2021년 2월

석사학위 논문

위성과 지상국 사이의 데이터 송수신 패킷 처리 소프트웨어 디커플링으로 재사용성이 향상된 프로그램 개발

조선대학교 대학원

컴퓨터공학과

송 문 식

위성과 지상국 사이의 데이터 송수신 패킷 처리 소프트웨어 디커플링으로 재사용성이 향상된 프로그램 개발

Applying Software Decoupling Techniques to Enhance Software Reuse for processing data transmission and reception packets between satellites and ground stations

2021년 02월 25일

조선대학교 대학원

컴퓨터공학과

송 문 식

위성과 지상국 사이의 데이터 송수신 패킷 처리 소프트웨어 디커플링으로 재사용성이 향상된 프로그램 개발

지도교수 이 정 아

이 논문을 공학석사학위 신청 논문으로 제출함

2020년 10월

조선대학교 대학원

컴퓨터공학과

송 문 식

송문식의 석사학위논문을 인준함

위원장 조선대학교 교 수 정 일 용 (인)

위 원 조선대학교 교 수 김 판 구 (인)

위 원 조선대학교 교 수 이 정 아 (인)

2020년 11월

조선대학교 대학원

목 차

표 목 차

그 림 목 차

NOMENCLATURE

ACRONYMS

AX.25	Amateur X.25
CCSDS	Consultative Committee for Space Data Systems
cFS	core Flight Software/System
FCS	Frame Check Sequence
FSK	Frequency Shift Keying
HDLC	High-level Data Link Control
KISS	Keep It Simple, Stupid
KMSL	Korea Microgravity Science Laboratory
OBC	On-Board Computer
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PSK	Phase Shift Keying
RF	Radio Frequency
TLM	Telemetry
TNC	Terminal Node Controller

ABSTRACT

Applying Software Decoupling Techniques to Enhance Software Reuse for processing data transmission and reception packets between satellites and ground stations

Song, Moon-sik
Advisor : Prof. Lee, Jeong-A, Ph.D.
Department of Computer Engineering.
Graduate School of Chosun University

In recent years, a cube satellite (hereinafter referred to as CubeSat) has attracted attention in the space sector, and its platform is being used to perform a variety of scientific or technical satellite missions at low cost. To successfully perform the satellite mission, the communication protocol with a ground station must be carefully designed and established. Software reusability is one of the important elements in implementing the communication protocol to effectively reduce the system development period and its budget. This thesis deals with the decoupling techniques for processing data transmission and reception packets between CubeSat and ground station to enhance software reusability.

Processing data transmission and reception packets which are coupled with the CubeSat communication protocol such as AX.25 and KISS is a critical step in a CubeSat communication with the ground station. Since processing data packets coupled with the communication protocol is highly dependent on the types of hardware used in the ground station, a source code for data packet processing in the ground station software must be modified or re-developed by hardware specifications if any hardware replacements are needed.

In this thesis, the packet processing program was newly-designed on the basis

of the decoupling techniques and was tested with COSMOS, an open-source program for the ground station. The packet processing program was separately placed and interfaced with the COSMOS to receive data packets from the CubeSat and transmit data packets to the CubeSat. A series of successful communication demonstrations between the CubeSat and ground station clearly shows that the packet processing program developed as part of this study can successfully remove the hardware dependency from ground station software and thus improve its reusability.

Ⅰ. 서론

A. 연구 배경 및 필요성

우주 개발 기술의 완성도는 한 나라의 국방, 과학 기술 및 산업계의 수준을 가늠할 수 있는 척도이다. 이러한 이유로 세계 각국은 우주 개발을 통해 자국의 과학 기술과 관련 산업의 발전을 추구하고 있으며, 궁극적으로는 국가 경제와 국방 등의 광범위한 분야에서 상대적 우위를 점하고 국가 경쟁력을 강화할 목적으로 우주 개발에 박차를 가하고 있다. 이러한 흐름에 맞추어 우리나라도 본격적으로 우주 개발 기술에 대한 투자를 늘리고 있으며, 그림 1에서 볼 수 있는 바와 같이 우리나라의 GDP 대비 우주 예산은 세계 10위권을 유지하고 있다.

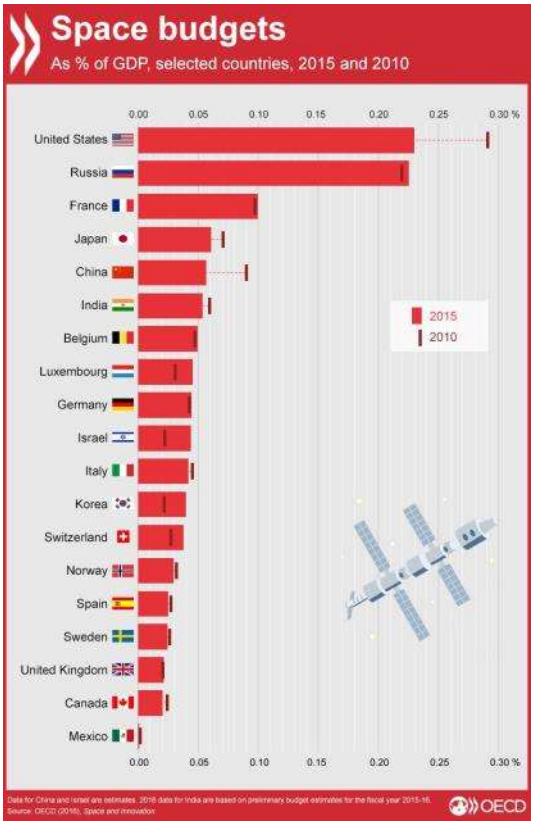


그림 1 세계 각국의 GDP 대비 우주 관련 예산의 비중[1]

출처: OECD (2016), Space and Innovation

최근의 우주 개발 경쟁은 세계 각국의 정부 주도에서 민간 경쟁으로 우주 개발 패러다임이 변모하고 있으며, 그중에서도 지구 주위를 돌면서 관측 및 다양한 서비스를 제공하는 인공위성 활용 분야가 우주 기술 경쟁에서 중요한 분야로 주목받고 있다. 우리나라도 위성 활용 서비스 및 관련 장비 개발 분야가 우주 기술 개발 분야의 약 34%로 다른 우주 기술 분야에 비해 높은 편이며, 우리나라도 최초의 인공위성인 우리별 1호를 성공적으로 제작·발사함으로써 위성 활용 분야에 필요한 기초 노하우를 습득하고 고급 연구인력 양성에 필요한 기틀을 마련하였다.

민간 주도의 위성 활용 서비스 경쟁을 견인한 요인 중 하나로 초소형 위성기술의 비약적인 발전을 꼽을 수 있다. 표 1에 정리된 바와 같이 기존 대형위성은 개발 기간이 길고 상대적으로 많은 제작 비용으로 인해 발사를 자주 하기 힘든 반면 소형위성은 짧은 개발 기간과 적은 비용으로도 발사가 가능하기 때문에 교육, 연구 및 대형위성에 탑재하기 전 시험용으로도 널리 사용되고 있다.

표 1 위성의 무게에 따른 분류 및 제작 비용

Class			무게	제작 비용 (USD)
기존 위성	대형 위성		> 1000kg	1억 5000만 이상
	소형 위성		500 ~ 1000kg	5000만 ~ 1억 5000만
소형 위성	미니 위성		100 ~ 500kg	1000만 ~ 5000만
	마이크로 위성		10 ~ 100kg	100만 ~ 1000만
	초소형 위 성	큐브(나노) 위성	1 ~ 10kg	10만 ~ 100만
		피코 위성	< 1kg	10만

출처: http://www.spacetimes.co.kr/news/articleView.html?idxno=451

1999년 칼 폴리 대학의 Jordi Puig-suari 교수와 스탠포드 대학의 Bob Twiggs 교수의 제안으로 큐브위성 프로젝트가 시작되었으며, 초소형 위성의 발전은 큐브위성과 함께 시작하였다고 볼 수 있다. 큐브위성 크기의 기본 단위를 1U(Unit)라고 하는데, 한 변의 길이가 10㎝인 정육면체 모양이고 무게는 1.3kg 이하로 구성된다. 임무에 따라 1U에서부터 12U 이상 다양한 크기로 확장될 수 있다. 국내에서는 2000년대 초부터 제작 발사를 하였으며, 한국항공우주연구원은 대학(원)생들에게 초소형 위성을 제작해

볼 수 있는 기회를 제공하여 우주 기술 개발의 저변을 확대하고 전문인력 양성을 위해 2012년을 시작으로 2013년부터 격년으로 “큐브위성 경연대회”를 개최하고 있다[2]. 표 2에는 2012년부터 2019년까지 경연을 통해 선정된 대학과 위성명이 정리되어 있다.

표 2 한국항공우주연구원 큐브위성(CubeSat) 경연대회 현황

경연연도	대학	위성명	소프트웨어
2012	KAIST	LINK	-
	연세대	CANYVAL-X	자체 개발[3]
	항공대	KAUSAT-5	자체 개발[4]
2013	경희대	SIGMA	자체 개발[5]
	조선대	STEP Cube Lab.	상용 소프트웨어[6]
	충남대	CNUSAIL-1	-
2015	서울대	SNUGLITE	자체 개발[7]
	서울대	SNUSAT-2	자체 개발[8]
	항공대	VisionCube	-
2017	연세대	CANYVAL-C	위성 : 공개 소프트웨어(cFS), 지상국 : 자체 개발
	조선대	KMSL	공개 소프트웨어(cFS, COSMOS)
2019	KAIST	ASTRIS-2	-
	서울대	SNUGLITE	-
	연세대	MIMAN	-
	조선대	STEP Cube Lab-2	-

큐브위성도 기존의 중대형 위성과 마찬가지로 임무 수행을 위해서는 지상국과의 교신이 필요하다. 따라서 임무 성공을 위해서는 비행 소프트웨어를 사전에 검증하고 위성에서 생성된 자료들(위성 정보, 임무 수행 결과 등)을 수신하기 위해 위성-지상국 간의 데이터 송수신이 안정적으로 이루어지는지 충분한 검증이 필요하다. 하지만 표 2에 요약된 바와 같이 기존에 경연을 통해서 발사하였던 팀이 제작한 위성을 살펴보면, 비행 소프트웨어와 지상국 소프트웨어를 대부분 자체 개발하여 사용하고 있다. 자체 개발하는 팀들 대부분이 위성 본연의 임무에 맞춰 소프트웨어를 개발하고 있지만, 위

성의 공통된 기본 기능(패킷 전송)의 구현에 필요한 소프트웨어뿐만 아니라 지상국 운용에서 공통된 작업(위성에서 전송한 패킷의 프로토콜 처리)을 처리하는 소프트웨어의 재사용성을 고려하지 않고 제작하는 실정이다. 즉, 위성 관제를 위해 필요한 위성-지상국 간 통신 소프트웨어를 개발하면서 위성을 제작하는 팀마다 혹은 임무마다 패킷 송신의 구현에 필수적인 공통 요소까지도 매번 새로 개발하여 사용함에 따라 개발된 소프트웨어의 검증 시간이 부족하거나 안정적인 운영 여부를 보장할 수 없는 실정이다. 안정성이 확보되어 잘 운영되고 있는 소프트웨어를 재사용하는 것이 위성 개발 기간의 단축은 물론 성공적인 위성의 관제에 있어서 매우 중요한 요소이다.

B. 연구의 목적 및 내용

KMSL (Korea Microgravity Science Laboratory) 초소형 위성은 과학기술정보통신부 주관으로 개최된 “2017년 큐브위성 경연대회”를 통해 선발되어 2020년 10월 현재 개발이 거의 완료되어 2021년 1분기 중으로 발사될 예정이다. KMSL 초소형 위성의 개발 및 운영 목적은 마이크로중력, 절대진공, 우주 방사능과 같은 극한 환경에서 연소실험과 생물육성실험의 2가지 과학 임무를 성공적으로 수행하고, 해외제품에 주로 의존하고 있는 전자모듈 일부를 국산 제품으로 대체하여 실제 우주 환경에서 작동 성능을 검증하는 것이다. 더 나아가 초소형 위성을 활용하여 우주 환경에서 다양한 과학실험을 수행할 수 있는 플랫폼을 구축하는 것이 최종 목표이다.[9]

앞서 언급한 바와 같이 KMSL 초소형 위성도 수행 임무의 용도에 맞게 비행 소프트웨어와 지상국 소프트웨어를 개발하고 있다. 위성마다 하드웨어 구성 및 소프트웨어 특징이 서로 다르므로 모든 위성의 운영 환경을 고려하여 위성-지상국 전반에 걸쳐 재사용이 가능한 소프트웨어를 제작하는 것은 거의 불가능하다. 따라서 이 논문에서는 조선대학교에서 개발하고 있는 KMSL 큐브위성 개발 환경을 바탕으로 위성-지상국 간 통신 패킷 처리 소프트웨어의 재사용성을 극대화하기 위하여 지상국 소프트웨어의 분리(decoupling)에 대해서 다룬다.

KMSL 큐브위성은 비행 소프트웨어로 NASA에서 개발한 cFS(core Flight System)를 사용하고, 지상국 소프트웨어는 COSMOS를 사용한다. 두 소프트웨어 모두 위성의 기본 운영을 위한 플랫폼을 제공하고 각 위성의 임무에 따른 기능을 추가할 수 있도록 구성된 Open Source 소프트웨어이다.

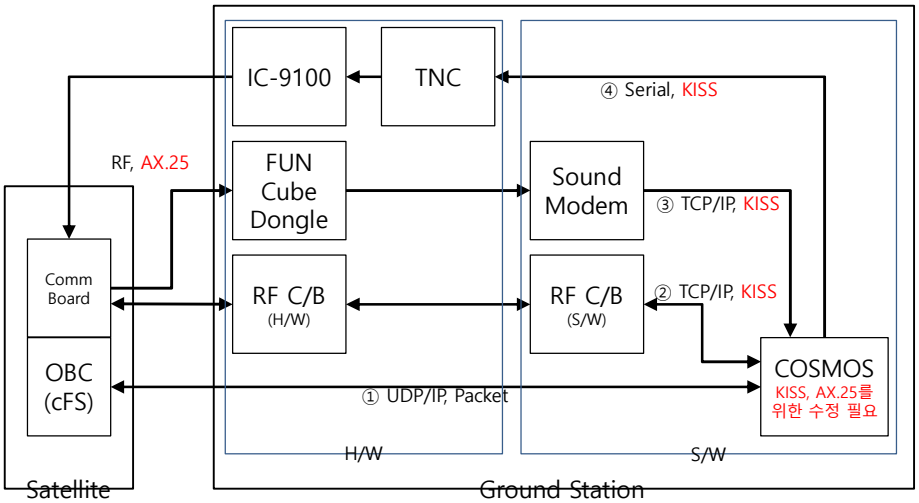


그림 2 COSMOS를 수정을 통한 위성 통신 시험 및 운영 환경

위성 개발 초기에는 그림 2의 ①과 같이 비행 소프트웨어(cFS)와 지상국 소프트웨어(COSMOS)를 이더넷(UDP/IP)으로 연결하여 패킷만 송수신하여 프로그램을 검증한다. 프로그램 검증이 되면, 그림 2의 ②와 같이 궤도에서 운영하는 위성과 통신하는 방법이 같은 실내용 RF 환경을 구축하여, 패킷에 프로토콜(KISS, AX.25)을 추가하여 시험한다. 최종적으로 그림 2의 ③, ④와 같이 실제 운영 장비를 사용하여 근거리 시험(far-field test)을 거친 후 발사한다. 그림 2와 같이 시험 및 운영 환경에 따라서 통신 및 프로토콜 방식이 바뀌게 되며, 이를 위하여 기존의 COSMOS를 수정하는 것이 필요하다.

그러나, 위성과 지상국 사이의 데이터 송수신 패킷 처리를 위해서 개발, 시험 및 운영 과정에서 COSMOS를 수정하게 되면, 어느 부분(응용 프로그램, 프로토콜 처리, 통신)에서 문제가 발생하였는지 파악하기 어렵고, 서비스 간의 종속성 증가로 인하여 한 서비스 수정 시 다른 서비스 수정(그림 2의 ①번에서 ②번으로 변경 시 COSMOS 프로그램에서 프로토콜 처리 추가, ②번에서 ①번으로 변경 시 COSMOS 프로그램에서 프로토콜 처리 제거)이 필요하다.

본 연구에서는 프로토콜(KISS, AX.25 등) 처리를 제공하지 않는 COSMOS를 수정하지 않고, 지상국의 재사용성 향상을 위해 패킷 처리에 소프트웨어 디커플링 기법을 적용하고자 한다. 즉, 송수신기/모뎀과 관제 프로그램(COSMOS) 사이에 데이터 송수신 패킷의 프로토콜 처리를 하기 위한 프로그램(이하 패킷 처리 프로그램)을 개발하여 검증한

다.

패킷 처리 프로그램을 개발하기 위하여, KMSL 큐브위성과 지상국에서 사용하는 통신 하드웨어 및 소프트웨어를 파악하고, 지원하는 통신 방법과 프로토콜 및 요구 사항을 분석한 뒤, 패킷 처리 프로그램을 송수신기와 COSMOS 사이에 적용하여 위성과 데이터 송수신이 제대로 처리되는지 검증한다.

