Canopen over EtherCAT 프로토콜 분석 도구 개발

윤승희 · 이효림 · 최국철 · 이창홍 · 김동현 · 김종덕*

부산대학교

Developing a Analysis Tool of Canopen Over EtherCat Protocol

Seung-Hui Youn · Hyo-Rim Lee · Guk-Choel Choi · Chang-Hong Lee ·

Dong-Hyun Kim · Jong-Deok Kim*

Pusan National University

E-mail : 0chameleon@pusan.ac.kr / kimjd@pusan.ac.kr

요 약

Canopen over Ethercat은 산업현장에서 EtherCAT 기반으로 동작하는 Canopen 프로토콜이다. 패킷 스니핑을 통하여 구축된 CoE 시스템의 성능을 분석하기 위해서 Canopen 프로토콜에서 사용하는 Process Data Object의 구성과 그 값의 변화를 분석하는 과정은 필수적이다. 하지만 Canopen의 Data Object는 네트워크를 구성하는 장치에 의존적이기 때문에 wireshark와 같은 패킷 분석 프로그램으로 분석하는데 한계가 있다. 따라서 본 연구에서는 CoE 프로토콜을 분석하여 시스템 구성과 Process Data Object를 유추하는 프로그램을 설계하고 구현하였다.

ABSTRACT

Canopen over EtherCAT(CoE) is a Canopen protocol that operates based on EtherCAT in industrial sites. In order to analyze a CoE network system and a performance through packet sniffing and reversing, it is necessary to know Data Objects structure and changes of its value. However, since Data Objects in Canopen is dependent on the devices, there is a limitation by using an existing packet analysis program like a Wireshark. Therefore, we designed and developed a system that infers Data Objects structure and system configuration.

키워드

Canopen, EtherCAT, CoE, Industiral network system, PDO

I. 서 론

CoE(CANopen over EtherCAT)는 EtherCAT을 데이터링크 계층으로 CANopen을 어플리케이션 계층으로 사용하는 네트워크 시스템이다. CoE 시스템은 제어 명령을 생성 및 전달하는 하나의 마스터와 하나 이상의 ESC(EtherCAT Slave Controller)가 내장된 슬레이브 장치로 구성된다. 일반적으로 산업현장의 시스템 개발자는 슬레이브에 해당되는 제품을 구매한 후 이를 제어하는 EtherCAT 마스터를 개발하여 CoE를 구성한다. 마스터 프로그램을 개발하기 위한 IgH, TwinCAT과 같은 프로그램은 이미 존재한다.

* corresponding author

기존 CoE 시스템에서 새로운 슬레이브를 추가하기 위해서 시스템 개발자는 마스터 프로그램에 접근할 수 있어야 한다. 만약 이것이 어렵다면 시스템 개발자는 패킷 분석을 바탕으로 해당 시스템의 동작과 구성을 분석해야 할 것이다. 그러나 이를 위한 프로그램은 많지 않다. 특히, 일반적으로 패킷 분석에 사용되는 패킷 분석 프로그램인 WireShark를 사용할 수 있지만, 이것은 CANopen 분석을 지원하지 않는다. 이는 CANopen 프로토콜에서 사용하는 Object Dictionary 정보가 슬레이브에 의존적이기 때문이다. 따라서 Wireshark에서는 CANopen 프로토콜에 대한 일반적인 분석 알고리즘을 제시할 수 없는 상황이다.

Wireshark는 이러한 경우를 대비하여 확장 플

러그인 개발 API를 제공한다[1]. 하지만 플러그인 개발을 위해서는 시스템의 구성과 설정값을 분석해야 하는데, 이 과정은 EtherCAT과 CANopen 프로토콜에 대한 내부적인 이해를 요구하기에 난이도가 높다. 따라서 본 논문에서는 CoE 시스템에서 패킷을 분석하여 시스템 구성을 확인하고 이에 적합한 PDO를 유추하여 Wireshark 플러그인을 생성하는 시스템을 제안한다. 제안하는 시스템은 슬레이브 장치에서 제공하는 ESI 파일과 슬레이브가 Init, Pre-Operation 상태일 때 전달되는 정보를 바탕으로 동작한다. 이를 설명하기 위하여 아래의 2장과 3장에서는 CoE시스템과 PDO의 동작을 상세히 설명하고, 4장에서는 이를 바탕으로 구체적인 시스템을 제한한다. 5장에서는 제안한 시스템의 구현과 6장에서 결론으로 논문을 마무리하겠다.

