JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 20, No. 7 : 1335~1341 Jul. 2016

실시간 철도안전 관제를 위한 프로토콜 변환 방안 연구

안진[1*] · 김성민[2]

Protocol converting method for the Real-time Safety Supervision System in Railway

Jin-Ahn[1*] · Sung-Min Kim[2]

[1*]Railway Technical Research Center, DAEATI Co., Ltd, Bucheon 14490, Korea
[2]Development Department, GurumNetworks, Inc., Yongin 16964, Korea

요 약

철도의 원활한 운행을 보장하기 위하여 신호, 전철전력, 통신, 설비 등 분야별로 독립된 시스템이 구축되어 운영되고 있으나, 각각의 현장 설비가 센싱 데이터를 중앙관제로 전송하는 매체나 프로토콜이 서로 상이하다. 현재 연구개발을 진행하고 있는 실시간 철도안전 관제시스템은 분산된 안전에 관련된 데이터를 통합하여 위험을 예측하고 철도사고를 예방하기 위한 것으로, 분산되어 있는 안전 관련 데이터를 통합하여 획득하는 것이 필수적이다. 이를 위하여 서로 상이한 매체나 다양한 프로토콜로 전송되는 정보를 기존 어플리케이션의 변경 없이 수용할 수 있는 데이터 변환이 필요하다. 본 연구에서 다양한 방식으로 전송되는 정보를 통합 관리하기 위한 데이터 변환 사례를 조사하고, 기존 방식에 전송기능을 부가하여 일체화한 XML 기반의 프로토콜 변환 방안을 제시하고, 이를 구현하고자 한다.

ABSTRACT

For the safety of train operation, monitoring & supervisory systems for train, signal, power, communication and facilities is operating independently in another place, so, its sensors are interdependently connected from each other to transfer gathering datas of sensing to control center. A Goal of Real-time railway safety supervision system is to improve the safety oversight efficiency and to prevent accidents by means of hazard prediction based on big data by integrating all of safety sensing data in wayside of railway, and the System is requested acquisition of all of sensing data of safety. So, we need special method of protocol converting for the purpose of integrating all of detecting data concerning safety without any changing application. In this paper we investigate the existing converting method in communication field, and propose a new progress to converting protocol adding function of transfer using XML file, and implemented this algorithm, and tested with example packets, finally.

키워드 : 실시간, 철도안전관제, 프로토콜 변환, 확장형, 마크업언어

Key word : Realtime, Railway Safety Supervision, Protocol Converting, Extensible, Markup Language

Received 17 March 2016, **Revised** 24 March 2016, **Accepted** 27 April 2016

* **Corresponding Author** Jin Ahn(E-mail:jinahn@daeati.co.kr, Tel:+82-32-680-0873)
Railway Technical Research Center, DAEATI Co., Ltd, Bucheon 14490, Korea

Open Access **http://dx.doi.org/10.6109/jkiice.2016.20.7.1335** print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

현재 연구를 진행하고 있는 실시간 철도안전 관제시스템은 기존의 관제시스템을 중심으로 차량, 역사 및 시설 등에 대한 모든 안전 관련 정보를 통합하여 실시간으로 감시하고 사고위험을 예측 및 진단하여 제어할 수 있는 것으로, 철도 안전관제시스템 플랫폼 개발과 분산된 안전관련 데이터를 통합하기 위하여 기존의 Client/Server 아키텍처를 Publish/Subscribe 구조로 표준화하는 연구를 동시에 진행하고 있다[1-3].

기존의 철도 감시 설비는 신호, 전철전력, 통신, 설비 등 분야별로 독립된 시스템이 구축되어 있으며, 각각의 현장 설비가 센싱 데이터를 중앙관제로 전송하는 매체나 프로토콜이 서로 상이하다.

이러한 데이터를 통합 관리를 위하여 실시간 철도안전 관제에서는 기존의 애플리케이션을 변경하지 않고 실시간 안전관제 표준 프로토콜로 변환하는 모듈의 구현이 요구된다.

본 연구에서는 기존 감시 센서의 네트워크 구조 변경 없이 데이터를 변환하는 사례를 조사하고, 실시간 철도안전 관제를 위하여 유연성 및 확장성을 갖춘 새로운 데이터 변환 방안을 제시하고 이에 대한 구현 과정을 상세히 설명한다.

