SPIN을 이용한 차량용 MOST Network Service 프로토콜 스택 정형검증

Formal Verification of Protocol Stack for MOST Network Service using SPIN

이동아, 윤상현, 이무열, 진현욱, 유준범

목차

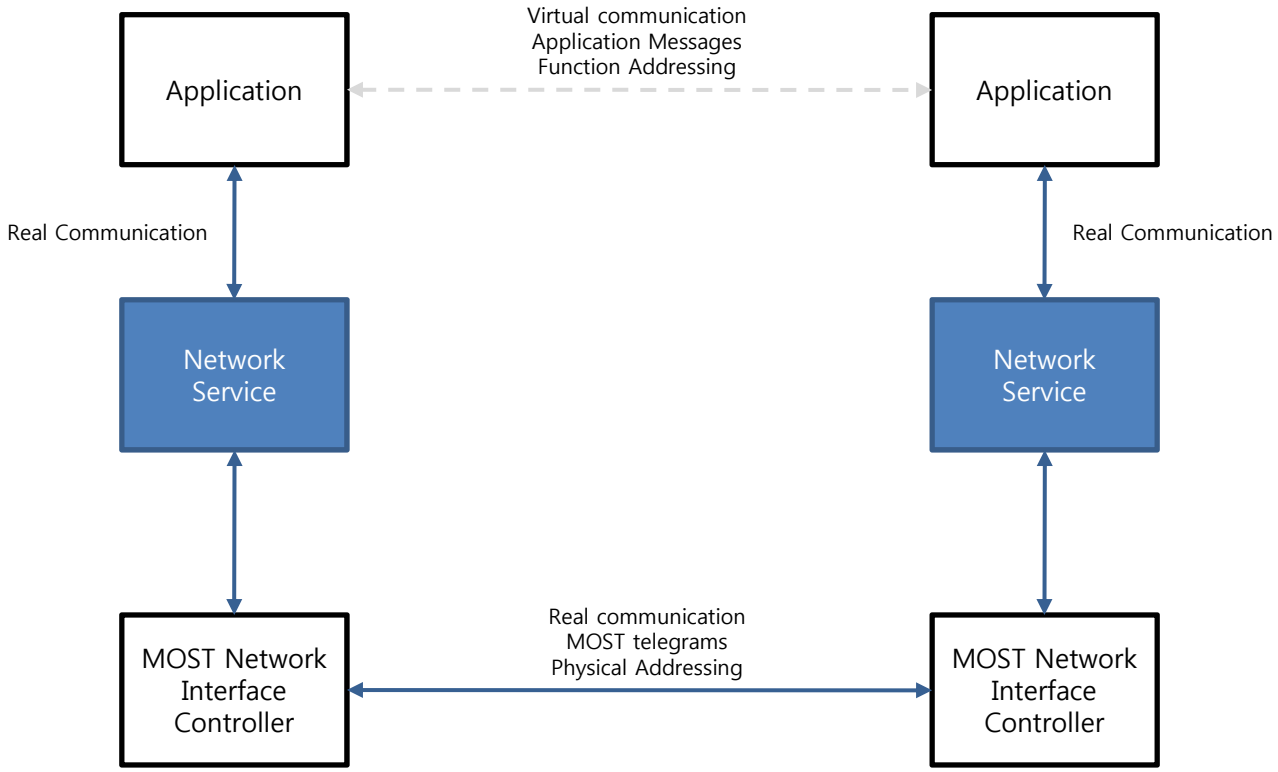
개요

- 차량 운행에 관한 장비들 간의 통신
  - CAN(Controller Area Network), LIN(Local Interconnect Network)

- 차량 내 멀티미디어 장비들의 통신을 위한 광대역 네트워크 시스템
  - MOST(Media Oriented Systems Transport)