II. CANopen over EtherCAT 개요

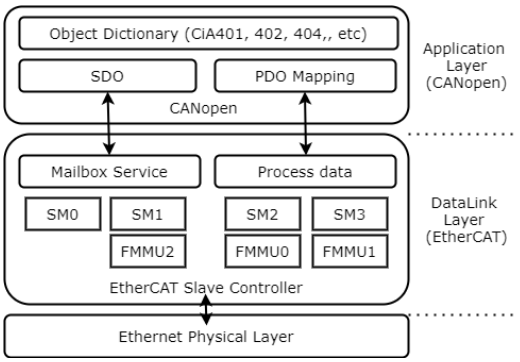

그림 1. CoE 슬레이브 전체 구조 (FMMU와 SyncManager구성은 ESC칩의 구성에 따라 달라질 수 있음)

CoE의 구조는 그림 1과 같다. CoE의 EtherCAT과 CANopen을 간략히 설명하면 다음과 같다.

- **EtherCAT**

EtherCAT은 기존의 필드버스 시스템을 보다 범용적이며 넓은 대역폭을 가진 Ethernet기반으로 개발되었다[2]. 독일의 BeckHoff사에 의해서 개발되었으며, 2003년 ETG(EtherCAT Technology Group)를 통하여 대중에게 공개되었다. EtherCAT의 장점은 다양한 상위 프로토콜을 지원할 수 있고 유연한 토폴로지를 구성할 수 있다는 점이다.

EtherCAT 시스템은 오직 마스터만이 EtherCAT 프레임을 생성하고 송신할 수 있다. 슬레이브는 ESC칩을 이용하여 데이터를 수신하고 on-the-fly 방식으로 처리한다. on-the-fly는 ESC칩 내부의 FMMU(Fieldbus Memory Management Unit), SyncManager 등의 장치를 통하여 데이터 프레임을 수 ns(nano second) 이내로 처리하는 방식이다[3].

ESC는 상태 머신을 바탕으로 동작한다. ESC의 상태 머신은 Init, Pre-Op (Pre-Operation), Safe-Op (Safe-Operation), Op(Operation)등의 상태를 가진다. 마스터는 슬레이브의 상태를 주기적으로 확인하고, 슬레이브의 상태에 맞춰 특정 데이터를 생성 및 송신한다.

- **CANopen**

CANopen은 1994년 CiA(CAN in Automation)에 의하여 개발되었다. CANopen은 OD(Object Dictionary)라는 참조테이블을 기반으로 동작하는 프로토콜로, OD를 어떻게 구성할지는 상위 어플리케이션에 따라 달라진다. OD를 구성하는 기본적인 표준은 CiA301이다. 이 외에도 I/O 모듈을 위한 표준인 CiA401, 드라이브와 모션 컨트롤을 위한 표준인 CiA 402등도 존재한다[4].

CANopen의 서비스는 SDO(Service Data Object)와 PDO(Process Data Object)등으로 구분된다. SDO는 매번 Data Object를 지정하여 해당 Object에 접근하는 프로토콜로, 주로 초기화 과정에 사용된다. PDO는 미리 슬레이브에서 정의한 Data Object 참조 테이블을 바탕으로 테이블에 포함된 Object에만 접근하는 프로토콜이다. PDO는 주로 주기적인 접근이 일어나는 데이터에 사용된다. PDO가 접근하는 데이터는 TxPDO(Transmit PDO)와 RxPDO (Receive PDO)로 구분된다. TxPDO는 마스터에서 업데이트하는 Object들이며, RxPDO는 어플리케이션에서 업데이트하는 Object들이다.

III. CoE PDO의 네트워크 스택 분석

PDO의 참조테이블을 유추하고 패킷을 분석하기 위해서는 각 네트워크 스택에서 어떤 값들의 설정이 필요한지를 확인할 필요가 있다. 따라서 해당 장에서 관련된 내용을 정리한다.

- **EtherCAT 레벨 스택**

EtherCAT이 상위 계층로 CANopen을 지원하기 위해서 마스터 프로그램은 슬레이브칩의 FMMU, SyncManager 장치를 적절히 설정해야 한다. FMMU는 LRW(Logical Read Write) 데이터그램에 담겨서 데이터 일부를 슬레이브의 특정 주소로 맵핑하는 장치이다. SyncManager는 마스터와 슬레이브의 어플리케이션에서 동시에 접근이 발생하는 메모리의 데이터 일관성을 유지해주는 장치이다.

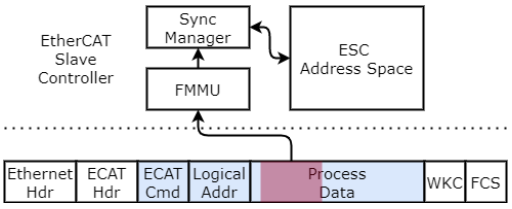

그림 2. ESC 내부의 LRW 명령어의 데이터 처리 과정

마스터 프로그램은 슬레이브가 Pre-Op 상태일 때 FMMU와 SyncManager 설정을 초기화한다. PDO 프로토콜을 지원하기 위해서 마스터 프로그램의 FMMU는 데이터를 SyncManager의 시작주소로 맵핑한다. 그러면 그림 2와 같이 SyncManager는 FMMU로부터 데이터를 받아서 그 데이터의 일관성을 관리하는 역할을 한다[5][6].