Ⅱ. 본 론

2.1. 변환 과정 소개

실시간 철도안전 관제에서의 정보 수집은 기존의 다양한 센서 데이터를 입력받아 표준화된 프로토콜로 변환하는 과정으로, 그림 1과 같이 기존의 프로토콜을 분석하는 과정, 데이터를 변환하는 과정, 새로운 프로토콜로 전송하는 과정으로 구분된다.

프로토콜을 분석하는 과정은 다양한 형태로 전송되는 데이터를 분석해서 변환 과정에 필요한 데이터를 추출하는 과정으로 Protocol Information 정보를 활용해 의미 있는 데이터를 추출하는 과정이다. 예를 들어, 어느 부분이 온도이고, 습도인지를 구분해서 그 값을 추출하는 과정이다.

데이터를 변환하는 과정은 프로토콜 분석 과정에서 추출된 정보를 토대로 표준화된 프로토콜에 맞추어서 데이터를 변환하는 과정이다. 예를 들어, 기존의 프로토콜이 화씨를 사용하고, 새로운 프로토콜이 섭씨를 사용할 경우 프로토콜 분석 과정을 통해 추출된 화씨 값을 섭씨로 변환하는 과정이 필요하다.

새로운 프로토콜로 전송하는 과정은 데이터를 변환하는 과정을 통해 수합된 데이터를 모아 새로운 프로토콜 형태로 포맷을 맞추어 전송하는 과정이다. 예를 들어 온도와 습도 값이 추출 되었고, 새로운 프로토콜이 특별한 형태의 UDP 패킷 형태라면 추출된 값을 UDP 패킷 형태로 변환하여 전송하는 과정이다.

기존의 철도 안전 감시를 위한 센서 네트워크는 수백 개의 센서가 서로 다른 프로토콜을 사용하여 연결되기 때문에 기존의 프로토콜을 분석하고 표준 프로토콜로 변경하는 데이터 변환 과정이 필수적이며, 기존의 어플리케이션을 변경하지 않고 다양한 정보를 분석하고 표준 프로토콜로 변경하기 위한 데이터 변환 방안 연구가 요구된다.

2.1.1. 사례조사

다양한 종류의 데이터를 수집하는 기존 솔루션에서 기존의 어플리케이션을 변경하지 않고 수집된 다양한 종류의 센서 데이터를 해석하기 위한 방법으로 범용적인 메타-프로토콜 방식을 주로 사용한다. 이러한 패킷의 프로토콜을 해석하기 위한 특허나 연구에 대한 조사 내용은 다음과 같다.

국내 특허로서는 「프로토콜 분석을 위한 패킷 분석 장치 및 방법」[4]이 있으며, 메타 파일 활용 사례로는 프로토콜 분석용인 libPDL[5], 통신 분야의 스펙 정의 언어인 SDL[6], 네트워크 프로토콜 정의용 언어인

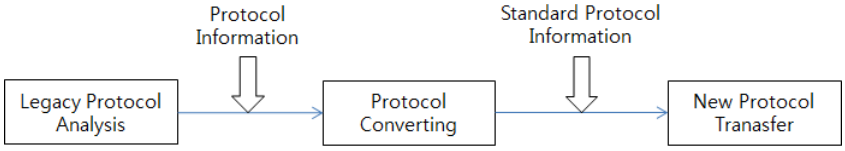

Fig. 1 Process of Protocol Converting to Standard Protocol

DSLs[7], 장애 복구를 위한 프로토콜 정의 언어인 DIL[8] 등이 있으며, 주요 기능에 대한 비교는 표 1과 같다.

메타 파일의 표현 방법(Describe Method)은 Script 또는 XML 형태가 있으며, 작동레벨(Operation Level)은 대부분 세부 프로토콜 수준까지 정의하고 있으며, 지원 요소(Reference Implementation)는 수준의 차이는 있지만 대부분 참조 가능한 소스를 공개하고 있다.

Table. 1 Comparison of Existing Converting Method

Items	Describe Method	Operation Level	Reference Implementations
libPDL	Script Type	Protocol Level	C++ Source Support
SDL	Modeling Type	Description Level	No or Not Opened
DSLs	Script Type	Protocol Level	No or Not Opened
DIL	Script Type	Protocol Level	C++ Source Support
NetPDL	XML Type	Protocol Level	C++ Source Support

2.1.2. 검토 결과

기존 감시 시스템의 다양한 센서를 통합하기 위하여 메타-프로토콜을 이용하는 것은 필수적으로 판단되며, 다양한 사례를 분석한 결과 XML을 기반으로 하는 프로토콜 정의 언어인 NetPDL[9]이 범용성이 높고, 직관적이며, 유연성과 확장성이 높은 것으로 판단되어 NetPDL의 스펙을 차용하여 실시간 안전관제를 위한 데이터 변환 엔진에 적용하고자 한다.