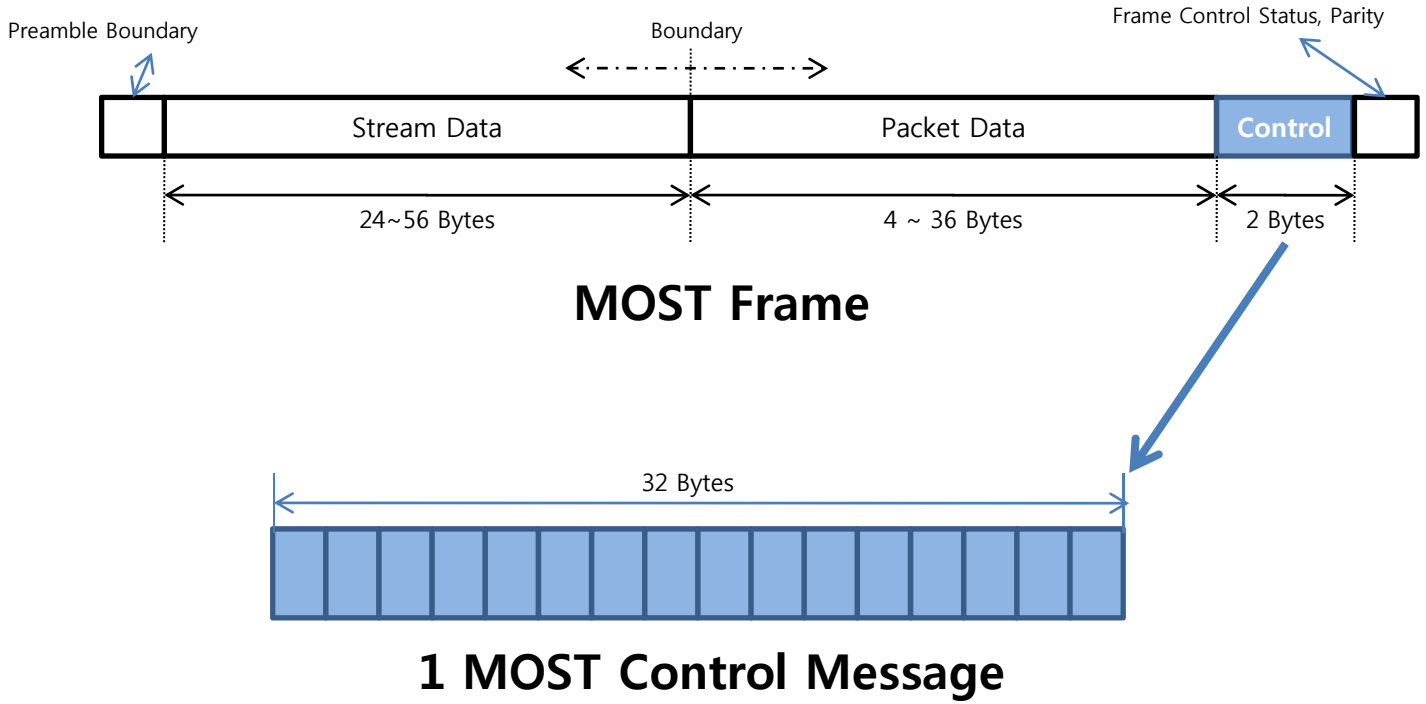
MOST Network Service

- Application 과 NIC 사이의 프로토콜 규약

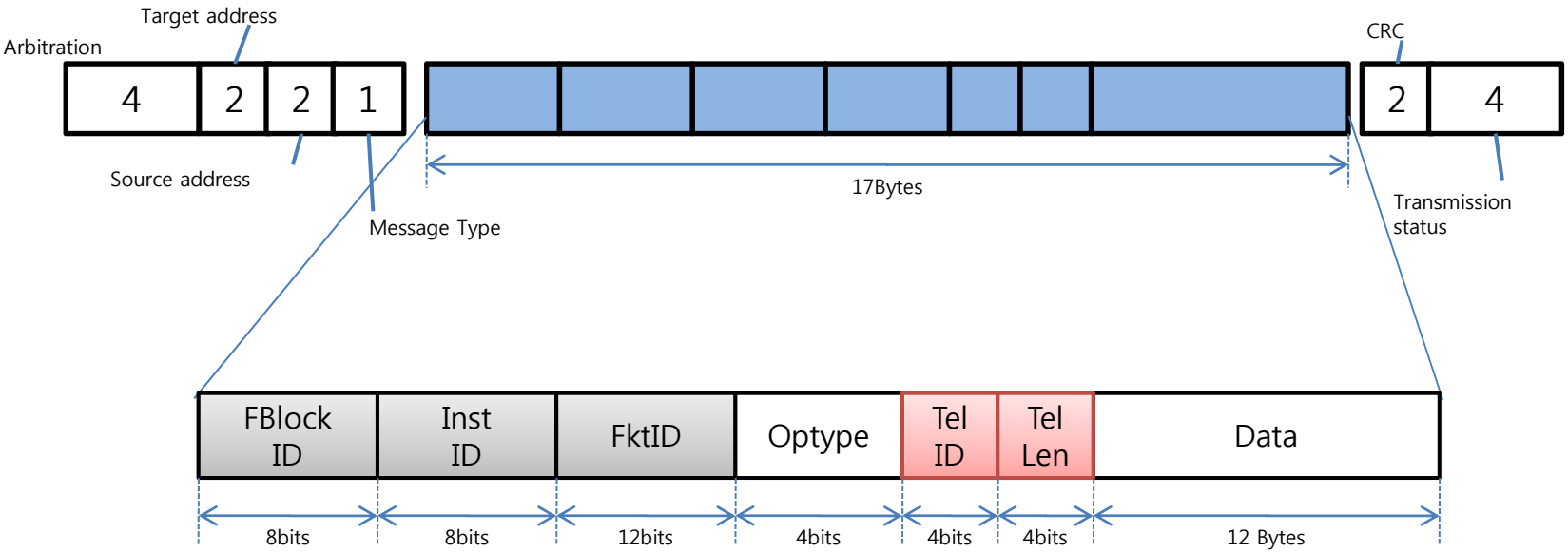


Protocol