본 논문은 1장 서론을 포함하여 총 5장으로 구성되었다. 2장에서는 논문과 관련되어 필요한 이론인 소프트웨어 디커플링 기법과 전파 및 통신 프로토콜에 관한 내용을 다루고, 위성과 지상국 소프트웨어 및 위성 운영을 위해 두 소프트웨어 사이에 필요한 하드웨어/소프트웨어를 설명한다. 3장에서는 디커플링 기법을 적용한 지상국 계층을 소개하고, 패킷 처리 프로그램의 요구 사항인 다양한 통신 방식, 프로토콜 처리, 패킷 데이터 병합 처리에 관해서 설명한다. 4장에서는 패킷 처리 프로그램에 대한 검증 및 평가하고, 5장에서는 연구 결과를 정리한다.

Ⅱ. 연구의 배경 이론

우주 임무를 수행하기 위해서는 실제로 우주에서 임무를 수행하는 위성(Satellite)과 지상에서 관제 및 명령을 수행하는 지상국(Ground Station)이 있고, 위성과 지상국 사이의 통신으로 구분할 수 있다. 위성 및 지상국 운영을 위한 소프트웨어와 통신 방식은 다양하다. 이 장에서는 KMSL 큐브위성 운영에 적용되는 소프트웨어와 통신, 프로그램 개발에 적용한 소프트웨어 디커플링 기법에 대해서 다룬다.

A. 비행 소프트웨어(cFS)

비행 소프트웨어는 위성의 운영에 사용되는 소프트웨어로 위성의 전원이 켜지면 지상의 명령 없이 자율적으로 실행이 가능하여야 한다. 지상의 통제를 받지 않은 상황에서도 위성을 운영하여야 하므로 안정성이 보장되어야 하며, 지상국으로부터 명령을 수신하여 임무 수행이 가능하고, 임무 수행 결과와 위성의 상태 자료를 지상국에 전송이 가능하여야 한다.

cFS(core Flight System)[10, 11, 12]는 미항공우주국(NASA - National Aeronautics and Space Administration) 고더즈 우주 센터(GSFC - Goddard Space Flight Center)의 FSSB(Flight Software Systems Branch)에서 개발한 재사용 가능한 비행 소프트웨어(FSW) framework이다. cFS는 이전 비행 소프트웨어의 재사용성이 개발 비용과 기간을 단축하지 못했기 때문에, 재사용성과 시스템 간 이전이 용이하게 만들었다. 2009년 6월에 발사된 달 탐사선인 LRO(Lunar Reconnaissance Orbiter)를 시작으로 현재까지 발사되는 대부분의 위성에 비행 소프트웨어로 사용하고 있는 검증된 플랫폼이다.

1. cFS 특징

a. 계층화된 서비스 아키텍쳐

각 계층 및 서비스는 표준 API를 제공하며, 미들웨어, 운영체제 및 하드웨어 플랫폼의 독립성을 제공한다. 각 계층은 다른 계층의 내부 및 구성 요소에 영향을 주지 않고 변경이 가능하다.

b. Plug and Play

기존의 구성 요소 간에 밀접하게 결합된(tightly-coupled) 함수 호출 방식이 아닌 message coupling(loosely-coupled)에 기반한 Software Bus를 통하여 어플리케이션 간에 메시지를 전송하여 서로 독립적으로 작동한다. cFE API는 어플리케이션의 추가 및 제거를 지원하며 시스템 구성 요소를 재부팅하거나 재구성하지 않고 런타임에 SW 구성 요소를 전환할 수 있다.

c. 재사용이 가능한 구성 요소

시험을 거쳐 인증된 구성 요소를 재사용하여 소프트웨어 개발 주기 단축, 비용 절감이 가능하다.

2. cFS 계층과 구성 요소

cFS 계층 구조와 구성 요소는 그림 3에 도식하였다.

<table>
<tr><td>Mission Specific
cFS Applications</td><td>cFS Reusable
Applications</td><td>Scheduler, Housekeeping, CFDP, 등의
어플리케이션으로 구성</td></tr>
<tr><td colspan="2">cFE(core Flight Executive) API</td><td>ES, EVS, SB, TBL,
Time Service로 구성</td></tr>
<tr><td colspan="2">Operating System Abstraction Layer
(OSAL) API</td><td>여러 OS에서의 시스템 함수를 동일한
인터페이스로 제공해 주는 계층</td></tr>
<tr><td colspan="2">Platform Specific Package
(PSP)</td><td>RTEMS, VxWorks, Linux support</td></tr>
</table>

그림 3 cFS 계층 구조

a. OSAL API, PSP

OSAL API 및 PSP는 공통 OS 및 BSP(Board Support Package) 서비스를 제공하는 플랫폼 독립적 (OS 및 하드웨어) 인터페이스를 제공한다. 플랫폼 추상화 계층은 하드웨어 및 OS 구현 세부 사항에서 상위 계층과 분리(디커플링)하여, cFE 및 어플리케이션을 다양한 플랫폼에서 변경 없이 실행할 수 있다.

b. cFE(core Flight Executive) API

비행 소프트웨어에서 공통으로 사용되는 핵심적인 기능을 제공하며, 기능에 따라 5가지 Service로 구분되며 표 3과 같다.

표 3 cFE API

Service	기능
ES (Executive Services)	◆ cFE의 시작을 관리 ◆ cFS 어플리케이션의 시작/재시작/정지하는 기능을 제공 ◆ 중요 데이터 저장소를 관리 ◆ 장치 드라이버를 지원
EVS (Event Services)	◆ 이벤트 메시지 생성 및 송신 서비스 제공 ◆ 메시지 필터 제공 ◆ 발생 이벤트의 파일 저장 가능
SB (Software Bus)	◆ 어플리케이션간 메시지 송/수신 서비스 제공 ◆ 메시지를 구독한 모든 application으로 메시지를 전달 ◆ 메시지 전송 중 감지된 오류 보고 ◆ 명령이 있을 때 통계 패킷 및 라우팅 정보를 출력 ◆ 메시지 형식은 CCSDS 표준을 따름
TBL (Table Services)	◆ 파라메터들의 집합 ◆ 각 어플리케이션에서 파라메터 관리를 일괄적으로 할 수 있게 해주는 서비스 제공 ◆ 런타임에서의 수정 및 업데이트 가능(커맨드 사용) ◆ 어플리케이션간 테이블 공유 가능
TIME (Time Services)	◆ cFE 어플리케이션이 시간을 관리하는 API 제공 ◆ “1Hz wakeup” 명령 패킷을 배포

c. cFS Reusable Applications

표 4와 같이 시험을 거쳐 검증된 어플리케이션을 사용하여 소프트웨어 개발 기간 단축, 비용 절감, 위험 감소할 수 있다.

표 4 cFS Reusable Applications

Application	설명
CFDP	지상국과 파일 데이터를 송수신
Checksum	메모리, 테이블 및 파일의 데이터 무결성을 검사

Command Ingest	전송 채널을 통해 외부 소스(예 : 지상국)로부터 명령을 수신하고, cFE 소프트웨어 버스(SB)를 통해 해당 응용 프로그램으로 명령을 전달
Command Ingest Lab	지상국으로부터 UDP/IP port를 통해 CCSDS 형식의 명령 패킷을 수신
Data Storage	downlink를 위해 위성 상태 및 임무 데이터를 기록
File Manager	파일 관리를 위한 인터페이스
Housekeeping	다른 어플리케이션에서 수집된 데이터를 원하는 형식으로 재가공
Health and Safety	위성의 상태를 점검하고 감시
Limit Checker	임계값을 초과할 때 값을 모니터링하고 조치를 수행
Memory Dwell	메모리 위치의 내용을 원격 측정할 수 있도록 조치
Memory Manager	메모리를 로드하고 덤프
Software Bus Network	다양한 네트워크 프로토콜을 통해 소프트웨어 버스 메시지를 전달
Scheduler	미리 정해진 타이밍 간격으로 메시지를 생성
Scheduler Lab	1초 단위의 간단한 메시지를 생성
Stored Command	절대 및 상대적인 시간으로 정해진 명령을 전달
Telemetry Output	전송 장치를 통해 외부 대상(예 : 지상국)으로 원격 측정을 전송하는 일을 담당
Telemetry Output Lab	UDP/IP port를 통해 지상국으로 CCSDS 형식의 명령 패킷을 송신

d. Mission Specific cFS Applications

위성의 임무에 맞춰 필요한 어플리케이션을 개발하여 사용한다. KMSL 위성의 경우, 그림 4와 같이 5개의 cFE API, 4개의 cFS Reusable Application, 7개의 Mission Specific Application과 6개의 Mission Library를 사용한다.

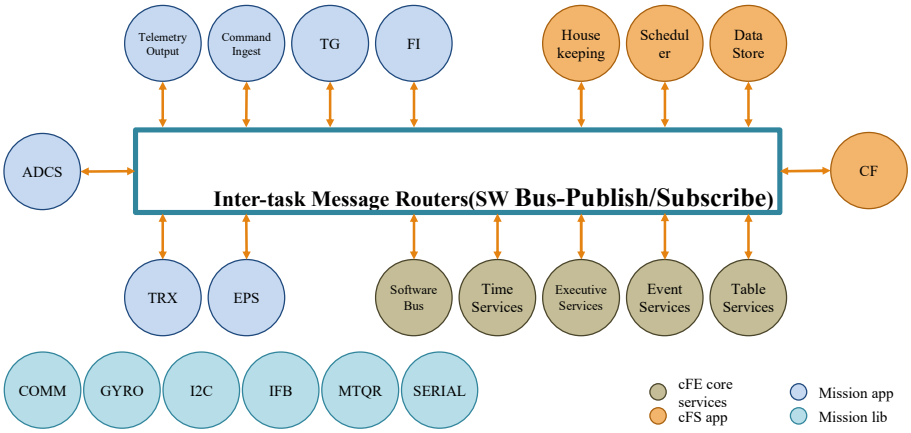


그림 4 KMSL Application Diagram

B. 지상국 소프트웨어(COSMOS)

Ball Aerospace COSMOS[13]는 위성의 운영 및 시험을 위한 지상국 소프트웨어로, 명령 및 제어 시스템을 지원하는 인터프리터 방식인 Ruby로 개발된 오픈 소스 프로그램이다.

1. 특징

- 도구 구성 : 실시간 명령 및 스크립팅, 실시간 원격 측정 시각화, 오프라인 분석, Utilities
- TCP/IP, UDP 및 Serial 연결을 통한 interface 제공
- 대상과의 실시간 통신에서 이루어진 모든 command와 telemetry 정보가 로그 파일에 기록
- 모든 tool은 일반 텍스트 설정 파일로 구성
- 탁월한 데이터 시각화
- 강력한 스크립트
- Cross platform : Windows, Linux 및 Mac OSX를 지원

2. 응용 프로그램 및 용어

명령을 전송하고 데이터(그래프, 텍스트 표시)를 시각화하고 자동화된 스크립트를 작성하며 로그 파일을 분석하고 원격 측정을 모니터링할 수 있는 응용 프로그램 세트

를 제공한다.

a. Launcher

Launcher는 COSMOS 시스템을 구성하는 각 응용 프로그램을 실행하기 위한 그래픽 사용자 인터페이스를 제공한다.

b. Command and Telemetry Server

Command and Telemetry Server는 COSMOS의 실시간 중계 역할을 한다. 모든 command 및 telemetry packet이 이 프로그램을 통하여 발생하는 모든 사항이 기록된다. 실시간 명령, 원격 측정 수신, 로깅, 제한 모니터링, 패킷 라우팅 및 시스템 상태를 제공한다.

c.. Command Sender

Command Sender는 개별 명령을 수동으로 전송하기 위한 그래픽 인터페이스를 제공한다. 시스템에서 모든 명령 및 명령 매개 변수를 드롭다운하여 개별 명령을 쉽게 전송할 수 있다. History pane(내역창)을 사용하면 이전 명령을 쉽게 다시 보낼 수 있다.

d. Packet Viewer

Packet Viewer는 정의된 모든 원격 분석 패킷의 실시간 시각화를 제공한다. 패킷 내의 값은 구성이 필요없는 간단한 key-value 형식으로 표시된다.

e. Telemetry Viewer

Telemetry Viewer는 고급 레이아웃 및 시각화 위젯과 함께 사용자 정의 원격 분석 화면 기능을 제공한다. 탭, 그래프, 한계 막대 및 기타 애니메이션 디스플레이를 빠르게 만들 수 있다. 또한 필요에 따라 사용자가 정의할 수 있는 모든 원격 분석 패킷에 대한 기본 화면 세트를 자동 생성할 수 있다.

f. Telemetry Grapher

Telemetry Grapher는 원격 측정 데이터의 실시간 및 오프라인 그래프를 제공한다. 여러 탭, 플롯 및 플롯당 항목을 사용하여 선 및 x-y 스타일 플로팅을 모두 지원한다.

g. Data Viewer

Data Viewer는 다른 데이터 시각화 패러다임에 맞지 않는 항목에 대한 텍스트 기반 원격 분석 시각화를 제공한다.

h. Handbook Creator

Handbook Creator는 사용 가능한 Command 및 Telemetry packet에 대한 html 및 pdf 문서를 생성한다.

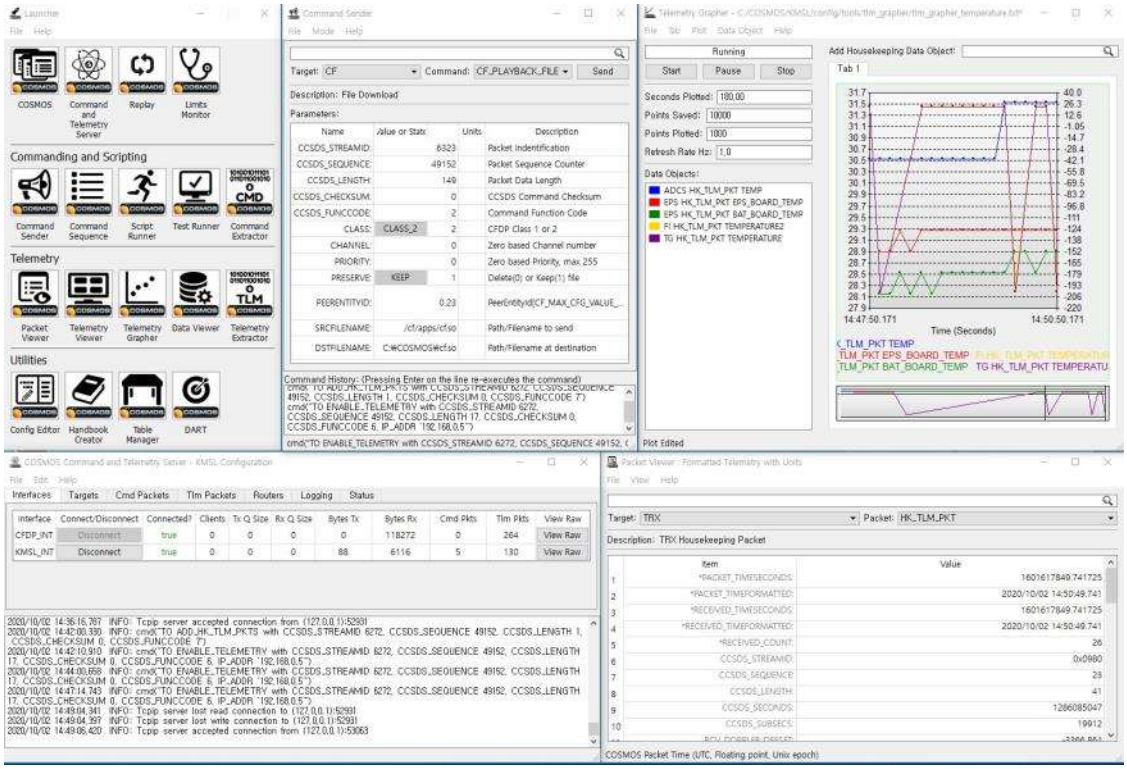


그림 5 COSMOS Tools

i. COSMOS 용어

용어	정의
Target	COSMOS가 위성과 명령을 송수신할 수 있는 시스템
Interface	대상에게 command를 보내거나 telemetry를 수신할 수 있는 방법을 정의한 Ruby 클래스
Configuration File	Command 및 telemetry 패킷을 정의하고 COSMOS application을 구성할 수 있는 텍스트 구성 파일
Tool	COSMOS application의 다른 이름

C. 위성 통신

1. 디지털 변복조 방식

디지털로 표현되는 데이터의 전송을 위해 데이터 신호에 따라 반송파(정보를 전달하는 높은 주파수의 정현파)인 기본 신호의 일부 특성을 변경시켜서 전송하는데, 디지털 데이터를 아날로그 신호로 변환시키는 장치를 변조기(Modulator)라고 한다. 반대로 들어오는 아날로그 신호를 디지털 데이터로 변환시키는 장치를 복조기(Demodulator)라고 한다. 변복조 장치는 아날로그 전송선과 디지털 장치를 연결하는데 사용되며, 변조기에서 mo와 복조기에서 dem을 합쳐서 modem이라고 한다. 과거에 모뎀(modem)은 하드웨어로 구현을 하였지만, 요즘은 컴퓨터 성능이 향상되어 소프트웨어로 구현을 많이 한다.

$$A\cos(2\pi ft+\phi)$$

정현파는 신호 값이 위 수식과 같이 삼각함수로 나타내는 신호로서, 3개의 파라미터(진폭, 주파수, 위상)를 가진다. A는 신호의 진폭(amplitude), f는 신호의 주파수(frequency), ϕ는 위상(phase)을 나타낸다.