2.2. 데이터 변환 기능 구현

실시간 철도안전 관제의 데이터 변환을 위하여 유연성과 확장성이 높은 것으로 판단되는 NetPDL의 기본 스펙을 적용하고, 이를 확장하여 다양한 센서의 프로토콜을 해석하고 표준 프로토콜로 변환한 후 재전송하는 XML 기반 메타-프로토콜 방식의 새로운 데이터 변환 과정을 구현하였다.

NetPDL의 확장은 일부 데이터를 변화하기 위한 필드 정의 및 재전송을 위한 함수 정의(assign-variable 태그)를 추가하였으며, 이는 NetPDL의 표준 안에 포함되어있는 기본을 적용하였다.

2.2.1. NetPDL 소개

NetPDL은 XML에 기반을 둔 언어로서 확장 가능하며 직관적인 표현 방법이 특징으로, 상당수의 네트워크 프로토콜이 이미 NetPDL 형식으로 정의가 되어 있고, 간단한 형태의 스크립트(대부분의 경우 사칙 연산이나 간단한 함수의 적용)를 지원하여 데이터 변환 과정에 응용할 수 있는 유연한 구조를 가지고 있다.

```
▼<protocol name="udp" longname="UDP (User Datagram protocol)" showsumtemplate="udp">
 ▼<format>
  ▼<fields>
     <field type="fixed" name="sport" longname="Source port" size="2" showtemplate="FieldDec"/>
     <field type="fixed" name="dport" longname="Destination port" size="2" showtemplate="FieldDec"/>
     <field type="fixed" name="len" longname="Payload length" size="2" showtemplate="FieldDec"/>
     <field type="fixed" name="crc" longname="Checksum" size="2" showtemplate="FieldHex"/>
   </fields>
 </format>
```

Fig. 2 A Part of UDP Protocol Definition NetPDL

그림 2는 NetPDL에서 UDP 프로토콜을 정의하는 메타-프로토콜의 일부인데, UDP 프로토콜의 Source/Destination 포트 번호, UDP 페이로드의 크기, CRC 값이 각각 2 byte 크기인 것을 정의하고 있다. 위와 같이 단순한 형태의 프로토콜은 물론 가변적이고 동적인 프로토콜 또한 융통성 있게 정의할 수 있는 것이 특징이다.

```
▼<switch expr="buf2int(type)">
 ▼<case value="0" maxvalue="1500">
     <nextproto proto="#llc" preferred="true"/>
   </case>
 ▼<case value="0x800">
     <nextproto proto="#ip" preferred="true"/>
   </case>
 ▼<case value="0x806">
     <nextproto proto="#arp" preferred="true"/>
   </case>
 ▼<case value="0x8863">
```

Fig. 3 Example of Simple Function, Variables, Branch

또한, NetPDL은 간단한 형태의 함수와 분기 그리고 변수를 사용할 수 있는 구조를 가지고 있다. 그림 3은 NetPDL로 정의된 Ethernet 프로토콜 정의 중 일부인데, Ethernet의 type을 숫자 형태(Integer)로 변환한 후 그 값에 따라 그 다음 프로토콜(Ethernet 안에 있는 프로토콜의 종류)이 무엇인지 분기하는 구조를 보인다.

NetPDL의 이러한 특성으로 인해 프로토콜을 분석

할 뿐만 아니라 값을 변환하고, 더 나아가 특정 함수를 실행시킴으로써 분석하고 변환한 값을 이용해 특정 기능을 실행시키는 기능을 제공한다.

2.2.2. NetPDL 확장

본 연구에서는 NetPDL의 정의를 확장해 프로토콜을 해석할 뿐만 아니라 프로토콜의 값을 변환하고, 다른 형태의 프로토콜로 전송하는 기능을 추가하여 일체화하도록 구현하였다.