- Stream Data / Pack Data / Control Data
- 1 MOST Control Message = 16 Control Data Frame



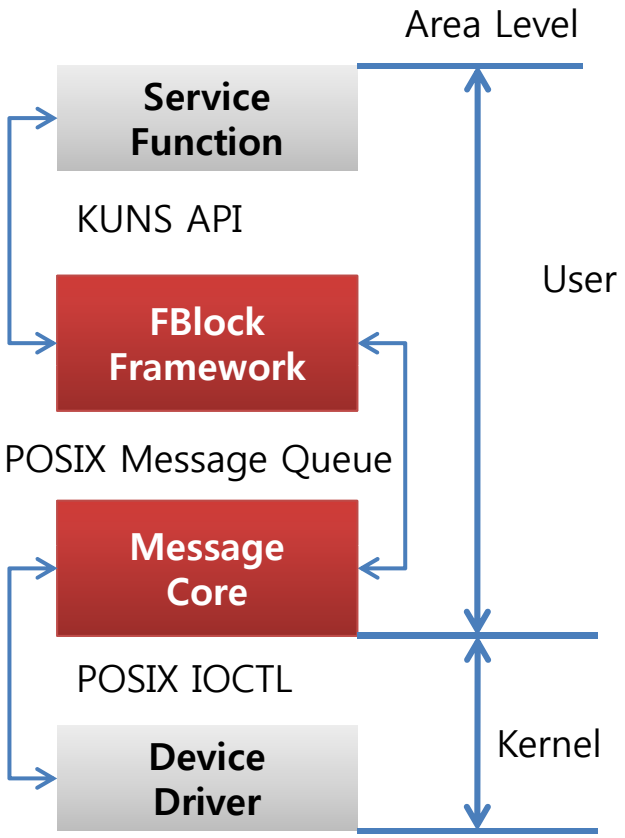
Protocol(Cont'd)



