사물인터넷 기기 간 데이터 교환을 위한 경량 데이터 표현 문법 구조

권동한[O1], 신준섭[2], 임도현[2]
[1]하나고등학교, [2]한국과학영재학교
kznm.develop@gmail.com, 22-065@ksa.hs.kr, shiueo.csh@gmail.com

Lightweight Syntax Structure for Describing Data to Transfer over Internet of Things (IoT) Devices

Ryan Donghan Kwon[O1], JunSeob Shin[2], Do Hyun Lim[2]
[1]Hana Academy Seoul, [2]Korea Science Academy of KAIST

요 약

이기종 기기 간의 통신이 이루어지는 사물인터넷 (IoT) 환경에서 데이터 교환을 위하여 내용과 구조를 정의하는 규격이 필요하게 되었다. 기존 ASN.1 표준은 범용성과 데이터 구조만의 표현에 초점이 맞추어져 있고 현대 환경에 도입하기 용이하지 않으며 직관적이지 않은 등, 현대 IoT 기기 간 통신에 사용되기 적합하지 않은 부분이 존재한다. 이에 보다 경량적인 데이터 표현 문법 구조 Lightweight Syntax Structure for Describing Data 를 정의 및 제안하고, 기기간 데이터 교환 시나리오에 적용하여 경량성과 신속성 등을 시험했다.

1. 서 론

사물 인터넷 (Internet of Things, IoT)은 다양한 임베디드 시스템에 센서와 통신 기술을 내장하여 각종 사물을 연결하는 기술로, 현대 사회에 있어 필수적인 도구이며 많은 공학적 시도를 통해 발전해오고 있다. 시장 조사기관 IDC는 사물 인터넷 시장의 규모는 2022년 1조 달러를 돌파하고, 2023년 1조 100억 달러까지 성장한다고 전망했다. 이러한 시대적 흐름에 따라 사물 인터넷의 트래픽 또한 2022년까지 4.8 ZB로 성장[1]해온 바 있다.

이러한 성장세에도 불구하고, 사물인터넷 기기 간의 프로토콜은 원시적인 형태를 띠고 서로 다른 상태로 포진하고 있어 이기종 IoT 기기 사이의 데이터 교환에 효과적이지 못했다. TCP/IP 등의 패킷 수준 데이터 교환 프로토콜 또는 ZigBee와 같은 통신 방식에 대한 연구가 주로 진행되어왔고, 데이터의 내용과 구조를 표현하는 방법에 대해서는 의미 있는 발전이 이루어지지 않았다.

소프트웨어 자체적으로 통신에 필요한 프로토콜을 정의하여 사용하는 경우가 현재로서는 가장 일반적인 방법이나, IoT는 용도를 불문하고 다양한 도메인에서 활용되는 기기들과 해당 기기간의 통신이 이루어지기 때문에 데이터 자체와 구조를 정의하는 규격이 필요하게 되었다. 기존의 ASN.1 등의 표준은 범용성과 데이터 구조의 표현에 초점[2]이 맞추어져 있고 직관적이지 않은 등 실사 시의 단점이 있어 현대 IoT 통신에 사용되기 적절하지 않은 부분이 있다.

이에 본 논문은 이기종 IoT 기기간 통신을 위하여 데이터의 내용과 구조를 효과적으로 표현하고 교환할 수 있는 규격인 Lightweight Syntax Structure for Describing Data (이하 LSD)를 정의 및 제안하고, IoT 기기간의 통신을 모사하여 검증토록 한다.

2. 이기종 **IoT** 기기간의 통신

KISA의 IoT 보안인증 제도에 따르면 IoT 기기는 3가지 범주로 구분할 수 있다. 펌웨어 기반의 센서 등 소형제품은 Lite 등급 기기, 저사양 OS를 탑재재한 중소형제품은 Basic 등급 기기, 중대형 스마트가전제품은 Standard 등급 기기로 구분되고 본 논문에서는 각각 1, 2, 3등급 IoT 기기라 한다.

1등급 IoT 기기는 일반적으로 초소형·초경량·초절전 기기로서 센서의 기능을 주로 수행한다. 메모리 및 프로세싱에 있어 상당한 제약이 있다. 2등급 기기는 스마트 도어락, 웨어러블 밴드 등 일반적으로 8~16비트 프로세서를 사용한다. 3등급 기기는 32비트 이상의 프로세서로 작동되는 게이트웨이, 스마트폰 등의 기기를 예시로 들 수 있다.