기존 NetPDL은 case와 field와 같이 프로토콜을 해석하는 부분은 정의하고 있지만, expr와 send 함수는 정의하지 않고 있다. 그 이유는 NetPDL 자체가 프로토콜을 해석하기 위한 표준이지 프로토콜을 분석한 후 특정한 기능을 수행하려는 표준은 아니기 때문이다.

본 연구에선 프로토콜을 해석할 뿐만 아니라 해석된 내용을 변환하고, 특정한 기능을 수행(표준화된 프로토콜로 재전송)해야 하기 때문에 NetPDL을 확장해 적용하였다.

```
▼<case value="0x0077" maxvalue="0x0082">
  ▼<field type="fixed" name="TrackBlock" size="1">
     <field type="bit" name="PYF" mask="0x80" size="1"/>
     <field type="bit" name="PY" mask="0x40" size="1"/>
     <field type="bit" name="PR" mask="0x20" size="1"/>
     <field type="bit" name="PRF" mask="0x10" size="1"/>
     <field type="bit" name="CU_TK" mask="0x0F" size="1"/>
   </field>
   <expr>send("TrackBlock", PYF, PY, PR, PRF)</expr>
 </case>
```

Fig. 4 Analysis of the Existing Sensor data, call “send” Function

그림 4는 기존의 센서 데이터를 해석해서 send라는 함수를 호출하는 NetPDL 형식을 확장해 적용한 경우이며, 실제적인 기능을 수행하기 위하여 expr 태그를 추가하고, send라는 함수를 추가함으로써 해석된 값을 재전송할 수 있도록 하였다.

그림 5는 단순히 기존의 프로토콜을 해석하고 그 내용을 재전송하는 것이 아니라, 해석된 데이터 필드의 일부를 변환하는 과정을 NetPDL 표준 안에 포함된 assign-variable 태그를 활용하여 정의한 것으로, SG와 SF는 서로 연동되어있는 정보이며, SG가 참일 때 SF 값은 항상 거짓이므로 두 개의 필드를 합하여 하나의 변수로 만드는 과정과, 화씨인 AL 값을 섭씨로 변환해 AL2로 저장하는 과정을 표현하고 있다.

```
▼<case value="0x000F" maxvalue="0x0014">
  ▼<field type="fixed" name="SignalMain" size="1">
     <field type="bit" name="SG" mask="0x80" size="1"/>
     <field type="bit" name="SF" mask="0x40" size="1"/>
     <field type="bit" name="CU_SG_1" mask="0x20" size="1"/>
     <field type="bit" name="TTB" mask="0x10" size="1"/>
     <field type="bit" name="AL" mask="0x08" size="1"/>
     <field type="bit" name="CG" mask="0x04" size="1"/>
     <field type="bit" name="PK" mask="0x02" size="1"/>
     <field type="bit" name="CU_SG_2" mask="0x01" size="1"/>
   </field>
   <assign-variable name="$SGF" value="SG && !SF"/>
   <assign-variable name="$AL2" value="(AL -32) * 1.8"/>
   <expr>send("SignalMain", $SGF, TTB, $AL2, CG, PK)</expr>
 </case>
```

Fig. 5 Analysis of the Existing Sensor data, retransmits after converting the value

2.2.3. 데이터 변환 아키텍처와 소스코드

그림 6은 데이터 변환 기능에 대한 전반적인 아키텍처 구조를 설명하는 것이다.

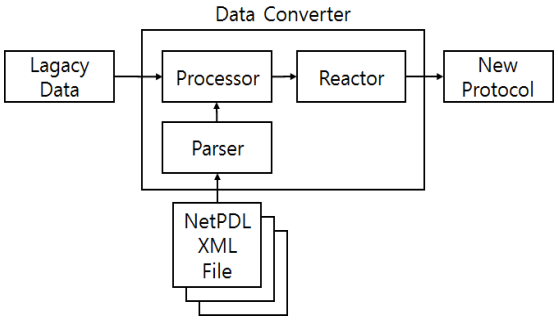

Fig. 6 Architecture of the Data Convert Function

먼저 데이터 변환 기능은 NetPDL로 정의된 XML 파일을 읽어 들여 기존의 데이터를 해석할 수 있는 자료구조인 Processor를 생성한다.

일반적으로 하나의 프로토콜은 하나의 NetPDL로 정의되며, 하나의 NetPDL을 읽어들임으로써 해당하는 프로토콜을 해석할 수 있게 된다.