- 사용자 Application 제어를 위한 데이터 영역의 프로토콜
- 5 Bytes 헤더
  - 주소 정보(IDs)와 데이터 분할 및 병합을 위한 정보(Tel ID, Tel Len)
- 12 Bytes 데이터

u-OMNiPro

- u-OMNiPro 1.0의 구조
- FBlock Framework
  - FBlock 개발 및 관리 지원
- Message Core
  - FBlock 과 Device Driver 간의 통신 지원
- IPC
  - POSIX Message Queue
  - POSI IOCTL



u-OMNiPro 구조

FBlock Framework

- FBlock 생성 및 관리를 위한 API 제공
- 여러 개의 Service Function을 가질 수 있음

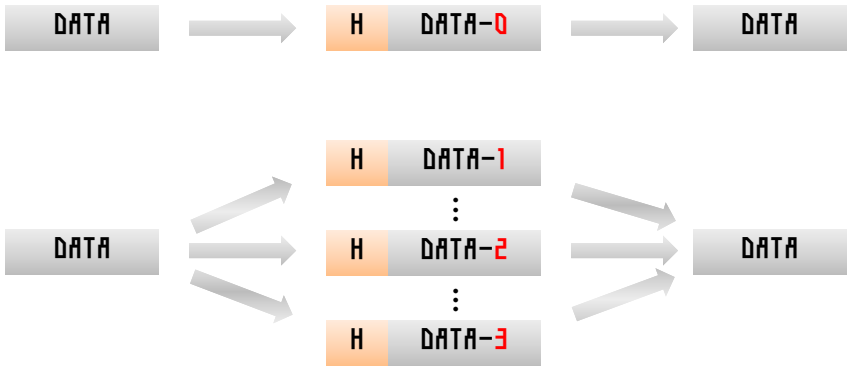
APIs	Description
KUNS_FBlock_Register	FBlock를 등록
KUNS_FBlock_Unregister	FBlock를 해지
KUNS_FuncktionID_Register	FBlock에 Function을 등록
KUNS_FunctionID_Unregister	FBlock에 Function을 해지
KUNS_FBlock_RUN	FBlock을 실행
KUNS_ctrl_recv	메시지 코어로부터 데이터 수신
KUNS_ctrl_send	메시지 코어로 데이터 송신

FBlock Framework API

Message Core

Single Telegram Messages

Tel ID	Tel Len	Data Byte 0 / Counter	Description
0	0 - C	Data	Single telegram

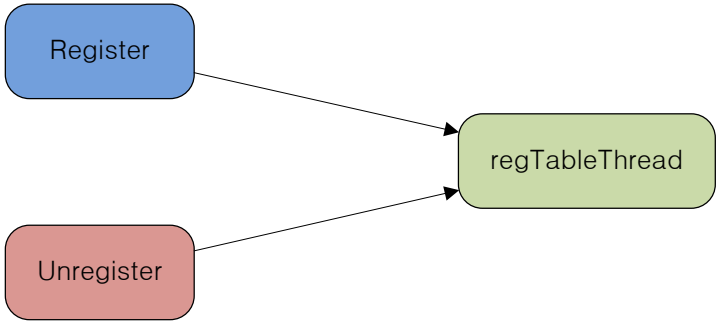


Multiple Telegram Messages

Tel ID	Tel Len	Data Byte 0 / Counter	Description
1	C	0X00	First Telegram
2	C	0x01	Subsequent Telegram
2	C	…	Subsequent Telegram
2	C	0xFF	Subsequent Telegram
2	C	0x00	Subsequent Telegram
2	C	…	Subsequent Telegram
2	C	0x0n – 1	Subsequent Telegram
3	2 - C	0x0n	Last Telegram

Modeling(FBlock Register & Unregister)