디지털 변조의 기본 형태는 전송되는 신호의 특성에 따라 ASK(Amplitude Shift Keying - 진폭 변조), FSK(Frequency Shift Keying - 주파수 변조), PSK(Phase Shift Keying - 위상 변조) 등으로 분류하고, 디지털 데이터 0 또는 1 값에 따른 파형은 그

림 6과 같다.

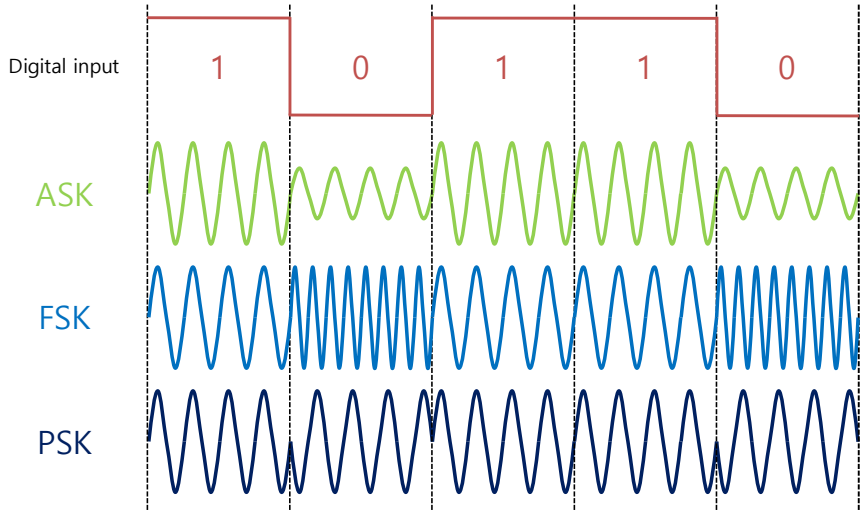


그림 6 ASK, FSK and PSK modulation

- ASK : 반송파의 진폭을 결정하는 변조 방식
- FSK : 반송파의 주파수가 여러 값 사이에서 천이되도록 하는 변조 방식
- PSK : 데이터 값에 따라 반송파의 위상이 여러 가지 상태로 천이되도록 하는 변조 방식. 2개의 다른 위상 값을 갖는 PSK를 BPSK(Binary Phase Shift Keying)라고 한다.

2. 통신 프로토콜

a. OSI 7계층

OSI 7계층은 서로 다른 여러 호스트를 연결해 통신하기 위하여 국제 표준 단체인 ISO(International Standard Organization)에서 제안한 표준 모델이며, 각 계층의 구성은 표 5와 같다.

표 5 OSI 7계층

구분	TCP/IP모델	설명	대표 프로토콜
Application layer (응용계층)	응용 계층	사용자가 네트워크에 접근할 수 있도록 함	HTTP, FTP, Telnet
Presentation layer (표현계층)		시스템 사이에서 교환되는 데이터 형식 규정	JPEG, MPEG, SMB
Session layer (세션계층)		통신하는 시스템들 사이의 연결을 설정	TLS, SSH, RPC
Transport layer (전송계층)	전송 계층 (TCP/UDP층)	프로세스 간의 메시지 전달	TCP, UDP, RTP
Network layer (네트워크계층)	인터넷 계층 (IP층)	전송 경로를 설정	IP, ICMP, ARP
Data link layer (데이터링크계층)	네트워크 액세스 계층	전송 흐름, 오류를 제어	LAN, Wifi, X.25
Physical layer (물리계층)		시스템의 물리적인 요건	RS-422, Cable, Fiber

b. AX.25 Link Access Protocol

AX.25(Amateur X.25)[14]는 OSI 모델의 2번째 계층인 데이터 링크 계층 프로토콜이며 아마추어 무선 통신 사업자가 사용하도록 설계되었다. 출발지와 목적지 주소 및 제어 정보 등을 포함한다.(표 6)

표 6 AX.25 Link Access Protocol 구성

Flag	Address	Control	PID	Info	FCS	Flag
7E	14/28 bytes	1/2 bytes	1 bytes	N bytes	2 bytes	7E

c. KISS Protocol

KISS(Keep It Simple, Stupid)는 호스트 시스템과 TNC(Terminal Node Controller) 간의 통신에 가장 일반적으로 사용하는 프로토콜이다. 시작과 끝을 표시하는 특수문자(C0)를 사용하여 비동기 프레임 데이터를 간단하게 구분할 수 있다(표 7). CRC나 체크섬은 지원하지 않으며, 특수문자 4개는 다음과 같다.

- C0 - FEND Frame End
- DB - FESC Frame Escape
- DC - TFEND Transposed Frame End
- DD - TFESC Transposed Frame Escape

표 7 KISS Protocol 구성

Start Flag	Port + Command	Info	End Flag
C0	4 bits + 4 bits	N bytes	C0

d. CCSDS

1982년 세계 주요 우주 기관이 설립한 CCSDS(Consultative Committee for Space Data Systems)는 우주 비행을 위한 통신 및 데이터 시스템 표준 개발을 위한 다국적 포럼이다. CCSDS Blue Books[15]은 특정 인터페이스, 기술 기능 또는 프로토콜을 정의하거나 인터페이스, 프로토콜 또는 인코딩 방식과 같은 기타 제어 표준의 규범들을 정의한다.

3. Command/Telemetry 패킷 구조

위성의 비행 소프트웨어 cFS와 지상국 관제 소프트웨어인 COSMOS 사이에 송수신하는 command와 telemetry 패킷 구조는 CCSDS 표준을 따르며 표 8에 정리하였다. 패킷은 packet primary header와 packet data field로 구분이 된다. Packet data field는 packet secondary header와 user data field로 구분이 되며, 최소 1 byte에서 최대 65536 bytes를 포함한다. cFS는 packet secondary header(command - 2 bytes, telemetry - 6 bytes)를 포함하여, 패킷에 따라서 user data field가 없을 수도 있다. 패킷 생성시 byte order에 주의하여야 한다.

표 8 Space packet 구성

a) Space packet

ID		size	description
Packet primary header		6 bytes	mandatory
Packet data field	Packet secondary header	Variable	1 to 65536 bytes, mandatory
	User data field	Variable	

b) CCSDS packet primary header(6 bytes)

ID	size	offset	description
StreamId	16 bit		packet identifier word
packet version number	3 bit	0xE000	0으로 설정
packet type	1 bit	0x1000	0 = TLM, 1 = CMD
secondary header flag	1 bit	0x0800	0 = absent, 1 = present
application ID	11 bit	0x07FF	
Sequence	16 bit		packet sequence word
sequence flags	2 bit	0xC000	3 = complete packet
sequence count	14 bit	0x3FFF	
Length	16 bit		packet length word
length	16 bit	0xFFFF	(total packet length) - 7

c) Command secondary header(2 bytes)

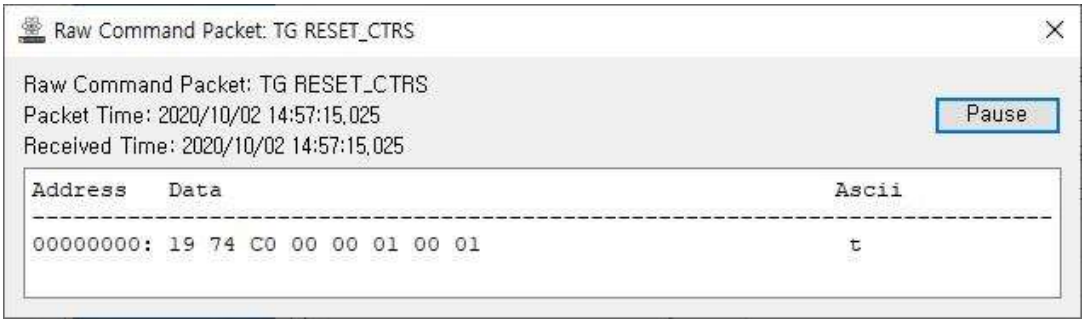
ID	size	offset	description
Command	16 bit		command secondary header
reserved	1 bit	0x8000	0으로 설정
command function code	7 bit	0x7F00	
checksum	8 bit	0x00FF	calculated by ground system

d) Telemetry secondary header(6 bytes)

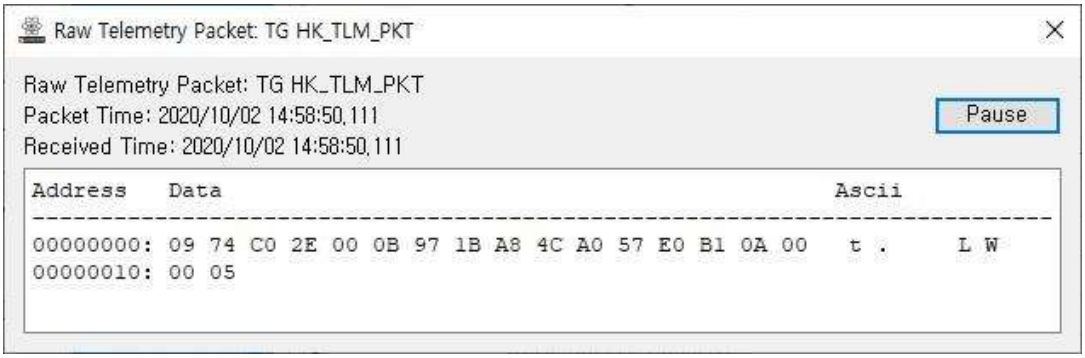
ID	size	offset	description
seconds	32 bit		
subseconds	16 bit		

cFS와 COSMOS를 연결하여 COSMOS의 Command Sender 프로그램을 통하여 “TG”

application의 reset command를 보내면 아래 그림 7의 a)와 같이 8 bytes command (packet primary header 6 bytes + command secondary header 2 bytes)가 전송되는 것을 확인할 수 있다. 위성에서 수신한 패킷은 b)와 같이 18 bytes telemetry (packet primary header 6 bytes + telemetry secondary header 6 bytes + user data field 6 bytes)를 확인할 수 있다.



a) Raw Command Packet



b) Raw Telemetry Packet

그림 7 COSMOS Raw Command/Telemetry Packet

본 논문에서는 프로토콜 header가 추가되기 전의 데이터와 추가된 데이터를 구분하기 위하여 패킷(packet)과 PDU(Protocol Data Unit)라는 용어를 사용한다. 패킷(packet)은 프로토콜 header가 추가되기 전의 command/telemetry 포맷이고, PDU는 패킷에 KISS or AX.25 프로토콜 header가 추가된 포맷이다.

ex) TG application Reset Command

패킷(packet) : 19 74 C0 00 00 01 00 01

PDU(KISS + AX.25 + packet) : C0 00 96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63
03 F0 19 74 DB DC 00 00 01 00 01 C0

D. Coupling

Coupling[16]은 서로 상호 작용하는 시스템들 사이의 의존성을 의미하며, 종속성이라고도 한다. Coupling 정도가 강할수록(tight) 서로 간에 복잡하게 얽혀 있어서 분석과 유지 보수가 어렵다. 일반적으로 느슨한 결합(loose coupling)은 의존성을 최소한으로 줄이는 구조를 의미하며, 이해하기 쉽고 테스트와 유지 보수가 수월하다. Coupling은 높은(tight) 수준에서 낮은(loose) 수준으로 여러 단계로 나눌 수 있고, 다음과 같다.

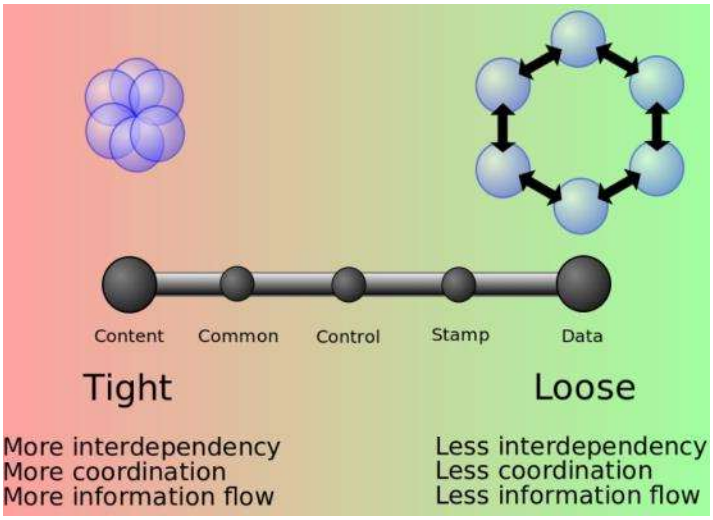


그림 8 Coupling의 개념적 모델
출처: https://en.wikipedia.org/wiki/Coupling_(computer_programming)

- Content coupling - 한 모듈이 다른 모듈의 코드를 직접적으로 참조하거나, 수정하는 경우이다. 이것은 기본 설계 개념인 정보 은닉에 위배된다.
- Common coupling - 여러 모듈이 동일한 전역 데이터를 참조하면서 발생한다. 모듈 사이에 공용 자료 영역이 추가되면서 프로그램의 복잡도를 증가시킨다.
- External coupling - 두 모듈이 외부적으로 부과된 데이터 형식, 통신 프로토

콜 또는 장치 인터페이스를 공유할 때 발생한다. 이것은 기본적으로 외부 도구 및 장치와의 통신과 관련이 있다.

- Control coupling - 수행할 작업에 대한 정보를 전달하여 다른 모듈의 흐름을 제어하는 하나의 모듈이다(예 : 수행할 작업 플래그 전달).
- Stamp coupling (data-structured coupling) - 모듈이 복합 데이터 구조를 공유하고 일부만 사용할 때 발생한다(예 : 하나의 필드만 필요한 함수에 전체 레코드 전달). 모듈이 필요하지 않으면 모듈이 레코드를 읽는 방식이 변경될 수 있다.
- Data coupling - 모듈 간에 같은 데이터 형식을 이용하여 자료를 교환하는 방식이다.
- Message coupling - 모듈끼리 자료를 교환하지 않고, 공용 interface를 통한 message 처리를 통해 상호 작용하는 방식이다.

Decoupling은 연결되거나 결합하지 않고 거래할 수 있는 둘 이상의 시스템이다. 시스템 간에 상호 작용하지 않으며 한 시스템은 다른 시스템에 대한 정보는 매우 제한적이며, 해당 정보는 일반적으로 공유 인터페이스에 관한 내용으로 제한된다. 분리된 시스템을 사용하면 다른 시스템에 영향을 주지 않고 한 시스템을 변경할 수 있으며, 다음과 같은 장단점을 가지고 있다.

- 장점
  - 각 구성 요소가 독립적으로 수행
  - 서비스를 수정해도 다른 서비스는 수정할 필요 없음
  - 코드 유지 관리 및 구현 변경 용이
  - 크로스 플랫폼, 다양한 언어 및 기술 사용 가능
  - 분산된 작업량 및 확장성

- 단점
  - 서비스 간에 추가적인 프로토콜 필요
  - 일관성 유지(데이터 동기화)에 문제가 발생할 수 있음

Ⅲ. 패킷 처리 프로그램

위성과 지상국 간 원활한 통신을 위해서는 관제 프로그램과 지상국 하드웨어(모뎀/송수신기)의 특성을 반영한 통신 프로토콜 처리가 필요하다. 이러한 통신 프로토콜 처리 과정은 지상국 하드웨어에 대한 의존성이 크기 때문에 하드웨어의 교체가 필요한 경우, 기존의 지상국 관제 프로그램에서 통신 프로토콜 처리를 하드웨어의 특성에 맞게 소스 코드를 수정하거나 재개발하여 사용해야 한다. 그림 9는 위성과 데이터 송수신을 담당하는 지상국 계층 구조를 도식화한 것이다.

