의 환경에 맞게 Physical 계층을 구성해야 한다.

둘째 개발환경에 따라 데이터 처리과정에서 생기는 오류를 확인하고 개선해야 한다. DNP는 1byte 단위로 데이터를 처리하는 구조를 갖는다. 따라서 어떤 데이터를 1byte로 분해하거나 2byte 혹은 4byte로 합성하는 과정을 취하므로 해당 장치의 마이크로프로세서가 LSBF(least-significant-byte-first) format인지 MSBF(first-significant-byte-first) format인지에 따라 다른 구조를 가지고 있어야 한다. 또한 마이크로프로세서의 특징에 따라 데이터 타입들의 크기가 달라진다. Imbedded system의 경우 char를 2byte로 처리하는 경우가 많은데 이 경우 라이브러리의 수정이 불가피하다. 또한 Compiler의 특성에 따라서도 문제를 일으킬 수 있다.

셋째 DNP의 데이터 처리과정과 통신 과정에서의 모든 setting을 정의하고 해당 장치의 여건에 따라 개발 후에도 변경해야할 setting들을 구현해야 한다.

넷째 DNP의 data와 해당 장치의 변수의 매핑이 필요하다. 변수의 타입과 DNP data의 타입을 분석하여 적절히 매핑 되어야 한다.

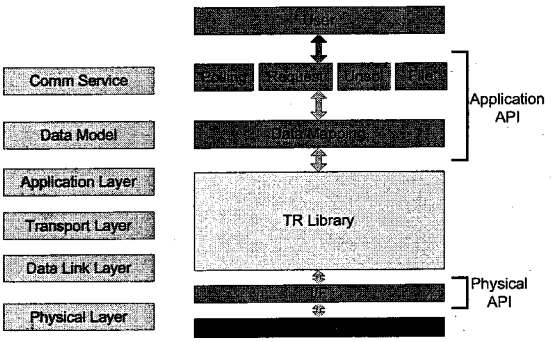
다섯째 DNP 통신이 잘 이루어지는가에 대한 신뢰도 평가가 필요하다. 이러한 과정들은 해당 장치에 DNP를 탑재하려는 개발자가 DNP의 복잡한 모든 데이터 처리 과정과 통신에 대한 지식을 숙지하고 있어야 가능하다.

본 논문에서는 이러한 과정들을 DNP의 데이터 처리 과정이나 통신에 대한 많은 지식 없이도 쉽게 DNP를 탑재할 수 있는 DNP API를 제안한다.

3. DNP API 개발

3.1 DNP API의 구조

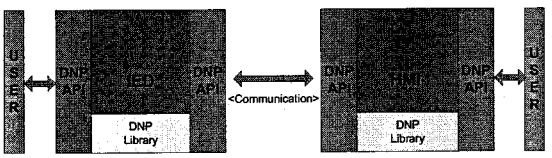
DNP SCL(Source Code Library)는 Data Link 계층과 Pseudo-transport 계층 Application 계층에 해당된다. DNP API는 그림 3에서와 같이 개발환경에 따라 DNP SCL과 운영체제 혹은 DSP와의 interface역할 을 수행하는 구조를 가지고 있고 DNP data 변수와 사용자 정의 변수와의 매핑 구조, DNP 통신 서비스를 편리하게 사용할 수 있는 구조를 가지고 있다. 또한 개발환경에 따라 (Windows, Linux, Non OS(DSP)) DNP SCL을 수정하였기 때문에 개발자는 개발환경을 선정하고, 구현 범위 (Subset Definition), Object Index만 정리 하면 어디에나 간단하게 DNP를 탑재할 수 있는 구조를 가지고 있다.