- 3 Processes
  - Register
  - Unregister
  - regTableThread
- Register
  - FBlock 등록 요청
- Unregister
  - FBlock 해지 요청
- regTableThread
  - FBlock 등록 및 해지
  - FBlock Table 관리



FBlock Register & Unregister model

```
typedef NSHandler{      byte fBlock;
                        byte instId;
                        byte status };
```

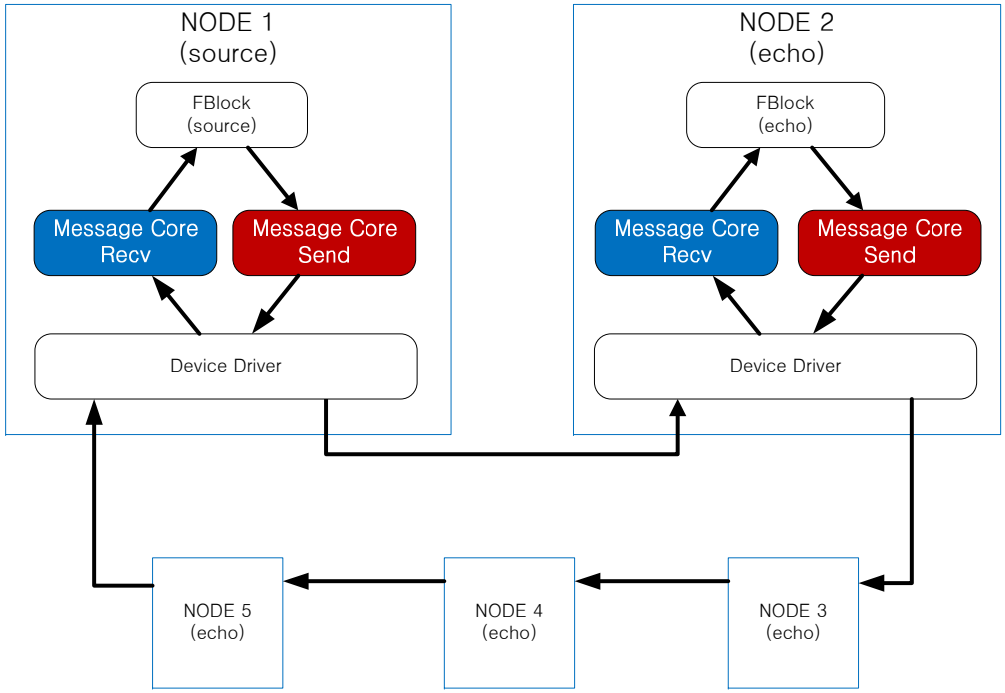
Data Structure

Modeling(Echo)

- 5 Nodes
  - 1 Source Node
  - 4 Echo Nodes
- FBlock
- Send / Receive
- Device Driver

```
typedef perfect_Ctrl_Data { byte tgt_addr;
                            short ctrl_data_len;
                            byte ctrl_data[6] };
typedef ctrl_data_t {       byte tgt_addr;
                            byte ctrl_data[6] };
```

Data Structure



u-OMNiPro echo model(5 Nodes)

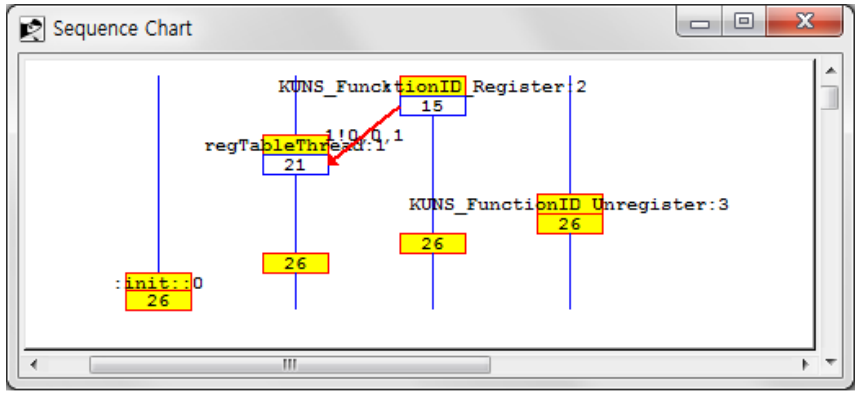
검증 결과(FBlock Register & Unregister)

- 현재 등록을 시도하려는 서비스 함수의 속성이 함수들을 관리하는 테이블에 올바르게 저장이 되었는가?