- CANopen 레벨 스택

SyncManager가 관리하는 데이터는 Process data 채널을 통하여 상위 CANopen 어플리케이션으로 전달된다. 해당 데이터를 어떻게 해석할지는 미리 정의된 PDO용 참조 테이블인 OD가 존재한다. RxPDO의 OD는 0x1600부터 0x17FF까지의 주소에 존재하고, TxPDO의 OD는 0x1A00부터 0x1BFF까지의 주소에 존재한다. 일반적으로 OD의 몇번째 index를 참조할지는 0x1C1n의 주소를 가진 Sync Manager Assign Object에 정의되어 있다[7]. 그림 3은 PDO의 Object 참조 과정이다. PDO_1은 0x1C12의 SyncManager Assign Object를 조회한다. 여기서 얻은 PDO Dictionary Table 주소를 바탕으로 관련 Object의 구성을 자세히 확인하게 된다.

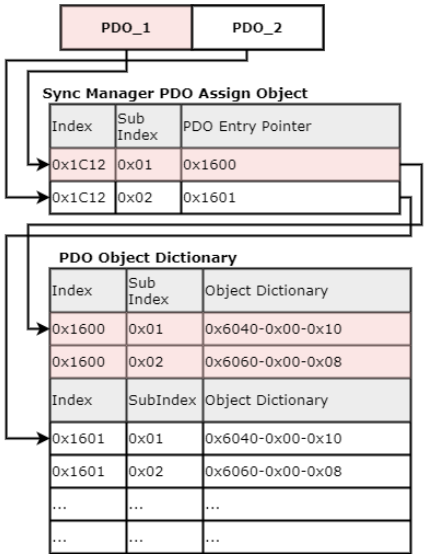

그림 3. PDO의 Object 참조 과정

OD 정보는 CANopen 레벨에서 관리되는 정보들로, 슬레이브에서 제공하는 ESI 파일에서 확인할 수 있다. 마스터 프로그램은 ESI 파일 정보를 바탕으로 슬레이브가 Pre-Op일 때 Sync Manager Assign Object를 설정한다.

Ⅳ. CoE 분석 프로그램 설계

CoE 패킷 분석 프로그램을 만들기 위해 필요한 정보를 정리하면 표1과 같다. 시스템을 구성하기 위해 필요한 파일은 패킷을 캡쳐한 pcap파일과 SI 슬레이브 정보를 담고 있는 ESI 파일이다. 이 때의 pcap 파일은 반드시 Pre-Op 상태의 설정 정보가 담긴 패킷을 포함하고 있어야한다.

표 1. 패킷 분석 프로그램에 필요한 정보

필요한 정보	추출 방법	추출 파일
시스템 구성 정보	Init 상태에서 초반 패킷 정보	패킷캡쳐 (.pcap)
FMMU 설정	Pre-Op 상태에서 0x600 주소에 접근하는 datagram 정보	패킷캡쳐 (.pcap)
Sync Manager Assign Object 설정	Pre-Op 상태에서 0x1c1n주소에 접근하는 SDO datagram 정보	패킷캡쳐 (.pcap)
	<Sm>, <RxPDO>, <TxPDO> 태그 정보	ESI (.xml)
Dictionary Object 정보	<RxPDO> , <TxPDO> 태그 정보	ESI (.xml)

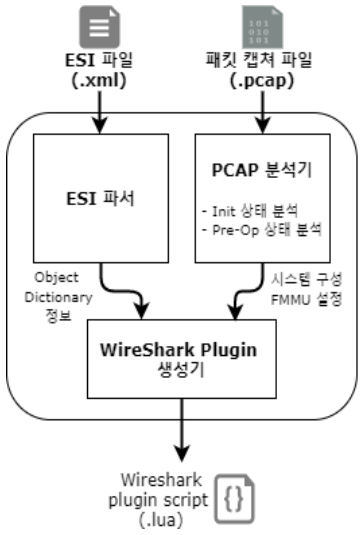

그림 4. PDO CoE 분석 프로그램 구조

추출한 정보를 바탕으로 WirsShark 플러그인을 자동으로 생성해주는 전체 시스템 구조는 그림4와 같다. PCAP 분석기와 ESI 파서가 표1에서 제시한 정보들을 추출한다.

이 정보들은 WireShark 플러그인 생성기로 되고, WireShark 플러그인 생성기가 자동으로 플러그인을 생성한다. Wireshark는 C언어와 Lua기반으로 플러그인 개발을 지원한다. C기반의 플러그인을 생성하는 경우 이를 이용하기 위해 사용자가 Wireshark 소스코드를 직접 컴파일 해야하는 불편함이 있다. 따라서 본 시스템에서는 Lua 기반의 플러그인을 자동으로 생성한다.