<그림 3 > DNP API의 구성

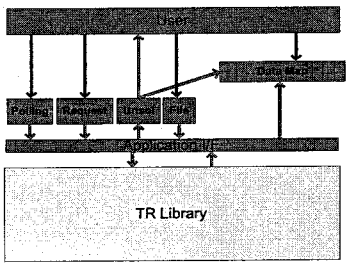
3.2 DNP API의 역할과 특징

DNP API는 사용자가 DNP와 접근하지 않고 DNP API만을 접근하여 DNP 통신을 가능하도록 하는 방식을 취한다. 따라서 DNP 내부에서 이루어지는 데이터의 처리 과정을 사용자가 숙지 할 필요 없이 DNP API를 다룸으로써 DNP를 탑재하고 모든 DNP의 통신 서비스를 이용할 수 있다(그림 4).



<그림 4> DNP API의 역할

DNP의 통신서비스에는 Polling, 단발적인 Request, Unsolicited message, File transfer가 있다. DNP 라이브러리에서 이들의 기능을 사용하기 위해서는 라이브러리의 함수들과 구조를 이해하고 있어야 가능하다. DNP API는 이러한 불편함 없이 API의 Function을 사용함으로써 편리하게 이용할 수 있다. 통신 서비스별로 API의 Function을 이용하여 매개변수로 Setting 값을 전달하여 사용하기만 하면 편리하게 DNP의 통신서비스를 사용할 수 있다(그림 5).



<그림 5> Application API의 구조

DNP API는 데이터 사용의 편리성을 가지고 있다.

요청에 의해 data를 취득하거나 Unsolicited massage로 인해 취득된 데이터는 DNP의 Memory DB에 저장되게 된다. DNP Library의 경우 요청에 의한 데이터와 Unsolicited massage에 의한 데이터에 대해 특별한 구분이 없기 때문에 Unsolicited massage에 의한 데이터가 매우 중요한 data로서 즉시 사용되어져야 할 경우 매우 불편하다. DNP API는 Function pointer를 사용하여 Unsolicited message를 취득할 경우 data의 타입, index, 시간, 값을 사용자에게 전달하고 DNP Memory DB에 저장하는 기법을 사용하였다. 따라서 DNP API를 이용할 경우 특별히 DNP Memory DB를 감시할 필요가 없이 Unsolicited message에 대한 빠른 처리가 가능하다.

DNP API는 데이터 매핑의 편리성을 가지고 있다. DNP 탑재를 위해서는 사용자의 변수와 DNP의 변수간의 매핑을 필요로 한다. DNP API는 매핑구조를 가지고 있다. 사용자 변수의 주소 값을 전달하여 사용자의 변수와 DNP의 변수간의 매핑을 할 수 있는 구조가 이미 있기 때문에 사용자는 매핑에 대한 고민 없이 DNP API가 제공하는 Mapping 구조에 사용자 변수의 주소 값을 전달해 줌으로써 매핑 할 수 있다. 또한 DNP는 Point에 대한 Class 등록이 필요하다. DNP API는 Point에 대한 Class 등록에 대한 구조도 만들어져 있기 때문에 사용자는 해당 Point에 대한 Class만 정의하면 된다.

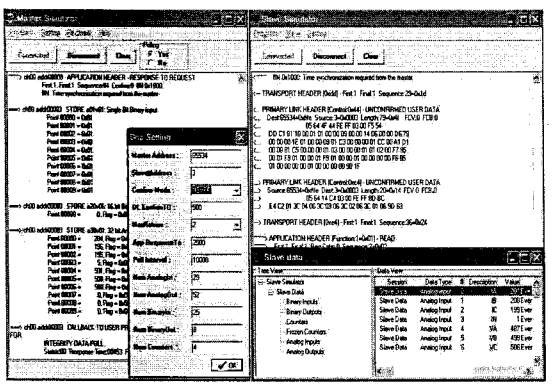
DNP API의 역할은 DNP와 사용자간, DNP와 개발 장치간의 접속을 모두 DNP API를 통하여 하게 함으로써 DNP API라는 DNP와 장치 간, DNP와 사용자 간의 편리한 매개체를 사용해서 편리하게 DNP를 탑재하고 운영하게 하는 것이다.