IoT 기기는 전통적인 컴퓨팅 디바이스와 플랫폼 측면에서 큰 차이를 보인다. 32비트 또는 64비트 프로세서의 경우 통상적인 통신을 이루고 데이터를 처리하는 것에 있어 자료형 호환 등의 문제가 발생하는 경우가 극히 드물다. 하지만 1, 2등급 IoT 기기의 경우 long int 등의 범용적으로 사용되는 자료형을 지원하지 않는

경우도 발생하고, 데이터 교환 시 이기종 플랫폼에서 추론하는 데이터 타입이 상이하여 오버플로 등의 오류를 발생시키기도 한다.

ASN.1에서는 이러한 점을 고려하기 위해 SEQUENCE 등의 자료형 명시를 통하여 데이터를 구조화하고 저장하도록 한다. 하지만 ASN.1은 컴파일러 등을 통하여 개발 언어에서 자료형을 지정하고 사용해야 하는 등 프로세싱 리소스에 제약이 있는 경우 유연한 사용이 불가능할 수 있다. 또한 데이터의 저장에 초점이 맞추어져 있으며, 유지보수는 지속해서서 이루어지고 있지만 암호화, 생체 인식 등 전반적인 자료의 핸들링을 위하여 정립된 바 있어 현 IoT의 사례에 사용하기 적합하지 않다. 또한 데이터의 구조가 아닌 내용과 맥락 (Context)를 표현하는 것에 있어 가독성이 저하된다는 문제점이 있다. ASN.1을 사용하여 데이터를 저장하는 인터넷 인증서[3]의 예시[표1] 에서 이를 확인할 수 있다.

```
Certificate  ::=  SEQUENCE  {
   tbsCertificate       TBSCertificate,
   signatureAlgorithm  AlgorithmIdentifier,
   signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
   version  [0] EXPLICIT Version DEFAULT v1,
   serialNumber CertificateSerialNumber,
   signature    AlgorithmIdentifier,
   issuer       Name,
   validity     Validity,
   subject      Name,
   (...)  }
```

표1. ASN.1을 통한 자료의 표현 예시

3. 경량 데이터 표현 문법 구조(LSD)의 정의 및 명세

본 논문에서는 위에서 살펴본 현재 IoT 환경에서의 데이터 전송을 개선하고, ASN.1에서 발생하는 가독성 문제를 개선하여 이기종 IoT 기기 간 데이터 내용과 구조를 효율적으로 교환할 수 있도록 하기 위해 경량 데이터 표현 문법 구조(LSD)를 정의, 규격화하여 제안한다.

```
academy: dict = {
    name: TEXT = “Korea Science Academy”;
    foundation: TEXT = “09/06”;
    principal: TEXT = “Final Boat”;
    location: LIST = [INT: 00, INT: 21];
    float_number: REAL = 123.4;
};

developer: TEXT(3) = [
    “Ryan”, “JunSeob”, “Shio”,
];
```

표2. LSD를 사용한 데이터의 표현 예시

3.1. 데이터 타입: 일반 자료형과 재정의 자료형

LSD에서 사용할 수 있는 자료형 (데이터 타입)은 일반 자료형 (General Data Type)과 재정의 자료형 (Wrapped Data Type)으로 구분할 수 있다.

일반 자료형은 기존 범용적으로 사용되는 프로그래밍 언어의 자료형을 기반으로 한 자료형으로, 다음과 같은 종류가 있다.

`null, int8, int16, int32, int64, uint8, uint16, uint32, uint64, float16, float32, float64, bool, char, uchar, dict, list`

재정의 자료형은 일반 자료형을 재정의한 자료형으로, 자동으로 일반 자료형을 추론하거나 일관성을 지키기 위하여 지정된 일반 자료형으로서 사용할 수 있다.

`INTEGER, REAL, TEXT, BLOB, REAL, DICT, LIST`

배열은 두 가지 형태로 사용할 수 있다. 일관된 일반 또는 재정의 자료형을 사용하여 배열을 정의하거나, 일관되지 않은 자료형을 가지는 데이터를 저장하기 위해 `list`를 사용할 수 있다.

`bool`과 `BOOL`의 경우와 같이 이름이 문자로만 이루어진 자료형의 경우 `uppercase`와 `lowercase`의 혼용을 허용하나, `uppercase`의 이름을 가지는 재정의 자료형이 지정 자료형이 아닌 추론 가능한 자료형의 형태를 띈다면 해당 속성이 우선시된다.