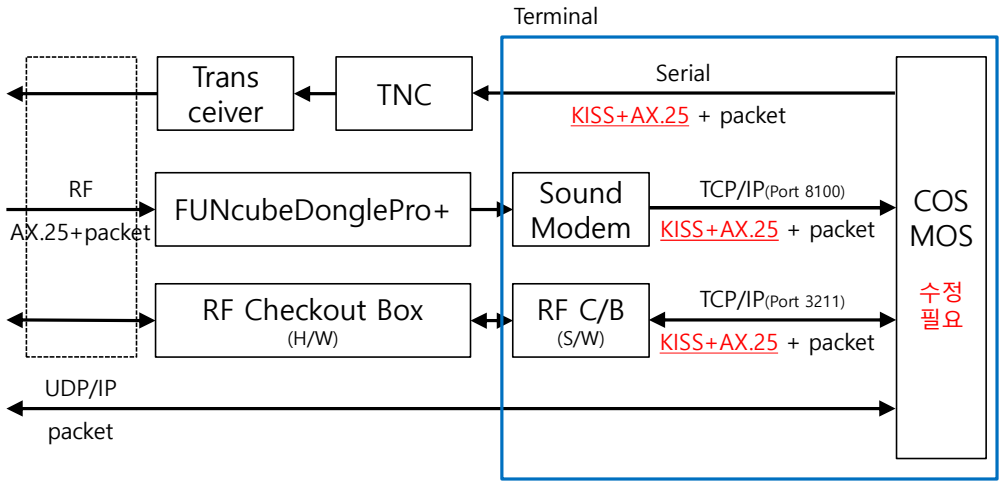
관제 프로그램	명령 전송, 위성 정보 수신 등을 통하여 위성을 운영하는 소프트웨어
프로토콜 처리	데이터 송수신을 위해 KISS, AX.25 등과 같은 프로토콜을 처리하는 프로그램
모뎀(변복조)	디지털과 아날로그 데이터를 서로 변복조를 담당하는 하드웨어나 소프트웨어
송수신기	위성과 데이터 송수신을 처리하는 장비

그림 9 지상국 계층 구조

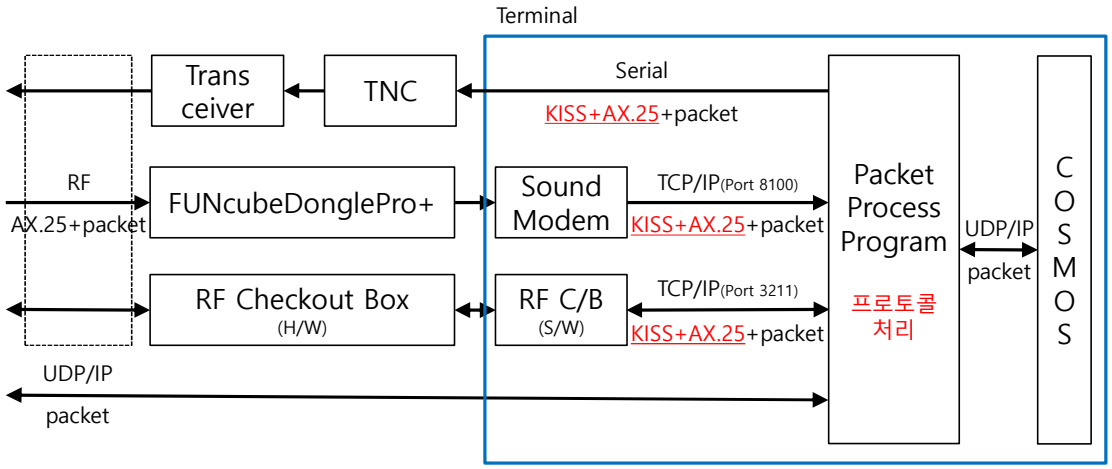
KMSL 큐브위성에 적용된 지상국을 보면, 그림 10에서 보는 것과 같이 개발, 시험 및 운영 환경에 따라서 송수신기/모뎀이 바뀌므로 통신 방식(이더넷, Serial)과 프로토콜(KISS, AX.25) 처리 여부가 변경된다. COSMOS 프로그램이 위성 운영을 위해 개발된 지상국 소프트웨어이지만, 위성과 패킷을 송수신하는데 필요한 통신 프로토콜(KISS, AX.25)은 지원하지 않는다. 프로토콜 처리를 위하여 COSMOS를 수정할 수도 있지만, 개발, 시험 및 운영 과정에서 문제가 발생 시 어느 부분에서 문제가 발생하였는지 파악하기 어렵고, 서비스 간의 종속성 증가로 환경 변경 시에 수정(통신 방식 변경, 프로토콜 처리 여부)이 필요하다.

본 연구에서는 소프트웨어 디커플링 기법을 적용하여 그림 10의 b)와 같이 다양한 지상국 장비(송수신기/모뎀)를 사용하여도 COSMOS를 수정하지 않고, 위성과 지상국 사이의 데이터 송수신 패킷 프로토콜을 처리하는 소프트웨어를 개발하고자 한다. COSMOS와 장비들 사이에서 시험 및 운영 환경에 따라 송수신 데이터(패킷) 프로토콜을 처리

해 주는 프로그램이므로 패킷 처리 프로그램(Packet process program)이라고 명칭을 정한다. 디커플링 기법을 검증하기 위한 패킷 처리 프로그램에서 필요한 사항을 정리하면 표 9와 같다.



a) COSMOS를 수정하여 프로토콜 처리



b) COSMOS를 수정하지 않고 프로토콜 처리

그림 10 KMSL 지상국 인터페이스

표 9 운영 환경에 따른 통신 방식 및 프로토콜 요구 사항

Device	통신 방식	IP Address	Port Up/Down	baud Rate	AX.25	KISS	KISS Port	COSMOS
OBC	UDP/IP	OBC	1234	-	×	-	-	가능
			1235					
Checkout Box(H/W)	TCP/IP	Checkout Box(S/W)	3211	-	○	○	1	AX.25, KISS Protocol 개발 필요
			3211					
FUNcube Dongle	TCP/IP	Sound Modem	-	-	-	-	-	
			8100	-	○	○	1	
TNC	SERIAL	-	COM?	9600	○	○	1 / 2	
			-	-	-	-	-	

※ IP Address는 OBC or 프로그램이 설치된 컴퓨터의 ip address임
※ TNC COM Port는 컴퓨터에서 인식하는 Port

A. 요구 사항

지상국 소프트웨어인 COSMOS와 위성, 지상국 하드웨어 및 소프트웨어와 연동을 위해 필요한 요구 사항은 크게 3가지로 구분할 수 있다. 첫째, 다양한 통신 방식을 지원해야 한다. 둘째, KISS 및 AX.25 프로토콜 처리를 지원해야 한다. 셋째, 수신 데이터 병합이 가능해야 한다.

- 통신 방식

표 10 통신 방식

No	Up/Down	통신 방식	Port	비고
1	Up	UDP	1234	같은 통신 방식에 다른 포트 사용
	Down		1235	
2	Up/Down	TCP	3211	같은 통신 방식과 포트 사용
3	Up	TCP	3211	같은 통신 방식에 다른 포트 사용
	Down		8100	
4	Up	Serial	COM	다른 통신 방식에 다른 포트 사용
	Down	TCP	8100	

위 표와 같이, 같은 통신 방식에서도 같은 포트를 사용하는 경우와 다른 포트를 사용하는 경우가 존재하고, 서로 다른 통신 방식을 사용하는 경우가 있어서 통신 방식과 포트를 별개로 설계하여야 한다.

- Protocol

표 11 패킷 처리 프로토콜

No	AX.25	KISS	KISS Port	비고
1	×	-	-	
2	○	○	Port 1 or 2	

KISS 프로토콜은 AX.25 프로토콜을 사용하지 않고 단독으로 사용하지는 않는다. 현재 사용하는 설정들에서는 AX.25와 KISS 프로토콜을 동시에 사용하고 있지만, 환경에 따라서는 AX.25 프로토콜만 사용할 수도 있도록 개발하여야 한다. 송신 시 KISS 프로토콜을 사용하는 경우, 포트에 따라서 헤더 데이터가 다르므로 포트를 설정할 수 있어야 한다.

- 데이터 병합

통신 보드를 사용하지 않고 UDP로 통신을 할 때는 패킷의 크기에 제한이 없지만, KMSL 큐브위성에 탑재된 통신 보드는 한 번에 송수신이 가능한 최대 패킷의 크기는 235 bytes이다. 위성과 송수신하는 한 개의 패킷이 235 bytes보다 크면, 송신 측에서는 나눠서 전송하고 수신 측에서는 분할되어 전송된 데이터를 병합하여 하나의 패킷으로 모으는 작업이 필요하다. 위성에서 수신된 데이터를 확인하여 분할된 패킷은 모든 패킷이 수신된 후 병합하여 COSMOS로 전송하는 기능을 지원하여야 한다.

B. 통신 처리

위성 또는 지상국 장비와 통신하기 위해서 그림 11과 같이 프로그램이 실행될 때 통신 방식(TCP, UDP, Serial)을 선택할 수 있도록 구성하고, 각 통신 방식에 따른 특성을 파악해야 한다. TCP와 UDP는 socket 방식이고, serial은 커널 오브젝트에 접근하기 위해서 부여된 번호인 handle을 통해서 제어한다.

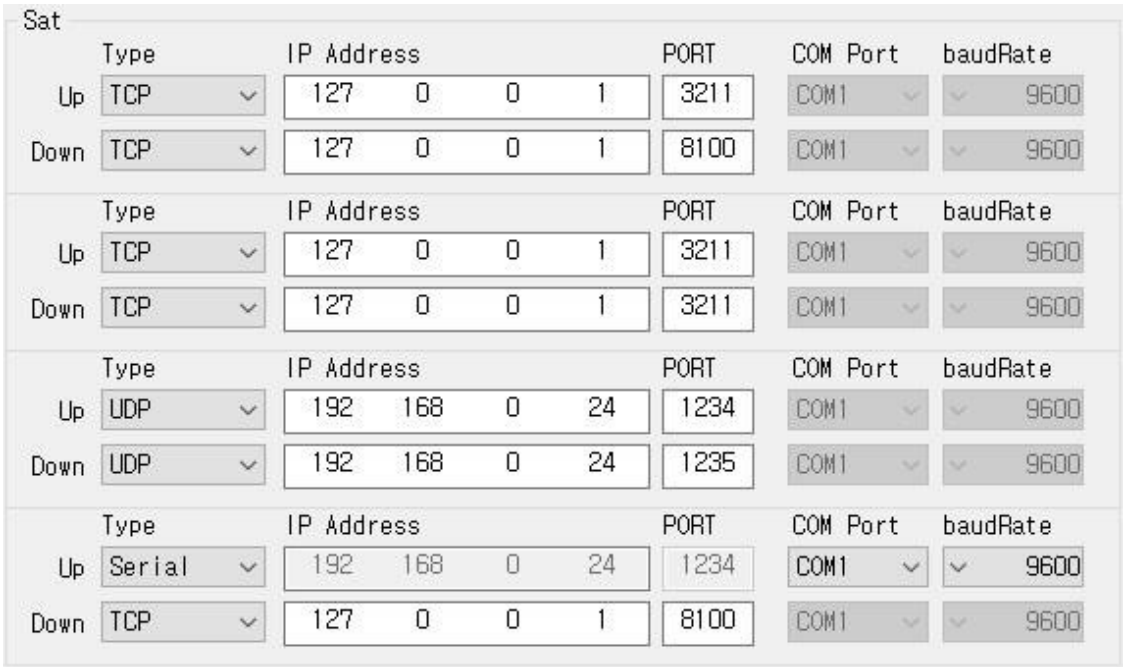


그림 11 Sat(Ground station) 설정

그림 12와 같이 TCP 방식은 server가 먼저 실행되어 연결 대기(listen)한 상태에서, client에서 server 연결에 필요한 IP와 Port 정보를 통하여 접속(connect)을 요청하면 수락(accept)하여 1:1로 연결지향 방식이다. 1:1로 통신을 하므로, 데이터를 송신(send)하고 수신(recv)할 때 상대방에 대해서 명시하지 않아도 된다. 지상국에서 사용하는 소프트웨어인 CheckoutBox program과 Soundmodem[17] program은 TCP server를 지원하므로, packet process program은 client 방식으로 개발한다.

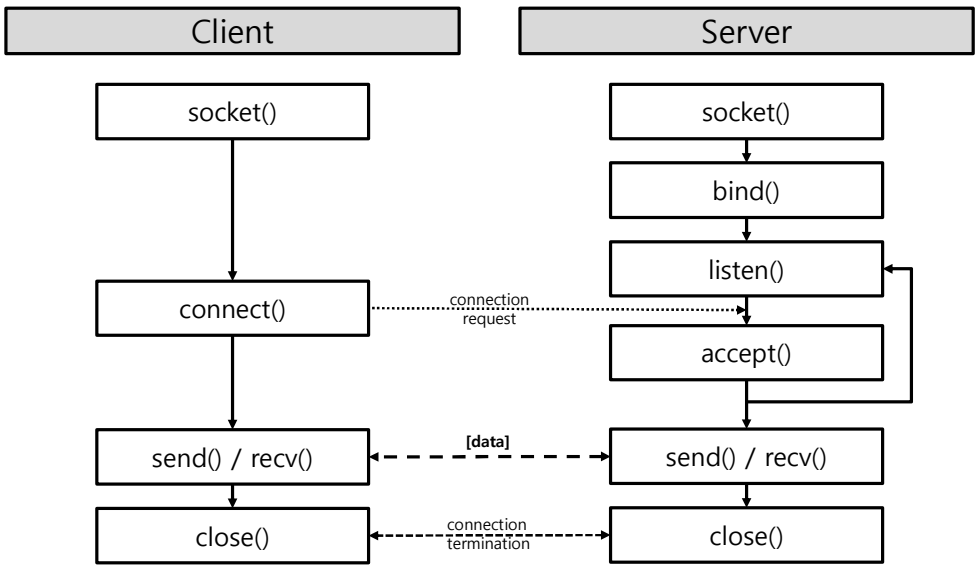
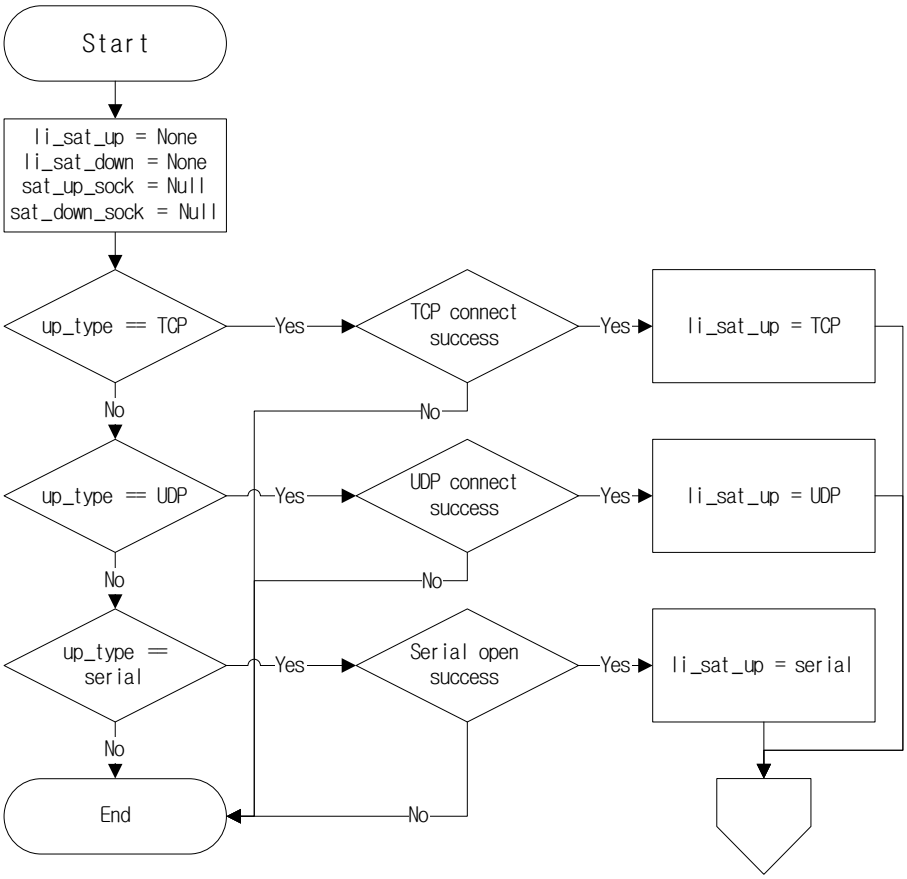


그림 12 TCP 통신

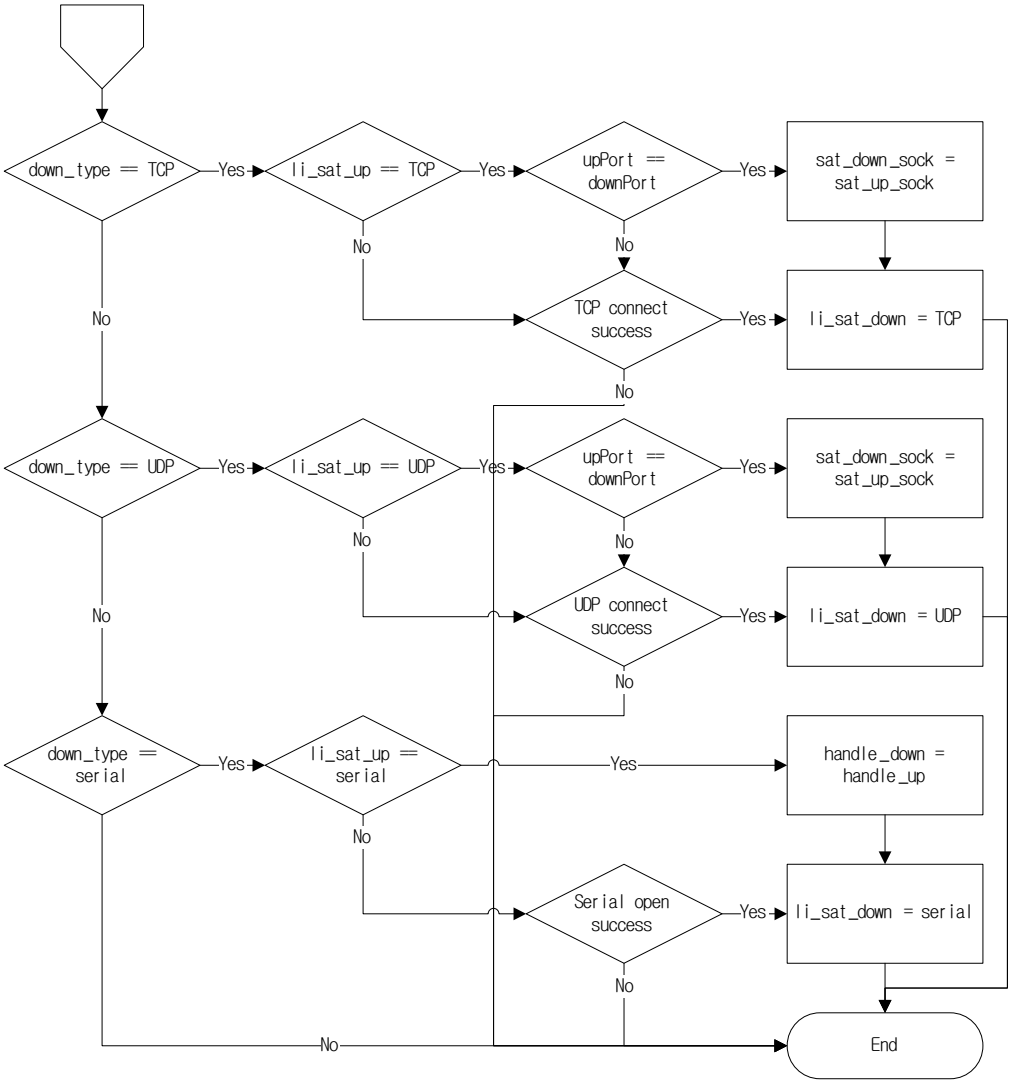
UDP는 TCP와 비슷하지만 비연결형 방식으로 1:1뿐만 아니라, 1:N, N:N으로 연결하여 통신이 가능하다. UDP는 비연결형 방식이므로 송신(sendto)과 수신(recvfrom)을 할 때마다 송수신할 대상에 대한 정보를 명시하여야 한다. UDP에서도 connect 함수를 지원하는데 TCP처럼 서버와 연결하는 것이 아니라, 송수신할 대상에 대해서 명시를 하는 것이다. 그러면 송수신을 할 때마다 상대에 대한 정보를 명시하지 않아도 되므로 소스코드가 더 간결해지고 성능 향상에도 도움이 된다.