데이터 변환 기능은 TCP나 UDP로 전송되는 기존의 데이터가 입력될 경우 Processor에서 그 내용을 해석하고 변환한 후 Reactor를 호출한다.

Reactor의 역할은 해석되거나 변환된 정보를 표준화된 또 다른 프로토콜로 재전송하는 역할을 한다. 즉, Reactor가 새로 정의한 프로토콜로 데이터를 전송함으로써 데이터 변환 기능은 기존의 데이터를 새로 정

의한 표준화된 데이터로 전송할 수 있는 기능을 수행한다.

```
void netpdl_process(NetPDL* netpdl, uint8_t* data, int size) {
        if(!netpdl)
                return;

        NPProtocol* protocol;
        if(!(protocol = map_get(netpdl->protocols, "startproto")))
                return;

        if(!data)
                return;

        if(size <= 0)
                return;

        netpdl_init_variables(netpdl);

        NPVariable* netpdl_packet = (NPVariable*)map_get(netpdl->variables, "$packet");
        netpdl_packet->value.buffer = (char*)data;
        netpdl_packet->size = size;

        NPVariable* framelength = (NPVariable*)map_get(netpdl->variables, "$framelength");
        framelength->value.number = size;

        NPVariable* packetlength = (NPVariable*)map_get(netpdl->variables, "$packetlength");
        packetlength->value.number = size;
```

Fig. 7 NetPDL Engine Source Code (part)

그림 7은 NetPDL 엔진(그림 5의 Processor를 실행시키는 엔진)의 소스코드 중 일부인데, UDP 또는 TCP로 전송된 Legacy Data (netpdl_process의 data, size 매개변수)를 Net Protocol로 변환하기 위해 XML을 해석해 C의 구조체 형태로 만든 Processor(netpdl_process의 netpdl 매개변수)를 실행시키는 기능 중 일부이다.

먼저 startproto(NetPDL에서 패킷 해석의 시작 점) 구조체를 가져와 $packet, $framelength, $packetlength라는 변수를 초기화 하는 코드이며, 이후에 좀 더 복잡한 과정은 그림에서 생략 하였다.

2.2.4. 구축 환경

확장된 NetPDL 엔진의 구현은 x86_64를 사용하는 CPU 상에서 구동되는 PC를 사용하였고, 운영체제는 64bit Linux를 사용하였다. 현재 구현된 코드는 64bit Linux에서 구동되지만, 플랫폼에 독립적으로 엔진을 구현하여 다양한 플랫폼에 이식 가능하도록 하였다.

실시간 안전 관제를 위하여 차용한 기존의 NetPDL 엔진은 C 라이브러리 형태로 되어 있어, 다양한 언어로 바인딩하여 사용할 수 있기 때문에 다양한 어플리케이션에 활용이 가능한 장점을 제공한다.

2.3. 구현 결과

그림 8은 데이터 변환 기능을 구현하고 실행시킨 결과이다. 데이터 변환 기능은 본 논문에서 구현한 NetPDL의 기본 엔진을 확장하여 적용하였으며, C 언어로 작성되었고, Linux 상에서 TCP와 UDP 형태의 데이터를 읽을 수 있도록 구현 하였다.

Fig. 8 Implement and Execution of The Data Convert Function

본 예제는 Linux 상에서 일반 어플리케이션을 실행하는 것과 동일한 과정을 거치되, RS-232나 RS-485와 같은 통신은 TCP로 변환한 후 전송되기 때문에 TCP와 UDP를 입력 받으면 철도의 센서 네트워크의 대부분의 기존 센서 데이터를 해석할 수 있게 된다.

기존의 센서 데이터로부터 스니핑 된 데이터가 입력되면 NetPDL로 정의된 프로토콜을 해석한 후, 변환해 새로운 프로토콜(본 연구에선 DDS, Data Distribution Service)로 변환하여 재전송하는 기능을 구현하였으며, 그림 8은 수행 과정 중 어떤 패킷을 어떤 데이터로 해석하였으며, 새로운 패킷으로 전송하기 위한 변환 결과를 보여주는 메시지의 일부이다.

Ⅲ. 결 론

실시간 철도안전 관제시스템은 철도의 안전에 관련된 정보를 종합적으로 관리하는 시스템이 없어 철도 관제시스템을 중심으로 차량, 역사 및 시설 등에 대한 안전 관련 정보를 실시간 감시하고 사고위험을 예측 및

진단하여 제어할 수 있는 것으로, 분야별로 분산된 시스템의 안전 검지 정보를 다양한 센서로부터 제조사별로 제각각 사용하고 있는 통신 방법과 프로토콜을 통해 수집되어야 한다.