3.2. 문법적 구조

데이터의 정의와 대입은 다음과 같은 구조를 따른다.

```
data_name: DATATYPE = data;
```

일관된 데이터 타입을 가지는 배열의 선언은 다음과 같다.

```
list_name: DATATYPE(length) = [...];
```

데이터가 `char`과 같은 문자인 경우 `‘...’`, 문자열의 경우 `“...”`, Dictionary는 `{...}`, List는 `[...]`과 같이 표현하여 데이터의 성질을 명시한다.

각 데이터 객체는 구분자 세미콜론(`;`) 또는 콤마(`,`)로 구분한다. 일반적으로 리스트 내부의 객체를 구분하는 경우에는 콤마, 기타 객체를 구분하는 경우에는 세미콜론을 사용하도록 하되 프로젝트에서 요구되는 일관성을 충족하기 위하여 통용할 수 있다.

3.3. 데이터의 직렬화(Serialization)

모든 데이터는 기본적으로 세미콜론, 콤마 등의 구분자로서 형태가 구분되어지고 (), {}, []를 통해 그 형태가 결정되기 때문에 space, linebreak 등의 indent와 불필요한 구분자를 생략하는것으로 단축을 행할 수 있다. 표2에 주어진 예시를 단축하는것은 다음과 같이 표현된다.

```
academy:dict={name:TEXT=“Korea Science
Academy”;location:LIST=[INT:00,INT:21];floa
t_number:REAL=123.4;};developer:
TEXT(3)=[“Ryan”,“JunSeob”,“Shio”];
```

표3. LSD 데이터의 단축

해당 데이터를 binary로 인코딩하는것으로 이진 직렬화를 행할 수 있다. 표3에 주어진 데이터를 이진 직렬화할 시, 1344bit의 리소스를 점유하게 된다.

3.4. 자료형의 추론

재정의 자료형을 사용하는 경우, 플랫폼에 따른 컴파일러 또는 파서 구현체는 해당 플렛폼에 따른 재정의 자료형이 가지는 일반 자료형의 추론을 행하게 된다. 재정의 자료형의 추론 시 적용될 수 있는 일반 자료형은 다음과 같다.

```
[INTEGER] int8, int16, int32, int64, uint8,
uint16, uint32, uint64

[REAL] float16, float32, float64

[TEXT] char(n)
```

자료형의 추론을 행할 시, INTEGER와 FLOAT의 경우 (플랫폼의 비트 수)/2를 범위로 가지는 signed type을 기본적으로 추론한다. 64비트 프로세서에서, INTEGER는 int32로 추론하게 된다.

데이터의 교환이 이루어지는 이기종 기기의 플랫폼의 프로세싱 성능이 일치하거나 유사한 경우에 재정의 자료형을 사용하는 것을 용도로 한다. 기기 간 프로세싱 파워에 유의미한 차이가 생긴다면 이에 따라 발생할 수 있는 버퍼 오버플로우 이슈 등에 적절히 대응하기 위해 일반 자료형을 사용하는 것이 좋다.

4. 기기간 데이터 교환의 모사

위에서 정의한 LSD의 규격을 기반으로, [4]에서 캔위성으로부터 수신받은 GPS, IMU, 조도, 온습도, 지상 영상 데이터를 교환한다. 캔위성과 지상과의 통신은 Parani-ESD110V2 모듈을 사용하여 구현되었으며 지상 520m 까지 통신이 유지되었다.

```
payload: dict = {
    ALTM: int4(10) = [
    64, 48, 46, 47, 44, 11, 21, 91, 74,
0x74
    ];
    DTLM4: bool(32) = [1,1,1,1,1,1,1,1,0,];
    CAN_TIME: TEXT = "15:28:07";
    RUN_TIME: TEXT = "00:00:09";
    IMU: REAL(3) = [-86.72, -44.99, 36.27];
    image: blob = (...);
};
cansat_detail: TEXT = "KSAT_포병";
```