패킷 처리 프로그램이 COSMOS와 통신 연결할 때는 송수신이 한 가지 방식으로 연결하므로 선택에 따라서 연결하면 된다. 지상국 장비들과 연결할 때는 통신 방식과 포트들이 같을 수도 있고 다를 수도 있다.

아래 그림 13 순서도에서 보는 것과 같이 uplink 연결 방식을 먼저 확인한다. 선택된 통신 방식에 따라서 연결하여 실패하면 종료하고, 성공하면 통신 방식을 변수에 저장한 다음 downlink 연결 방식을 처리한다. downlink 연결 방식 확인하면서 uplink 연결 방식과 비교하여 통신 방식과 포트가 같으면 uplink를 위해 연결하였던 socket (TCP/UDP) 또는 handle(Serial)을 복사하여 사용하고, 다르면 새로운 connection을 연결한다. Downlink 통신 연결 시 실패하면 uplink를 위해 연결하였던 socket 또는 handle을 해제하고 종료한다.



a) Uplink connect flowchart



b) Downlink connect flowchart

그림 13 통신 연결 순서도

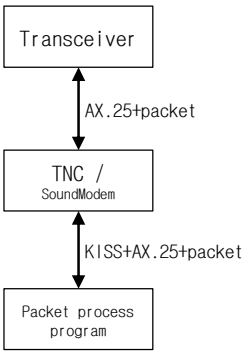
C. 프로토콜 처리

위성과 지상국 사이에 UDP로 직접 연결할 때는 별도의 프로토콜 추가 없이 패킷 데이터를 그대로 전송하면 된다. 하지만, RF를 통해서 송수신할 때는 AX.25, HDLC 또는 다른 프로토콜을 사용하고, 지상국 하드웨어 또는 모뎀 소프트웨어와 통신하면서 KISS 또는 다른 프로토콜을 사용한다. 본 논문에서는 KMSL 큐브위성에서 사용하는 AX.25와

KISS 프로토콜에 대해서만 다룬다.

프로토콜 구조는 2장에서 언급하였지만, 위성의 OBC(On-board computer)와 지상국 소프트웨어인 COSMOS 사이에서 하드웨어/소프트웨어가 추가되면서 패킷에 어떤 프로토콜의 header/tail이 추가되어 전송되는지 확인하여야 한다. 그림 14에서 보는 것과 같이 송수신기와 모뎀 사이에는 패킷에 AX.25 프로토콜 header가 모두 포함된다. 모뎀과 패킷 처리 프로그램 사이에는 패킷에서 KISS, AX.25 프로토콜 header가 포함되는데, AX.25 프로토콜에서 데이터를 검증하는 항목인 FCS(Frame Check Sequence)는 제외된다.

위성에서 수신된 PDU(Protocol Data Unit)는 모뎀에서 FCS를 검증하여 FCS를 제거한 후에 KISS 프로토콜을 추가하여 전송한다. 지상국에서 송신된 패킷은 모뎀에서 KISS 프로토콜을 제거하고, FCS를 추가하여 송신기로 전송한다. 지상국의 패킷 처리 프로그램에서는 모뎀과 KISS 프로토콜로 데이터를 송수신하므로 FCS를 포함하지 않고 처리하면 된다.



Address	Control	PID	packet	FCS
14/28 Bytes	1/2 Bytes	1 Byte	N Bytes	2 Bytes

Flag	Command	Address	Control	PID	packet	Flag
C0	1Byte	14/28 Bytes	1/2 Bytes	1 Byte	N Bytes	C0

그림 14 송수신 간 프로토콜 헤더

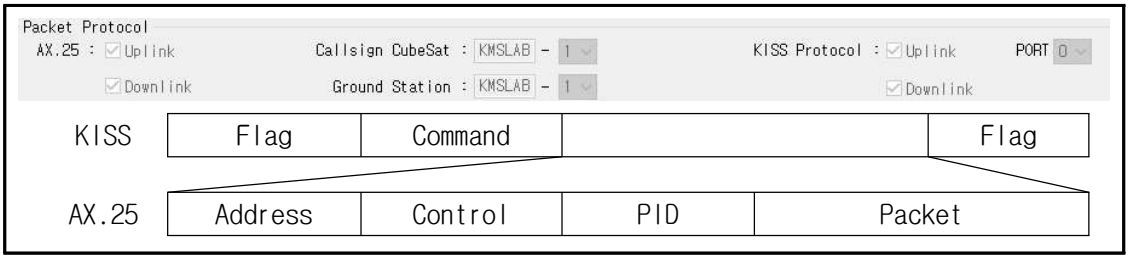


그림 15 프로토콜 Header/Tail

그림 14의 모뎀과 패킷 처리 프로그램 사이의 KISS/AX.25 프로토콜 header를 프로토콜별로 구분하면 그림 15와 같다.

그림 16에서 보는 것과 같이 모뎀에서 수신된 PDU가 존재하면 AX.25, KISS 프로토콜 사용 여부에 확인하여 header를 제거한 다음 COSMOS 프로그램에 패킷만 전송한다. 반대로 COSMOS에서 위성으로 송신하는 패킷에는 AX.25와 KISS header를 추가하여 모뎀에 전송한다.

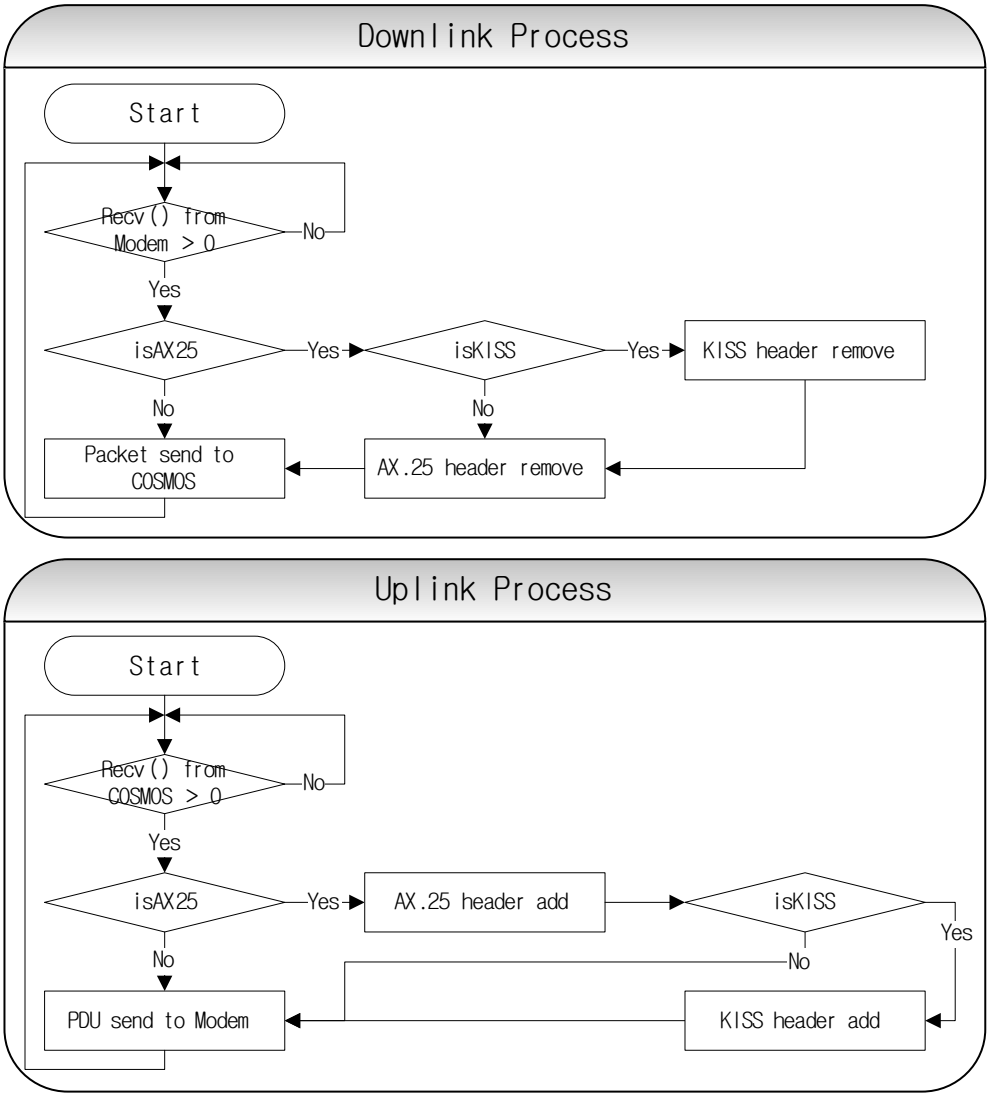


그림 16 Downlink/Uplink 프로토콜 처리

D. 데이터 병합

위성의 통신 보드에서 한 번에 송수신할 수 있는 패킷의 최대 크기가 제한되어 있다. 지상국 장비를 통해 수신한 PDU는 하나의 패킷을 여러 번에 분할되어 수신할 수도 있고, 지상국 장비에서 받은 여러 개의 PDU가 한 번에 패킷 처리 프로그램으로 전송될 수도 있다. 지상국 장비에서 수신한 PDU에 대한 순서도는 그림 17에 정리하였으며, 처리 방법은 다음과 같다.

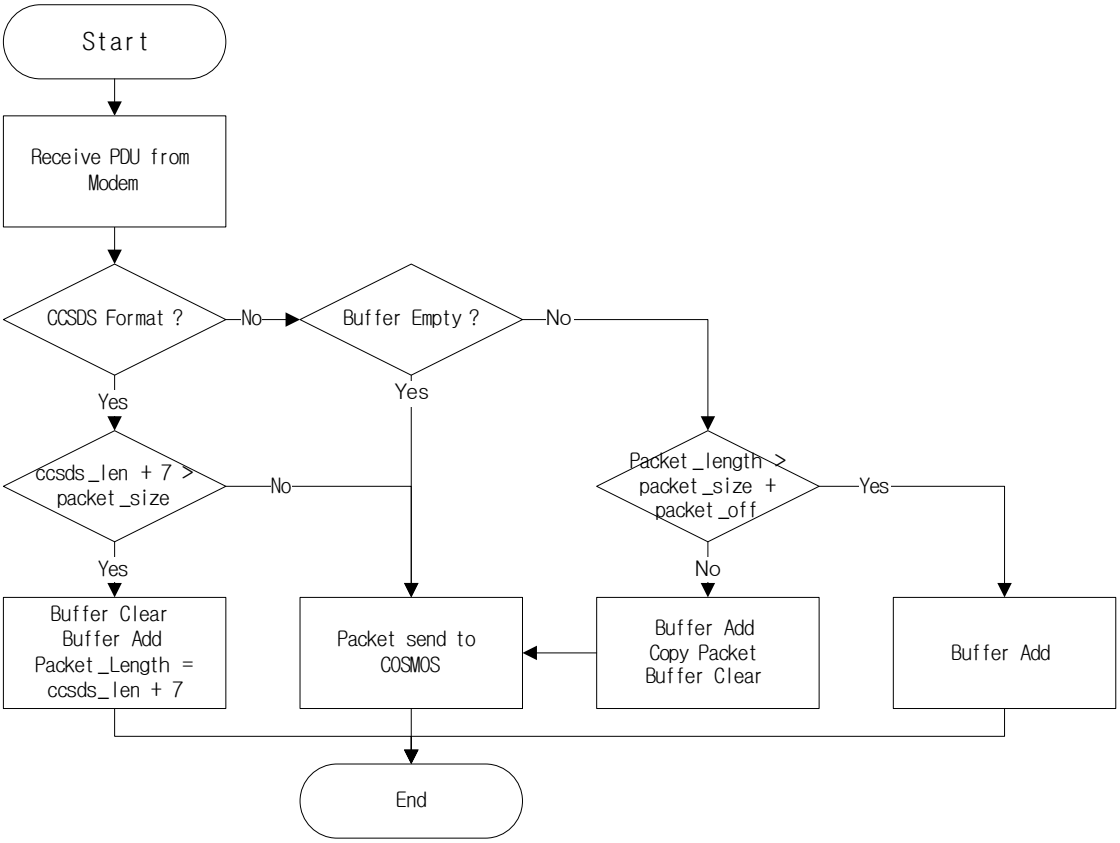


그림 17 패킷 병합 처리

1) 모뎀에서 수신한 PDU를 KISS flag(C0)를 이용하여 PDU를 구분하여 수신 버퍼에 저장
2) PDU가 표 8 CCSDS header의 규칙을 이용하여 CCSDS 포맷인지 확인
   가) CCSDS 형식의 패킷이면

CCSDS Length 필드를 이용하여 패킷의 크기와 수신한 패킷의 크기를 확인하여, 수신한 패킷의 크기가 같거나 크면 COSMOS에 전송한다. 수신한 패킷의 크기가 작으면, 추가로 수신하여야 하므로 임시 버퍼에 저장한다.

나) CCSDS 형식의 패킷이 아니면 기존에 수신한 패킷의 부족한 패킷이 추가로 수신된 데이터로 임시 버퍼에 미완성된 패킷이 존재하는지 확인한다.

a) 임시 버퍼가 비어있으면, 패킷 크기를 확인할 수 없으므로 COSMOS에 전송한다.

b) 임시 버퍼에 미완성된 패킷이 있으면

임시 버퍼에 데이터를 추가하고, 패킷이 완성되었으면 COSMOS에 전송하고 임시 버퍼를 삭제한다.

Ⅳ. 패킷 처리 프로그램 검증과 분석

패킷 처리 프로그램의 검증을 위한 테스트 항목은 표 12에 정리하였으며 절차는 다음과 같다.

- 테스트 절차
  ① 지상국에서 위성에 상태 정보 전송 시작 명령(TO ADD_HK_TLM_PKTS)을 송신
  ② 위성에서 5개 application(ADCS, EPS, FI, TG, TRX)의 HK_TLM_PKT 패킷을 10초 간격으로 전송
  ③ 지상국에서 전송 종료 명령(TO REMOVE_HK_TLM_PKTS)을 송신
  ④ 위성에서 전송을 종료

- 테스트 항목

표 12 테스트 항목

No	위성통신	지상국 장비	Up / down 구분	통신방식	패킷 처리 프로그램	COSMOS / Other
1	Ethernet	-	up / down	UDP	×	COSMOS
2		-	up / down	UDP	○	COSMOS
3	RF	Checkout Box	up / down	TCP	○	COSMOS
4		Checkout Box	up	TCP	○	COSMOS
		FUNcubeDongle	down	TCP		
5		TNC	up	Serial	○	COSMOS
		FUNcubeDongle	down	TCP		
6		Checkout Box	up / down	TCP	○	Other

패킷 처리 프로그램을 사용하지 않고 cFS와 COSMOS에서 직접 통신이 되는지 검증을 하고, 패킷 처리 프로그램을 사용하여 다양한 조건에서 테스트를 진행하여 패킷이 정상적으로 송수신되는지 검증한다.