Ⅴ. CoE 분석 프로그램 구현

ESI 파서는 C++의 pugixml을 이용하여 개발하였다. PCAP 분석기는 각 계층별로 프로토콜을 클래스로 구현하여 이를 바탕으로 .pcap 파일을 적절히 해석하였다. WireShark Plugin 생성기의 경우, EtherCAT LRW 데이터그램을 해석하는 Lua 코드를 기본 템플릿으로 두고 PDO 참조 테이블의 필드명과 데이터 타입에 따라서 관련 코드만 수정해주는 방식으로 구현하였다.

이렇게 생성한 WireShark 플러그인을 적용한 결과는 아래 그림 5, 6과 같다. 그림 5는 기존 WireSahrk의 패킷 분석 화면이라 CANopen 프로토콜이 해석되지 않고 있다. 그림 6은 플러그인을 추가한 모습으로 PDO의 구조를 확인할 수 있다.

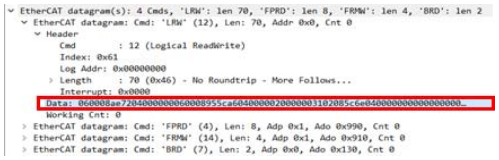

그림 5. 기존 WireShark의 패킷 분석 화면

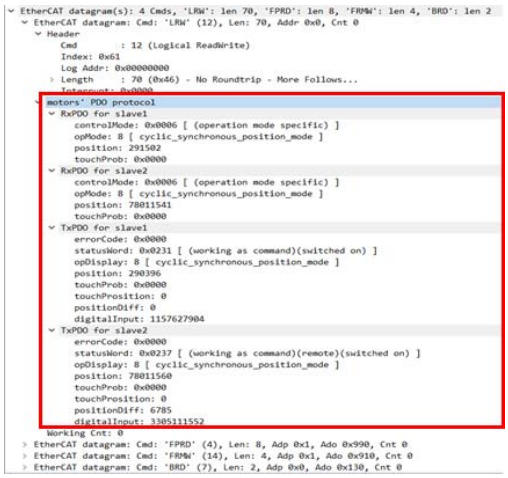

그림 6. 플러그인 추가 후의 WireShark의 패킷 분석 화면

Ⅵ. 결론 및 향후 연구 계획

CoE 네트워크 시스템의 프로토콜을 바탕으로 시스템을 분석하는 시스템을 설계하고 설계된 시스템을 구현하였다. 이러한 시스템의 개발로 CoE 네트워크 시스템의 유지보수의 편의성과 접근성을 높일 수 있을 것이라 기대된다.

현재 논문에서 제시한 시스템은 제일 먼저 관측되는 Pre-Op 상태를 바탕으로 전체 시스템을 유추한다. 하지만 시스템의 설정에 따라서 여러 번의 Pre-Op 상태를 거칠 수 있다. 따라서 설정이 시스템 운영 도중에 변경되는 경우, 현재 개발된 시스템은 제대로 대응하지 못한다. 이에 동적으로 대응할 수 있는 방법을 추가 연구할 필요가 있다.

Acknowledgement

“본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음”(2016-0-00019)

References

[1] Wireshark. Wireshark’s Lua API Reference Manual [Internet]. Available : https://www.wireshark.org/docs/wsdg_html_chunked/wsluarm_modules.html

[2] K. Langlois, Tom van der Hoeven, and D. Rodriguez, “EtherCAT Tutorial, an introduction for real-time hardware communication on Windows”, IEEE Robotics & Automation Magazinem Vol.25 pp. 12 - 122, March 2018

[3] ETG. EtherCAT Slave Implementation Guide ETG.2200. V.3.1.0 [Internet] : https://www.ethercat.org/en/downloads/downloads_7BA2567EB9F443219AD0014448F674F2.htm

[4] CiA. CANopen history [Internet] : https://www.can-cia.org/can-knowledge/canopen/canopen-history/

[5] stun Automation Technology Co., Ltd. EtherCAT User’s Manual V1.06 [Internet] : https://www.estuneurope.eu/wp-content/uploads/download/Manuali/Fielbus/EtherCAT-User-s-Manual-V1-06.pdf

[7] Fastech Co., Ltd. Closed Loop Stepping System EtherCAT Network. Ezi-SERVOII EtherCAT ALL [Internet] : http://fastech.co.kr/pdf/manual/en/ 170615_Manual_Ezi-SERVOII-EtherCAT_ENG.pdf

[8] ETG. EtherCAT Slave Information Specification ETG.2000S (R) V1.0.6 : https://www.ethercat.org/en/downloads/downloads_6A46D45EA33C47ECB2BB2686BBA963EC.htm