표4. LSD를 통해 표현된 CANSAT-지상관제 통신 데이터

해당 데이터의 구조를 ASN.1 형식으로 표현할 시, 표5와 같은 형태를 갖추게 된다. 해당 자료는 DER, CER 등 Encoding Rules에 따라 반영되어야 한다.

```
payload ::= SEQUENCE {
  ATLM     SEQUENCE OF INTEGER,
  DTLM4    BOOLEAN,
  CAN_TIME TIME-OF-DAY,
  RUN_TIME TIME-OF-DAY,
  IMU      SEQUENCE OF REAL,
  Image    OCTET STRING
}

Cansat_detail ::= UTF8String

---

data payload ::= {
  ALTM     {
    64, 48, 46, 47, 44, 11, 21, 91, 74,
0x74
    },
  DTLM4    {1,1,1,1,1,1,1,1,0,},
  CAN_TIME "15:28:07",
  RUN_TIME "00:00:09",
  IMU      {-86.72, -44.99, 36.27},
  Image    (...)
}

name Cansat_detail ::= “KSAT_포병”
```

표5. ASN.1를 통해 표현된 CANSAT-지상관제 통신 데이터

지상 관제 어플리케이션과 CANSAT 간 LSD를 통하여 데이터를 교환함으로써 다음과 같은 개선을 이루었다.

- 위성 데이터 자료를 일관성 있게 통신 및 저장한다.

- 속성을 하나의 패킷으로 처리하여 통신 부하를 절감한다.

5. 실사시 **ASN.1** 과 **LSD**의 성능 비교

ASN.1 과 LSD의 성능을 비교하기 위하여 실제 IoT 운영 환경과 비슷 할 것으로 고려되는 형식의 데이터를 처리한 후 그 수행 시간을 비교하였다. 처리 과정을 수행시킨 기기는 Raspberry Pi 4 Model B로, 제원은 표6과 같다.

Raspberry Pi 4 Model B
CPU – Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
MEM – 4GB LPDDR4-3200 SDRAM

표6. IoT Process를 모사한 기기 제원

IoT 기기에서 Python 3 Runtime을 통하여 시간을 측정하였으며 Encode-Decode 쌍을 100회 실행하여 실행 시간(ms)를 도출하였다. 본 논문에서 제안하고자 하는 LSD와 달리, ASN.1은 데이터 구조를 컴파일한 후 모델에 데이터를 삽입/반출하는 특성이 있어 정적 데이터 형태에서의 트랜젝션 처리 속도[표8] 및 동적 데이터 형태에 따른 속도[표9,10]에 대한 측정을 진행하였다.

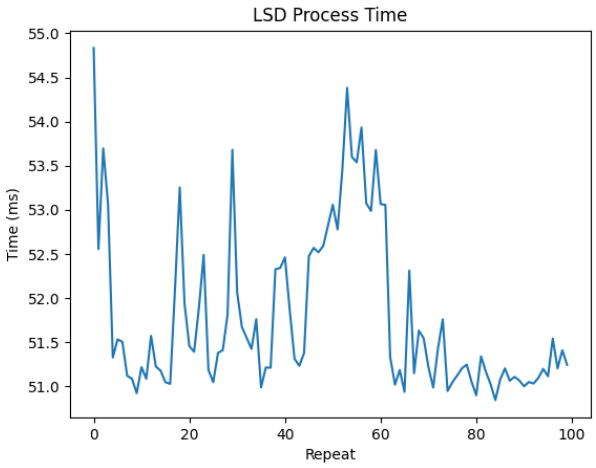

표7. LSD Process TIme

ASN.1 데이터의 반복적인 처리 시간을 분석한 결과, 주기적인 처리시간 급증 이슈가 발생해 표7에서 보여지는 아웃라이어 데이터에 대한 대하여 Threshold를 설정하여 표10과 같이 보정하였다.