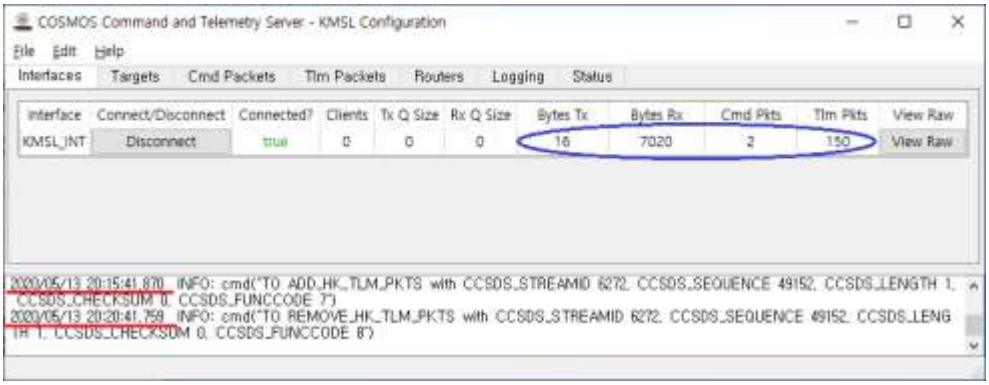
위성과 송수신하는 명령과 원격 측정 데이터 크기는 아래 표 13과 같다.

표 13 Command/Telemetry 패킷 크기

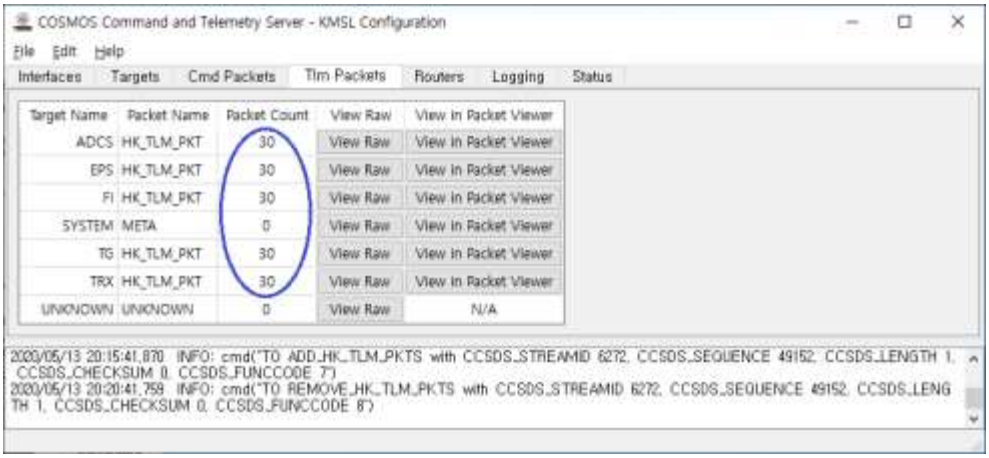
Tx/Rx	Application	Packet Name	Packet Size
Tx	TO	ADD_HK_TLM_PKTS	8 bytes
		REMOVE_HK_TLM_PKTS	8 bytes
Rx	ADCS	HK_TLM_PKT	50 bytes
	EPS	HK_TLM_PKT	94 bytes
	FI	HK_TLM_PKT	24 bytes
	TG	HK_TLM_PKT	18 bytes
	TRX	HK_TLM_PKT	48 bytes

A. cFS/COSMOS 직접 송수신

송수신 패킷의 검증을 위하여 위성(cFS)과 지상국(COSMOS)을 UDP/IP를 통하여 테스트를 진행한다. 검증은 COSMOS에서 시작 명령으로 전송이 시작하고, 종료 명령으로 전송이 종료되는지를 확인하여 송신을 확인한다. 수신은 COSMOS Command and Telemetry Server Tool의 Tlm Packets 탭과 Interfaces 탭에서 송수신된 패킷의 개수와 바이트를 확인한다. 그림 18과 같이 5분 동안 테스트를 진행한 결과를 보면, 수신(Rx)은 150개 7,020 bytes, 송신(Tx)은 2개 16 bytes를 송수신된 것을 확인할 수 있다. cFS와 COSMOS 사이에 패킷이 정상적으로 송수신이 되는 것을 확인하였다.



a) Interfaces Tab

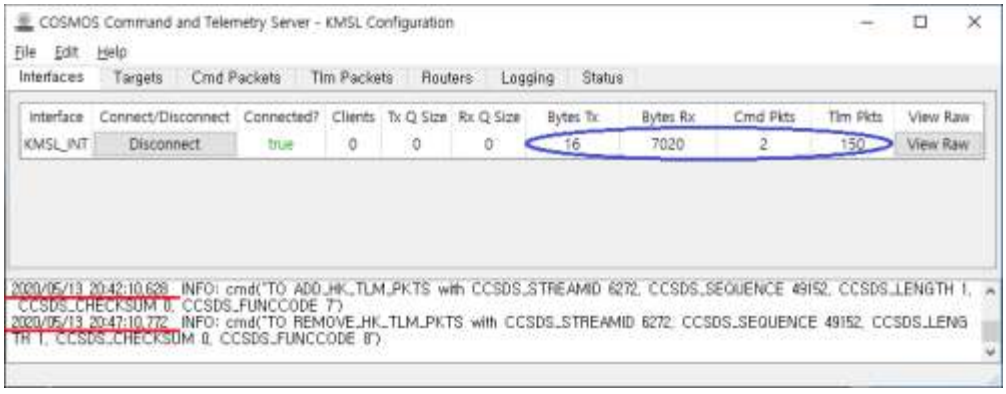


b) Tlm Packets Tab

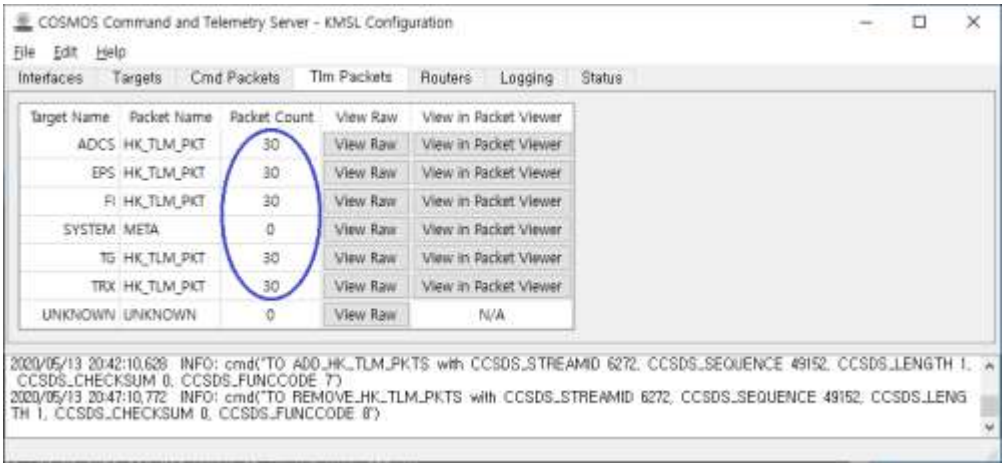
그림 18 cFS/COSMOS 직접 송수신 결과

B. UDP를 통한 프로그램 검증

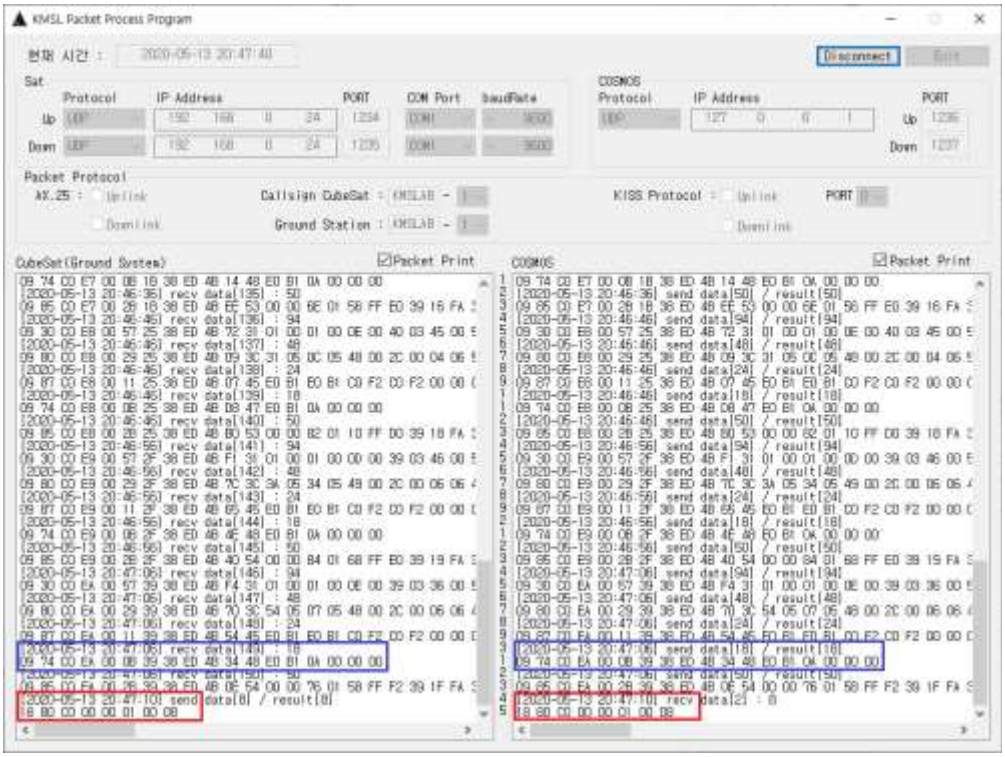
위성과 지상국 소프트웨어 사이에 패킷 처리 프로그램을 포함하여 송수신을 확인한다. 그림 19의 c)를 통하여 위성에서 수신한 18 bytes 패킷을 별도의 처리 없이 지상국으로 전송하고, 지상국으로부터 받은 8 bytes 패킷을 위성에 전송한다. 위성과 지상국 사이에 통신은 모두 UDP 방식으로 처리하여 별도의 프로토콜을 추가하지 않아도 송수신되고, 그림 19의 a), b)에서 보는 것과 같이 UDP 방식에서는 패킷의 손실이 없는 것을 확인하였다.



a) Interfaces Tab



b) Tlm Packets Tab



c) UDP Packet program screen

그림 19 UDP를 통한 패킷 처리 프로그램 결과

C. RF 통신을 위한 프로그램 검증

1. Uplink Process

Uplink를 위해 지상에서 위성으로 보내는 명령(TO REMOVE_HK_TLM_PKTS : 18 80 C0 00 00 01 00 08)은 다음과 같은 과정을 거치면서 패킷이 수정되어 전송된다. 그림 21에 지상에서 보내는 명령이 어떻게 변경이 되는지 도식하였다.

① COSMOS에서 패킷 처리 프로그램에게 패킷 전송
② 패킷 처리 프로그램은 AX.25 header 추가
③ 패킷 처리 프로그램은 KISS header 추가
④ header가 추가된 PDU를 모뎀(TNC, CheckoutBox Software)에 전송
⑤ 모뎀은 KISS header를 제거하고, AX.25 FCS 추가
⑥ 하드웨어(Transceiver, CheckoutBox Hardware)를 통하여 위성에 PDU 전송

패킷 처리 프로그램은 먼저 AX.25 header(Address, Control, PID)를 추가한다. Callsign은 TCP/IP의 IP address와 비슷하며, 각각 무선국을 구분하는 고유의 부호이다. 그림 20과 같이 AX.25 address는 위성과 지상국의 callsign과 ssid를 변환하여 생성하는데 다음과 같다.

① callsign (KMSLAB)과 ssid(1)의 ascii code를 왼쪽으로 1bit 쉬프트한다.
② 쉬프트된 값에서 ssid값에 대해서는 목적지는 0x80을, 출발지는 0x01을 Bit OR 연산한다.

	Destination							Source							비고
문자	K	M	S	L	A	B	1	K	M	S	L	A	B	1	callsign + SSID
HEX	4B	4D	53	4C	41	42	31	4B	4D	53	4C	41	42	31	
Bit shift	96	9A	A6	98	82	84	62	96	9A	A6	98	82	84	62	Left 1bit shift
SSID Bit OR	96	9A	A6	98	82	84	E2	96	9A	A6	98	82	84	63	Destination(0x80) Source(0x01)

그림 20 AX.25 address

위 과정을 거쳐 callsign(KMSLAB-1)의 목적지와 출발지의 address는 “96 9A A6 98 82 84 E2”, “96 9A A6 98 82 84 63”와 같이 생성되고, control(03)과 PID(F0)를 추가하면 AX.25 header는 완성된다.

AX.25 + Packet	Address	Control	PID	Packet	FCS
	96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63	03	F0	18 80 C0 00 00 01 00 08	C6 43

⬆

KISS + AX.25 + Packet	Flag	Command	Address	Control	PID	Packet	Flag
	C0	00	96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63	03	F0	18 80 DB DC 00 00 01 00 08	C0

⬆

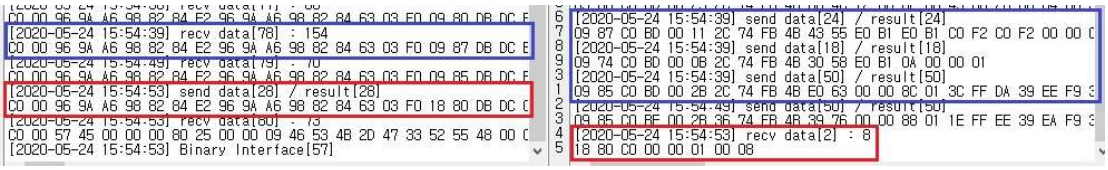
Packet	Packet (TO REMOVE_HK_TLM_PKTS)
	18 80 C0 00 00 01 00 08

그림 21 Uplink 패킷 데이터

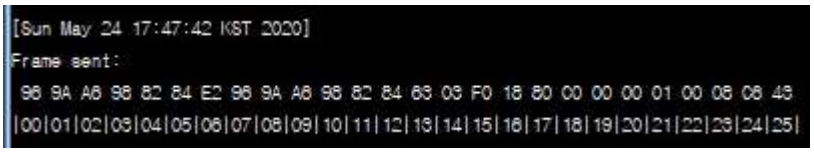
KISS header는 PDU의 시작과 끝을 나타내는 flag ‘C0’와 command로 구성된다. Command는 4 bits port와 4 bits command로 구성되는데, port ‘0’과 command “data frame” 값(0)으로 command는 “00”값을 가진다. 전송하는 PDU의 내용에 KISS protocol의 escape 문자(C0, DB)가 있으면 KISS header와 구분하기 위해서, ‘C0’는 ‘DB DC’로, ‘DB’는 ‘DB DD’로 변경한다. 그림 22 a)의 빨간색 상자를 보면, 명령의 3번째 바이트가 ‘C0’이므로 ‘DB DC’로 변경된 것을 확인할 수 있다.

CheckoutBox 소프트웨어는 KISS header를 제거하고, 변경된 KISS escape 문자를 원래 문자로 변경한다. AX.25 header와 패킷으로 FCS(Frame Check Sequence)를 추가한 PDU를 송신기를 통해 전송한다.

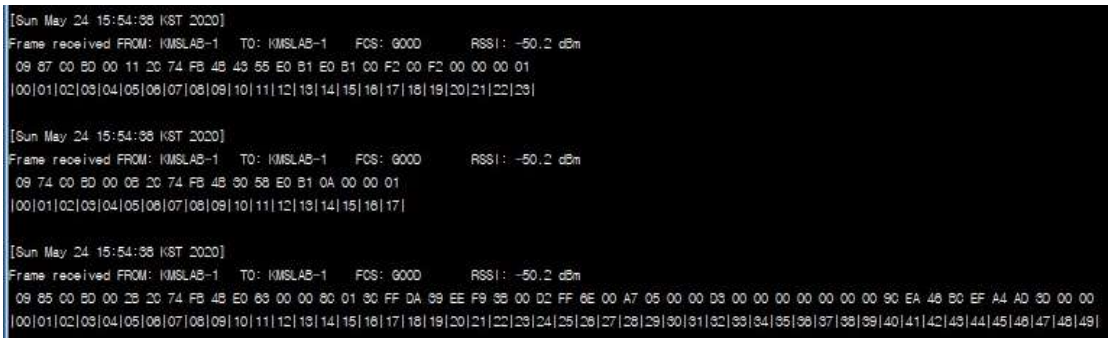
그림 22의 b)와 같이 지상에서 보낸 명령을 위성에서 수신하여 위성의 상태 정보를 c), d)와 같이 전송하는 것을 확인할 수 있다.