그렇기 때문에 기존의 센서 네트워크에 새로운 센서를 추가/제거 하거나 새로운 안전 관제를 위한 어플리케이션을 그대로 적용하는 것이 쉽지 않다.

따라서 본 논문에선 기존의 시스템의 변경이나 추가적인 어플리케이션을 구현하는 것이 없이 수집된 데이터를 표준화된 프로토콜로 전송하는 방안을 연구하고자 표준 프로토콜로의 데이터 변환 기능은 다양한 형태의 프로토콜에 하나의 엔진으로 대응하기 위해 프로토콜을 XML로 표현하는 메타-프로토콜을 검토하였으며, 그 스펙은 NetPDL을 활용하여 새로운 프로토콜로 재전송 하는 기능을 부가하여 일체화하기 위하여 NetPDL의 일부 속성을 확대 적용하였다.

본 연구에서 구현한 데이터 변환 기능은 하나의 엔진으로 다양한 형태의 프로토콜을 해석하고, 표준화된 형태의 새로운 프로토콜로 재전송할 수 있는 유연한 구조를 가지고 있는 엔진으로, 과제에서 추진되는 실시간 안전 관제를 위한 통합 안전검지 정보 수집에 유용할 것으로 판단한다.

ACKNOWLEDGEMENTS

This research was supported by a grant (16RTRP-B082515-03) from Railroad Technology Research Program (RTRP) funded by Ministry of Land, Infrastructure and Transport of Korean government.

REFERENCES

[1] Development of Real-time Integrated Railway Safety Monitoring & Control System Project [Internet] Available : www.kaia.re.kr/portal/landmark/readTskView.do?menuNo=200060&tskId=82515&yearCnt=1

[2] KRRI, “R&D Report of Real-time Integrated Railway Monitoring & Control System,” The Korea Railroad Research Institute: Korea, 2015.

[3] K. H. Shin, T. H. Um, D. S. Lim and J. Ahn, “A Study of the Data Interfacing Device Architecture of the Safety Supervision System for Protocol Standardization and Interoperability,” *Journal of the Korean Society for Railway*, vol. 205, no. 10, pp. 989-994, Oct. 2015.

[4] J-H Jung and H-S Choi. *Packet Analysis Apparatus and Method for Protocol Analysis*, South Korea Patent 10-1466017 to ADD, Korean Intellectual Property Office: Daejeon, 2014.

[5] libPDL, Protocol Definition Language [Internet] Available : http://nmedit.sourceforge.net/subprojects/libpdl.html

[6] F. Belina, D. Hogrefe and A. Sarma, “*SDL with APPLICATIONS from PROTOCOL specification,*” New Jersey: NJ, Prentice Hall, 1991.

[7] S. Bhatti, E. Brady, K. Hammond and James McKinna “Domain Specific Languages (DSLs) for Network Protocols(Position Paper),” *Distributed Computing Systems Workshops, ICDCS Workshops '09. 29th IEEE International Conference*, P-ISBN 978-0-7695-3660-6, pp. 208-213, Jun. 2009.

[8] D. Sturman and G. Agha “A protocol description language for customizing failure semantics,” *IEEE Reliable Distributed Systems*, P-ISBN 0-8186-6575-0, pp. 148-157, Oct 1994.

[9] F. Risso and M. Baldi, “NetPDL:An extensible XML-based language for packet header description,” *Computer Networks*, Vol. 50, no. 5, pp. 668-706, Apr. 2006.

안진(Jin Ahn)

2003 ~ 현재 대아티아이(주) 철도기술연구소
2008.3 ~ 2010.2 서울과학기술대학교 철도전기신호공학 석사
※관심분야 : 철도관제, 안전관제, IoT 솔루션, 웹 솔루션, XML

김성민(Seong-Min Kim)

2014.2 ~ 현재 (주)구름네트웍스 대표이사
2009.11 ~ 2012.1 한국전자통신연구원 연구원
2007.4 ~ 2008.9 한컴시큐어(주) 과장
2005.9 ~ 2007.3 한국국방연구원 전문연구원
2003.9 ~ 2005.8 건국대학교 컴퓨터공학 석사