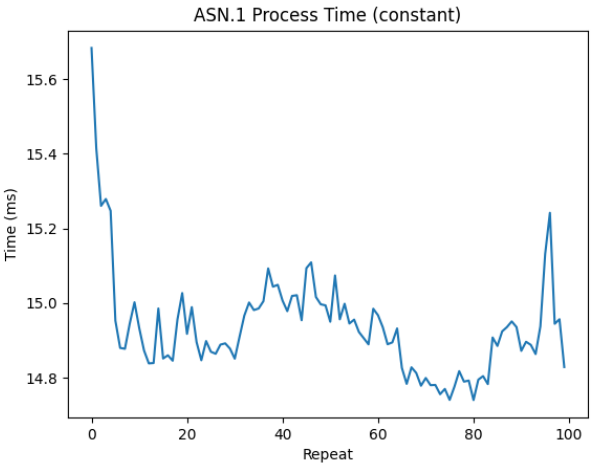

표8. 고정된 데이터 형태에 대한 ASN.1 처리 시간

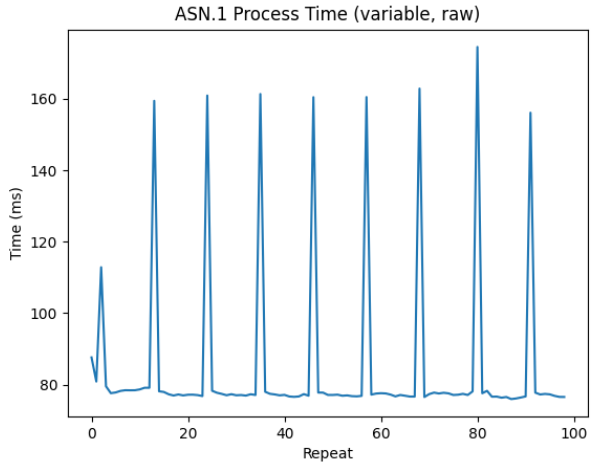

표9. 유동적 데이터 형태에 대한 ASN.1 처리 시간

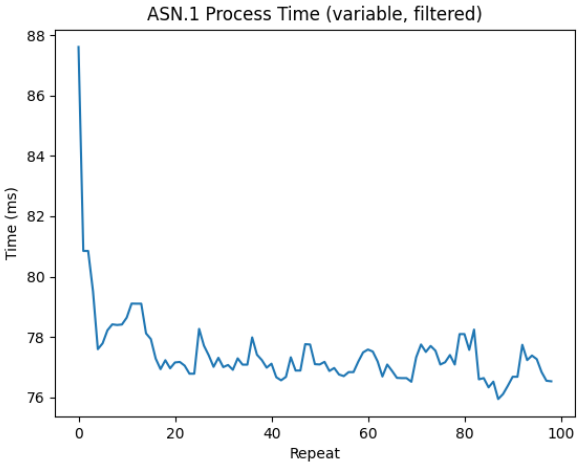

표10. Outliar를 제거한 유동구조 ASN.1 처리시간

ASN 데이터를 분석한 결과 주기적인 처리시간 급증 이슈가 발생해 표7.와 같이 Outliar Manual 보정을 거쳐 비교했다. 비교 결과 LSD는 정적 형태의 데이터

전송에서는 ASN.1에 비해 그 처리 속도가 느렸으나, 동적인 형태의 데이터를 전송할때는 더 우수했다. 이는 동적인 형태의 데이터를 전송할때 LSD는 ASN.1과 비교하여 전처리 작업 시간이 절약되기 때문에 말단에 작용하는 부하 및 전반적인 속도의 개선을 가져올 수 있기 때문으로 고찰된다.

6. 결론 및 제언

LSD는 이기종 IoT 기기 간 통신 환경에서 더욱 체계적인 데이터 교환을 위한 ASN.1을 대체하기 위한 규격으로,

- 플랫폼 간 데이터 고정 자료형의 준수
- 데이터의 일관성 준수 및 가독성의 개선
- 이기종 IoT 통신 시 사용되는 프로토콜의 정립

을 이루고자 한다. 이러한 개선을 끌어내기 위하여 3.1-3.4를 통하여 LSD의 문법을 명세하였으며, [4]의 CANSAT-지상관제 간 통신에 LSD를 적용하여 검증하였다. 이후 Raspberry Pi 4 Model B에서 IoT Data Transaction Process 과정을 모사하여 LSD와 ASN.1의 처리 속도를 비교하며 유동 데이터 처리에서의 성능 향상을 확인하였다.

본 규격을 통하여 이기종 IoT 기기 간의 효과적인 통신을 구성할 수 있었으며, 앞으로 IoT 시장이 더 성장하는 만큼 경량화를 통하여 경제적, 공학적 반사 이익을 취할 수 있을 것으로 관측된다.

7. 참고문헌

[1] Rick Villars, Andrea Siviero, Mary Johnston Turner, Nancy Gohring. “Worldwide Monthly Technology Investment Monitor: December 2022.” IDC, 2022.

[2] Barry, P. T. "Abstract syntax notation-one (ASN. 1)." IEE Tutorial Colloquium on Formal Methods and Notations Applicable to Telecommunications. IET, 1992.

[3] Housley, Russell, et al. “Internet X. 509 public key infrastructure certificate and CRL profile.” No. rfc2459. 1999.

[4] Jun Seob Shin, Ji Hoo Gwak, Ha Neul Park, Ryan Donghan Kwon. “Implementation of CANSAT for gathering information to assist artillery fire.” KSAS, 2022.