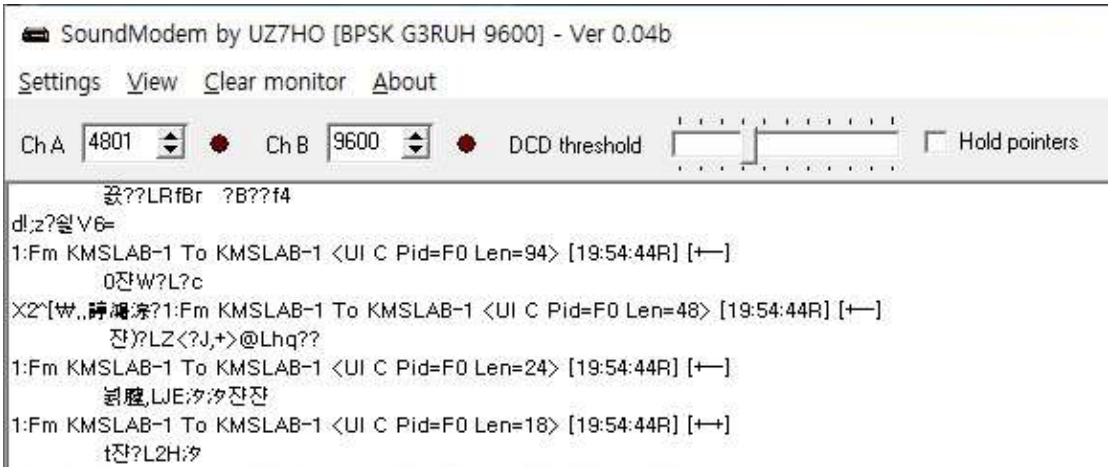
a) Packet process program



b) RF CheckoutBox TX



c) RF CheckoutBox RX



d) SoundModem RX

그림 22 RF를 통한 시험 screenshot

2. Downlink Process

Downlink를 위해 위성에서 지상으로 보내는 PDU는 다음과 같은 과정을 거치면서 전송된다. 그림 23에 위성으로부터 수신한 PDU가 어떻게 변경이 되는지 도식하였다.

① 모뎀은 AX.25 FCS를 통하여 PDU 검사
② FCS를 제거하고, KISS header 추가하여 전송
③ 패킷 처리 프로그램은 KISS, AX.25 header를 제거
④ CCSDS length field의 값과 패킷의 길이를 확인하여 일치하면 COSMOS에게 전송

모뎀은 RF를 통해 수신된 PDU를 FCS 항목과 확인하여 오류가 없는 PDU는 그림 22의 c)와 같이 callsign과 패킷을 내용을 확인할 수 있다. AX.25 FCS 항목을 제거하고, KISS header를 추가한 PDU를 패킷 처리 프로그램에 전송한다.

패킷 처리 프로그램은 수신한 PDU를 KISS flag를 이용하여 PDU를 구분하고, PDU에서 KISS header를 먼저 제거한다. AX.25 header의 callsign과 지상국 callsign을 비교하여, 다른 지상국으로 전송될 PDU는 버리고, 정상적으로 수신된 PDU는 AX.25 header를 제거한다. CCSDS length 필드의 값과 전체 패킷의 길이를 비교하여, 패킷의 길이가 짧으면 임시 버퍼에 저장하여 추가로 수신된 패킷을 확인하여 정상적인 패킷을 COSMOS에 전송한다.

패킷 처리 프로그램에서 받은 PDU는 위성에서 RF를 통해 모뎀에서 수신한 PDU와 1:1로 받을 수도 있고, N개의 PDU가 한 번에 전송이 될 수도 있다. 그림 22의 c)와 같이 24, 18, 50 bytes로 구성된 3개의 패킷이 a)의 파란색 상자와 같이 154 bytes 하나의 PDU로 받은 것을 확인할 수 있다. 그림 23과 같이 KISS flag(C0)를 기준으로 46, 38, 70 bytes PDU 3개로 나누어지는 것을 확인할 수 있다.

154 bytes 데이터에서 46 bytes만 처리하고 나머지 108 bytes는 수신 버퍼에 남겨둔다. KISS header를 제거하고 escape 문자에 대해서 처리하면, “96 9A A6 98 82 84 E2 ..... C0 F2 00 00 00 01”의 40 bytes PDU로 변경이 된다. 목적지(지상국) callsign이 “96 9A A6 98 82 84 E2”이면 KMSLAB-1으로 지상국 callsign과 일치하므로, AX.25 header를 제거하여 “09 87 C0 BD 00 11 ... 00 01”의 24 bytes 패킷만 남는다. 5~6번째 bytes가 CCSDS Length 필드로 “00 11”이면 17이다. 패킷의 길이 24에 7을 빼면

17(24 - 7)로 정상적으로 처리된 패킷이므로 그림 22의 a)와 같이 COSMOS로 전송한다. 버퍼에 남겨놓은 나머지 108 bytes의 데이터도 반복적으로 처리하여 전송한다.

Receive PDU 154(46 + 38 + 70) bytes
C0 00 96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63 03 F0 09 87 DB DC BD 00 11 2C 74 FB 4B 43 55 E0 B1 E0 B1 DB DC F2 DB DC F2 00 00 00 01 C0
C0 00 96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63 03 F0 09 74 DB DC BD 00 0B 2C 74 FB 4B 30 58 E0 B1 0A 00 00 01 C0
C0 00 96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63 03 F0 09 85 DB DC BD 00 2B 2C 74 FB 4B E0 63 00 00 8C 01 3C FF DA 39 EE F9 3B 00 D2 FF 6E 00 A7 05 00 00 D3 00 00 00 00 00 00 00 9C EA 46 BC EF A4 AD 3D 00 00 C0

	Flag	Command	Address	Control	PID	Packet	Flag
KISS + AX.25 + Packet	C0	00	96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63	03	F0	09 87 DB DC BD 00 11 2C 74 FB 4B 43 55 E0 B1 E0 B1 DB DC F2 DB DC F2 00 00 00 01	C0
	C0	00	96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63	03	F0	09 74 DB DC BD 00 0B 2C 74 FB 4B 30 58 E0 B1 0A 00 00 01	C0
	C0	00	96 9A A6 98 82 84 E2 96 9A A6 98 82 84 63	03	F0	09 85 DB DC BD 00 2B 2C 74 FB 4B E0 63 00 00 8C 01 3C FF DA 39 EE F9 3B 00 D2 FF 6E 00 A7 05 00 00 D3 00 00 00 00 00 00 00 9C EA 46 BC EF A4 AD 3D 00 00	C0

⬇

	Packet (HK_TLM_PKTS)	
Packet	09 87 C0 BD 00 11 2C 74 FB 4B 43 55 E0 B1 E0 B1 C0 F2 C0 F2 00 00 00 01	FI (24 bytes)
	09 74 C0 BD 00 0B 2C 74 FB 4B 30 58 E0 B1 0A 00 00 01	TG (18 bytes)
	09 85 C0 BD 00 2B 2C 74 FB 4B E0 63 00 00 8C 01 3C FF DA 39 EE F9 3B 00 D2 FF 6E 00 A7 05 00 00 D3 00 00 00 00 00 00 00 9C EA 46 BC EF A4 AD 3D 00 00	ADCS (50 bytes)

그림 23 Downlink 패킷 데이터

그림 22와 같이 CheckoutBox를 통해서 송신하고, CheckoutBox와 SoundModem을 통해서 수신되는 것을 확인하였다.

3. 패킷 병합

궤도에서 운영 중인 위성의 통신 보드는 한 번에 전송 가능한 패킷의 크기는 제한이 있다. 위성에서 보내는 패킷의 크기는 다양하여 한 번에 전송이 가능한 것도 있지만, 분할하여 여러 번에 전송할 수도 있다. COSMOS에서는 수신된 패킷에서 CCSDS header의 패킷 크기와 수신한 패킷의 크기만 비교한다. COSMOS는 한 번에 수신한 패킷의 정보를 보여주는 기능만 있고, 분할되어 수신된 데이터를 합쳐서 출력하는 기능은 없다. 그러므로 패킷 처리 프로그램에서 분할되어 수신된 데이터를 합쳐서 한 개의 패킷을 COSMOS에 전송할 필요가 있다.

송수신 테스트를 위한 패킷은 235 bytes보다 큰 패킷이 없어 별도의 위성 정보를 수신하여 테스트한다. 그림 24에서 보는 것과 같이 1,356 bytes 크기의 패킷을 전송하였으며, 1,356 bytes는 6번에 나누어서 전송된다.

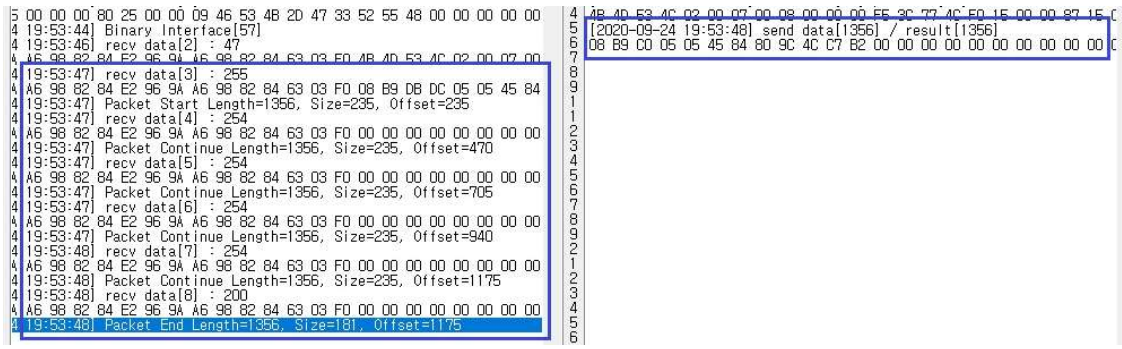


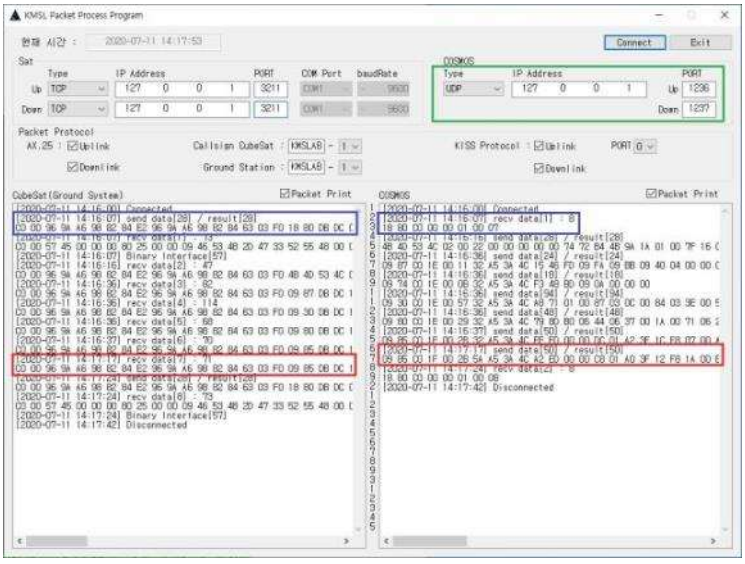
그림 24 패킷 병합 결과 screenshot

패킷 처리 프로그램은 위성으로부터 수신한 PDU에서 프로토콜(KISS, AX.25)을 제거한 후, CCSDS header 정보를 통해 패킷의 길이를 확인한다. 그림 24와 같이 수신해야 할 패킷의 CCSDS length 값은 “05 45”이면 1,349이다. 1,349에 7을 더하면 패킷의 길이는 1,356 bytes이다. 수신한 패킷의 크기는 235 bytes이므로 패킷을 임시 버퍼에 저장하고, CCSDS length 값(1,356)과 임시 버퍼에 저장된 패킷의 offset 값(235)을 저장한다. 두 번째 수신한 패킷을 확인하여 CCSDS 포맷이 아니면, 임시 버퍼에 추가하고 offset 값에 수신한 패킷의 크기를 더한다. 임시 버퍼의 패킷이 CCSDS length 값과 같으면 COSMOS에 패킷을 전송하고, 임시 버퍼를 초기화한다. 이 과정을 반복하여 그림 24와 같이 1,356 bytes 크기의 패킷이 COSMOS에 전송이 되는 것을 확인하였다.

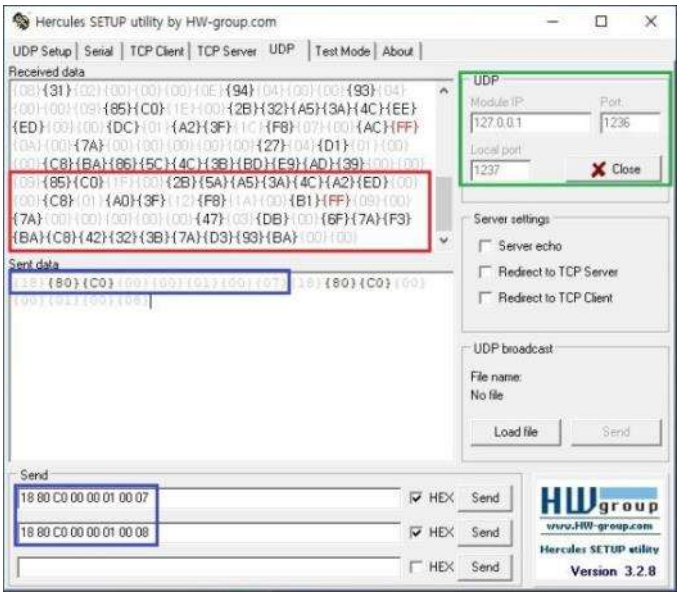
4. 다른 지상국 프로그램(Hercules)

KMSL 큐브위성을 운영하기 위해 지상국 소프트웨어로 COSMOS를 사용하지만, 다른 위성을 운영하면서 지상국 소프트웨어가 변경될 수도 있다. 테스트를 위해 다른 지상국 소프트웨어를 사용하지 않고, hex 값을 전송하고 수신할 수 있는 hercules라는 프로그램을 사용하여 검증하였다.

그림 25 a)의 녹색 상자와 같이 패킷 처리 프로그램은 COSMOS를 위한 설정을 그대로 사용한다. 그림 24의 파란색 상자와 같이 hercules 프로그램에서 보낸 명령(18 80 C0 00 00 01 00 07)이 패킷 처리 프로그램을 거쳐 위성에 전송이 되고, 위성에서 상태 정보가 전송되기 시작한다. 위성에서 보낸 위성 정보는 빨간색 상자와 같이 패킷 처리 프로그램을 거쳐 hercules에서 hex 값으로 출력되는 것을 확인하였다. Hercules 프로그램은 지상국 소프트웨어가 아니어서, hex 값이 어떤 정보인지는 파악할 수는 없다. 하지만, COSMOS가 아닌 다른 소프트웨어에서 명령이 송신되고, 위성 정보가 수신되는 것을 통하여 패킷 처리 프로그램 수정 없이 재사용이 가능한 것을 확인하였다.



a) Packet process program



b) Hercules program

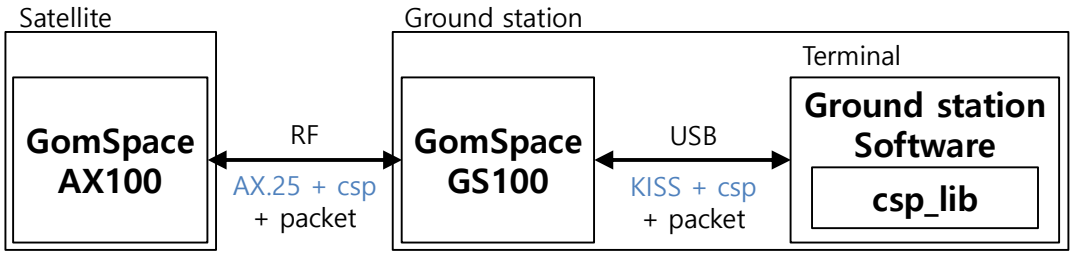
그림 25 프로그램을 이용한 테스트 결과 screenshot

D. 지상국 구성 비교

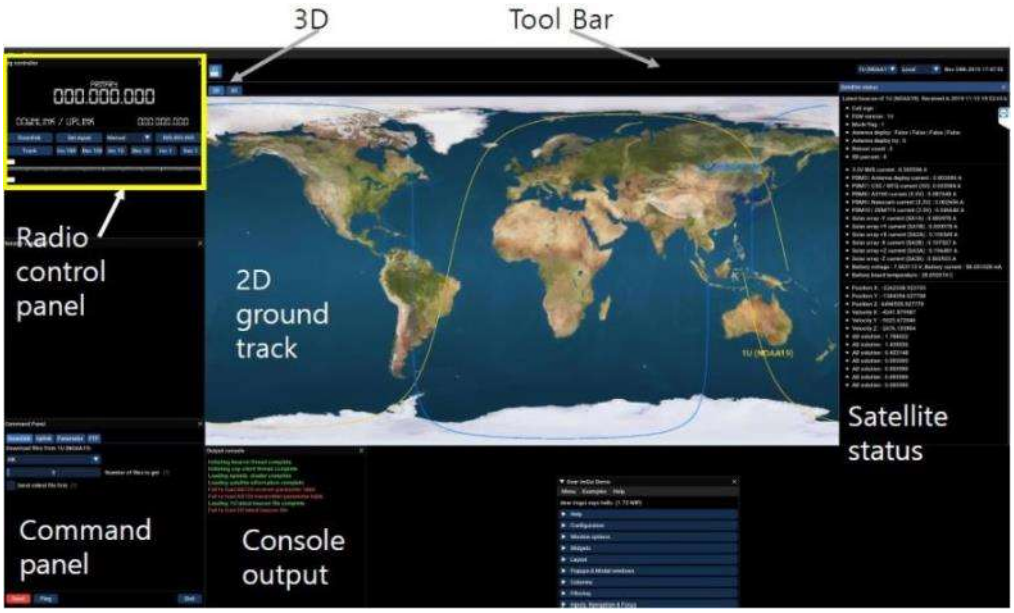
조선대 KMSL과 연세대 Canyval-C 위성 인터페이스는 표 14에 정리하였고, KMSL과 Canyval-C 위성은 동일한 비행 소프트웨어(cFS)와 프로토콜(AX.25)을 사용하고 있다. KMSL 지상국은 각 계층을 분리(decoupled)하여 개발하였고, Canyval-C 지상국은 그림 26에서 보는 것과 같이 csp_lib를 이용하여 위성과 송수신 패킷을 처리하고, 모든 기능(주파수 설정, 위성 관제 등)을 하나의 소프트웨어로 구성되도록 자체 개발하였다.