If
::(count < MAX_Table_Size) -> assert(pNsHandler[count-1].fBlock!=FBlock.buff[0]);
*::(count == 0) -> assert(pNsHandler[MAX_FUNCTION * MAX_INSTID - 1].fBlock != FBlock.buff[0]);*
fi;

- 해지 신청이 요청됐을 경우, 현재 저장되어 있는 해당 함수가 올바르게 해지되었는가?

assert(pNsHandler[count].fBlock != FBlock.buff[0])



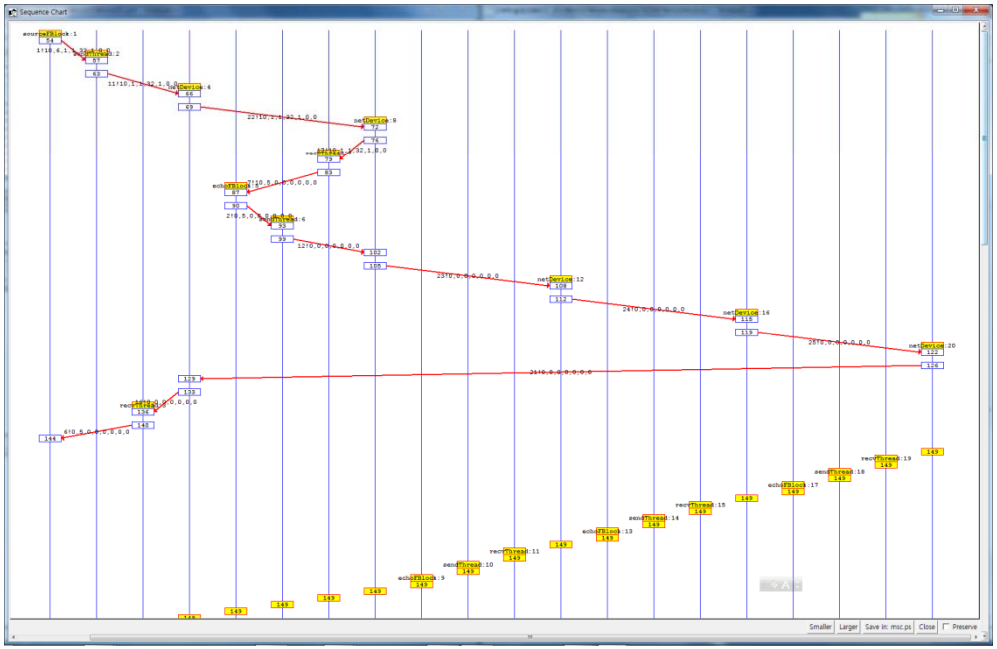
검증 결과(Echo)

- echo Node로부터 되돌아온 메시지가 원본 메시지와 동일한가?

  assert(recvdCtrlData.ctrl_data_len != sentCtrlData.ctrl_data_llen)

- 첫 번째 패킷이 전송되었다면 언제가 마지막 패킷이 보내짐을 보장하는가?

  []((sendTelID == 1) -> <> (sendTelID == 3))



결론 및 향후 연구

- 결론
  - u-OMNiPro 1.0 정형명세 및 정형검증
  - 모델 내의 두 개의 오류 발견
    - 미구현으로 인한 오류
    - 모델 밖에서 적용된 속성으로 인한 오류

- 향후 연구 방향
  - k-OMNiPro (kernel-level OMNiPro) 정형명세 및 정형검증