4. 사례 연구

4.1 Master / Slave Simulator 개발을 통한 검증

개발된 Master Simulator와 Slave Simulator(그림 6)는 Borland C++ builder 6을 사용하여 개발하였다. Master Simulator와 Slave Simulator의 기능은 화면에 통신과정을 보여주고 DNP의 각종 setting들과 통신설정들을 변경할 수 있으며 Data의 정보와 변경사항들을 보여준다. 이 Master/Slave Simulator를 통해서 Master 혹은 Slave가 정상적으로 DNP통신을 수행할 수 있는지 평가할 수 있다.

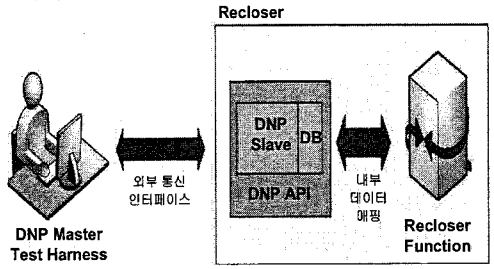
본 시뮬레이터는 개발된 DNP API를 통해 개발하였고 DNP 통신관련 부분은 모두 DNP API의 함수들을 사용함으로써 단기간에 쉽게 개발하였다. 또한 Triangle 사의 Test Harness를 통해 통신기능을 검증하였다. 따라서 DNP API를 이용함으로써 간편하게 DNP를 탑재하고 신뢰성을 유지할 수 있음을 검증하였다.



<그림 6> Master / Slave Simulator

4.2 Recloser에 DNP 탑재를 통한 검증

Recloser에 DNP API를 이용하여 DNP를 탑재함으로써 DNP API의 신뢰도를 검증하였다. Recloser의 마이크로프로세서는 TMS320C2812이며 개발환경은 Code composer 3.1을 사용하였다. 탑재된 DNP의 통신 신뢰도 검증은 Triangle사의 Test harness와 Master Simulator를 통해 검증하였다(그림 7).



<그림 7> Recloser에 DNP탑재를 통한 검증

따라서 DNP API는 OS환경 뿐 아니라 Imbedded 환경에서도 높은 편리성과 신뢰도를 가지고 있음을 검증하였다.

5. 결론

현재에는 전력산업에서 통신은 필수 항목이 되었다. 대부분의 장치나 시스템들은 통신기능을 가지고 있고 한 장치에 DNP, IEC60870, Modbus등 다수의 프로토콜을 내장하는 추세이다. 따라서 전력 개발자들은 장치에 통신프로토콜을 탑재하기 위해 많은 시간과 노력이 필요하다.

본 논문에서 소개된 DNP API는 간편하게 DNP를 탑재하는 기법을 취해 개발자가 DNP의 상세한 특성과 기능을 모르더라도 DNP API에서 제공하는 함수들과 구조체를 사용함으로써 쉽게 DNP 탑재하고 신뢰도를 가질 수 있으므로 개발자의 시간과 노력을 줄여줄 것으로 기대된다.

본 논문에서 소개된 DNP API는 향후 DNP 뿐 아니라 IEC60870, Modbus, OPC등도 하나의 API에 포함시켜 그 활용가치를 높일 예정이다.

본 논문에서 소개된 API가 전력개발자들에게 많은 도움이 되기를 기대해본다.

감사의 글

본 연구는 에너지자원 인력양성 사업의 전력IT 인력양성사업 지원으로 수행되었으며(전력IT 인력양성 사업센터), 과학기술부/한국과학재단 우수연구센터 육성사업의 지원으로 수행되었음(차세대전력기술연구센터)

[참 고 문 헌]

[1] 박종찬, 김병진, "CAN 통신을 기반으로 한 전격 시스템 자동화 구축", KIEE, 52P-3-1, p95~99, 2003

[2] DNP User Group, "Distributed Network Protocol 3.0 Basic 4 Documents",

[3] IEEE, "IEEE Recommended Practice for Data Communications Between Remote Terminal Units and Intelligent Electronic Devices in a Substation ", IEEE Std 1397-2000, 2000