표 14 KMSL과 Canyval-C 위성 인터페이스

대학		조선대학교	연세대학교
위성명		KMSL	Canyval-C
위성	비행 소프트웨어	cFS	cFS
	통신보드	ISIS TRXVU	GomSpace NanoCom AX100
	프로토콜	AX.25	AX.25, CSP
지상국 통신 장비		IC-9100, CheckoutBox 등등	GomSpace NanoCom GS100



a) Canyval-C 통신 인터페이스



b) Canyval-C 지상국 소프트웨어 (연세대학교 제공)

그림 26 연세대학교 Canyval-C 구성

KMSL과 Canyval-C 지상국 통신 인터페이스를 그림 27에 도식하였다. 그림에서 보는 것과 같이, 계층화로 분리된 KMSL 지상국과 다르게, Canyval-C 지상국은 송수신기/모뎀이 하나의 장비(Gomspace GS100)에서 담당하고, 프로토콜 처리는 csp_lib를 이용하여 자체 개발한 지상국 소프트웨어에서 처리한다.

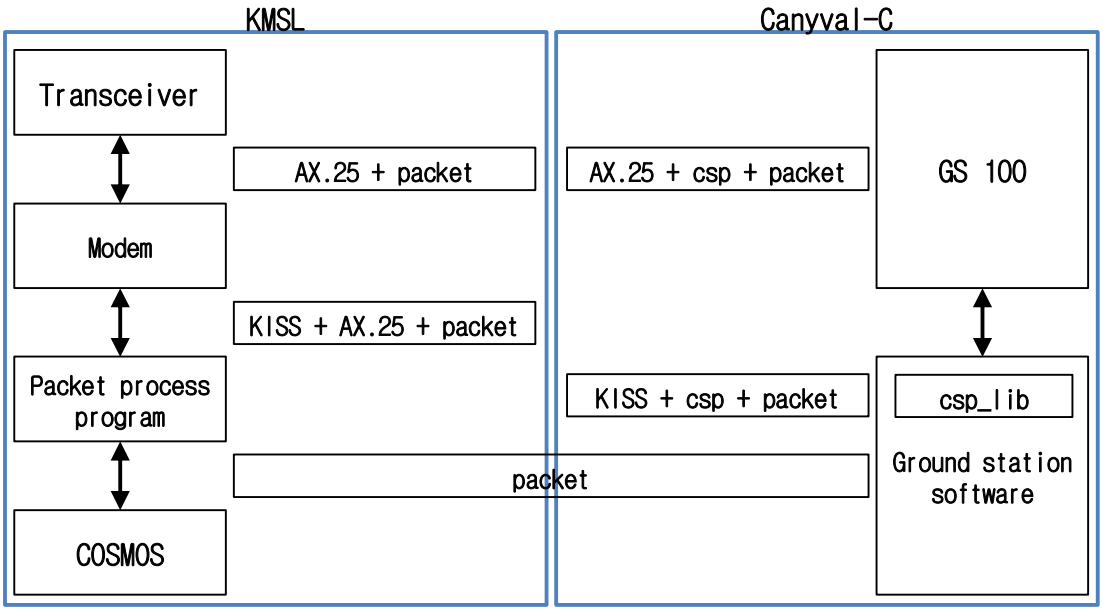


그림 27 KMSL/Canyval-C 지상국 통신 인터페이스 비교

Canyval-C 지상국은 GomSpace GS100 송수신기와 같이 csp_lib를 지원하는 장비에만 적용이 가능하다. 지상국 송수신기가 고장이 발생하거나 다른 장비로 교체하면서 csp_lib를 지원하지 않는 장비를 사용하게 되면, 패킷 처리가 불가능하여 지상국 소프트웨어 전체를 사용할 수 없게 되어 위성 운영이 불가능하다. KMSL 지상국의 경우에는 계층간 분리(decoupling)되어, 다양한 지상국 송수신기와 모뎀을 지원하고 관제 프로그램이 변경이 되더라도 수정없이 사용이 가능한 것을 확인하였다. 두 지상국의 패킷 처리 방식에 대해서는 표 15에 정리하였다.

표 15 KMSL과 Canyval-C 패킷 처리 방식 비교

구분	KMSL	Canyval-C
개발 기법	decoupled	tightly-coupled
확장성	다양한 통신 장비 지원	통신 장비 제한
재사용성	용이함	제한적
기술 추가	다양한 기술 적용 가능	제한적

위성 개발을 한 번으로 끝나면, 개발 중인 플랫폼만 고려하면 된다. 하지만, 위성 개발을 계속 진행하기 위해서는 개발 비용과 기간 단축을 고려하지 않을 수 없다. 개발 비용과 기간을 단축하기 위해서는 기존에 사용하는 플랫폼의 재사용이 중요하며, 소스 코드가 단순하여 유지 보수 및 수정이 쉬워야 한다.

Ⅴ. 결론

궤도에서 운영 중인 위성을 관제하기 위해서는 지상국이 필요하다. 지상국은 위성과 통신을 위한 안테나, 증폭기, 트랜시버와 같은 통신 하드웨어와 소프트웨어로 구성된다. 지상국 소프트웨어는 통신 하드웨어를 제어하는 프로그램, 명령을 전송하고 위성 정보를 수신하는 프로그램들이 있다. 조선대학교에서 개발하고 있는 KMSL 큐브위성을 관제하기 위해 적용된 지상국 소프트웨어는 오픈 소스 프로그램인 COSMOS이다. COSMOS는 지상국 운영에 필요한 핵심적인 기능(command sender, packet viewer, telemetry viewer, telemetry grapher, data viewer 등)과 다양한 통신 방식(TCP, UDP, Serial 등)을 지원한다. 하지만, COSMOS와 지상국 통신 하드웨어(TNC)나 다른 소프트웨어(SoundModem, RF Checkout Box)를 이용하여 패킷을 송수신하기 위해서는 KISS, AX.25 같은 프로토콜은 지원하지 않아서 별도로 개발하여야 한다.

위성 개발을 한 번으로 끝나면 재사용에 대해서 고려할 필요 없이 현재 플랫폼만 고려하면 된다. 하지만, 위성 개발을 계속 진행하기 위해서는 개발 비용과 기간 단축을 고려하지 않을 수 없다. 개발 비용과 기간을 단축하기 위해서는 기존에 사용하는 플랫폼의 재사용이 중요하다.

본 논문에서는 지상국 소프트웨어의 재사용성 향상을 위해 COSMOS를 수정하지 않고, 소프트웨어 디커플링을 적용하여 지상국을 계층화 구조(layer structure)로 구성하여, 프로토콜(KISS, AX.25) 처리 계층을 구성하였다. 프로토콜 처리 계층을 검증하기 위하여 패킷 처리 프로그램을 개발하였다.

프로토콜 처리 계층의 검증을 위한 패킷 처리 프로그램에 필요한 요구 사항을 기재하였으며, 첫 번째 요구 사항인 다양한 통신 방식 지원은 UDP, TCP, Serial 통신을 통해 위성과 송수신이 되는 것을 확인하였다. 두 번째 요구 사항인 프로토콜 지원은 패킷 처리 프로그램의 화면을 통하여 KISS/AX.25 프로토콜의 header/tail이 추가되고 제거되는 것을 확인하였다. 세 번째 요구 사항인 데이터 병합은 1,356 bytes 크기의 위성 정보가 COSMOS로 전송되는 것을 확인하였다. 지상국 관제 프로그램의 재사용성을 검증하기 위해 COSMOS 대신 hercules라는 프로그램을 사용하여, 위성에 명령을 전송하고 위성 정보가 수신되는 것을 hex 값을 통해서 확인하였다. 송수신기/모뎀과 관제 프로그램이 서로 독립적으로 운영이 가능하여, 다른 위성 개발이나 수정 시 재사용이 가능한 것을 검증하였다.

참고문헌

[1] DOI : https://dx.doi.org/10.1787/9789264264014-en

[2] Han, S. H., Choi, Y. J., Cho, D. H., Choi, W. S., Gong, H. C. Kim, H. D., Choi, G. H., “Analysis of Cubesat Development Status in Korea”, Journal of the Korean Society for Aeronautical & Space Sciences 45(11), 2017.11, pp. 975-988

[3] Lee, E. J., “Design and Construction of Cummunication System and Ground Station for CANYVAL-X Cubesat”, Yonsei University, Master’s Thesis, 2014

[4] Lee, S. Y., “Design and Construction of Cubesat KAUSAT-5 Ground Station Implementing Automatic Tracking and Command System”, Korea Aerospace University, Master’s Thesis, 2017

[5] Jung, S. Y., “Performance Test for the SIGMA Communication System”, Kyunghee University, Master’s Thesis, 2017

[6] Jeon, S. Y., “Flight Software and Function Test of Flight Model STEP Cube Lab.”, Chosun University, Master’s Thesis, 2016

[7] Kim, H. G., Yu, S. K., Kim, O. J., Kee, C. D., Han, S. H., “SNUGLITE CubeSat Ground Station Constuction and Communication System Verification”, The Korean Society for Aeronautical & Space Sciences, 2017.11, pp. 459-460

[8] Park, J. H., Vishnu, A. M., Lee, S. Y., Jeung, I. S., “Amateur ground station development and testing for university CubeSat operations”, The Korean Society for Aeronautical & Space Sciences, 2017.04, pp. 279-280

[9] Kim, J. H., Jang, J. I., Park, S. H., “Vibration Analysis of a Nanosatellite for Microgravity Science Missions”, Journal of the Korean Society of Manufacturing Process Engineers 18(12), 2019.12, pp. 104-110

[10] cFS source, URL: https://github.com/nasa/cFS

[11] core Flight System (cFS) Background and Overview, URL: https://cfs.gsfc.nasa.gov/cFS-OviewBGSlideDeck-ExportControl-Final.pdf

[12] D. McComas, "Increasing flight software reuse with OpenSatKit," 2018 IEEE Aerospace Conference, Big Sky, MT, 2018, pp. 1-8, doi: 10.1109/AERO.2018.8396631.

[13] COSMOS, URL: https://cosmosrb.com/

[14] AX.25 Link Access Protocol for Amateur Packet Radio, Version 2.2, (1998)

[15] CCSDS Blue Books, URL: https://public.ccsds.org/Publications/BlueBooks.aspx

[16] Coupling, URL: https://en.wikipedia.org/wiki/Coupling_(computer_programming)

[17] Soundmodem, URL: http://uz7.ho.ua/

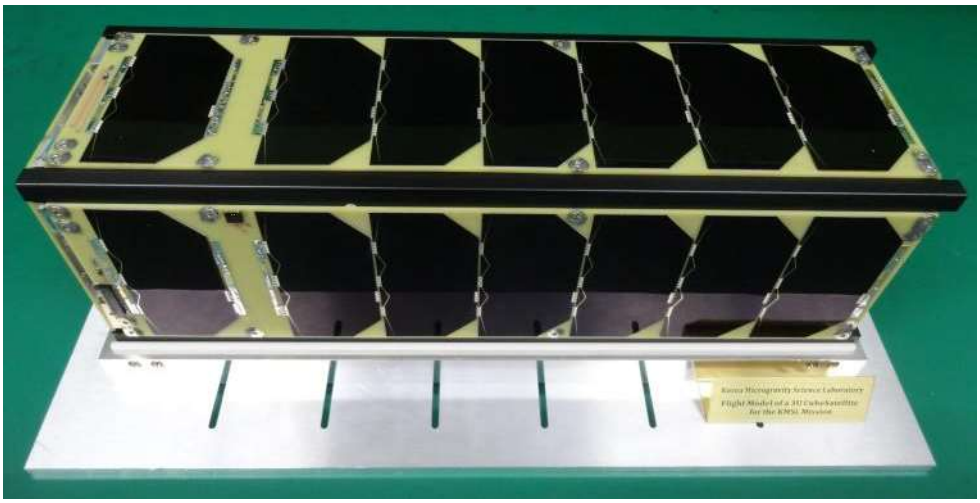
부록

조선대학교에서 개발하고 있는 KMSL 큐브위성은 지상국과 RF(Radio Frequency) 방식으로 통신하며, 통신 규격은 아래 표와 같다.

KMSL RF 통신 규격

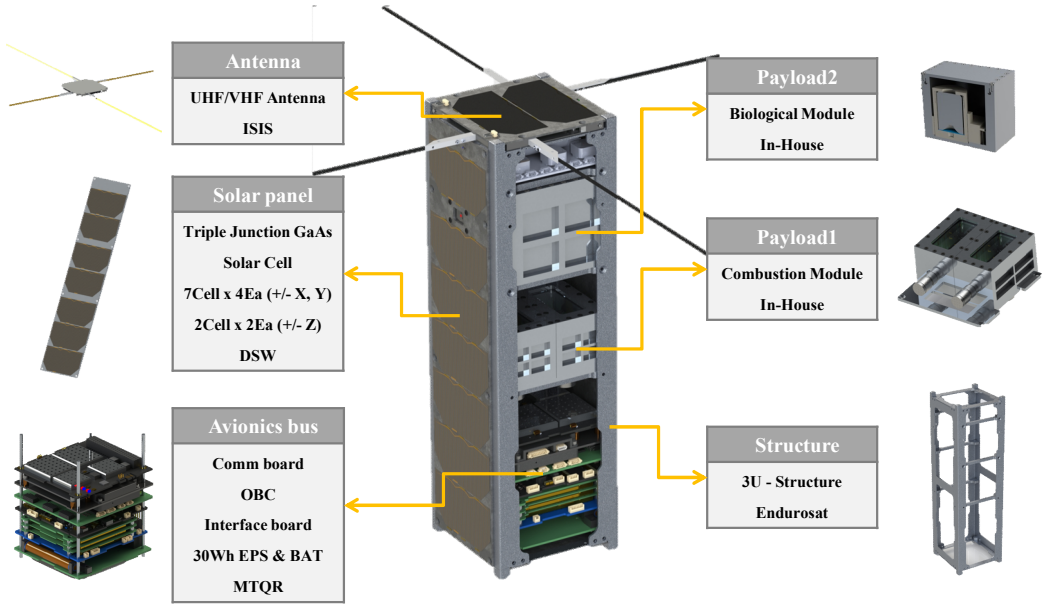
	Uplink	Downlink
Center Frequency	145.835 MHz (VHF)	437.265 MHz (UHF)
Bitrate	9600 bps	9600 bps
Protocol	AX.25	AX.25
Modulation	FSK-G3RUH	BPSK-G3RUH

1. KMSL cubesat hardware



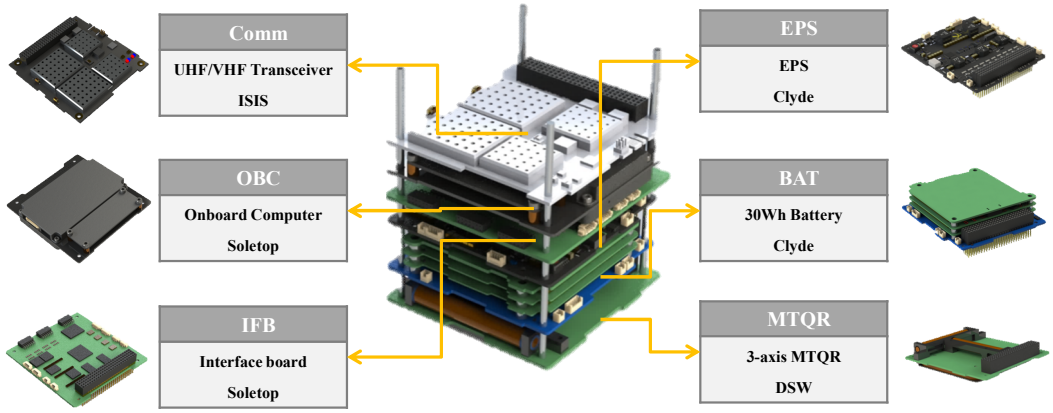
KMSL 3U Cubesat

KMSL 큐브위성은 100x100x340.5mm 크기에 3.595kg의 3U 큐브위성으로 완성된 모습은 위 그림과 같다. 서브 시스템의 구성은 아래 그림과 같고, 크게 Avionics, 과학 임무 탑재체로 이루어져 있으며, Center of Mass와 Data Interface(Harness)를 고려해 배치하였다.



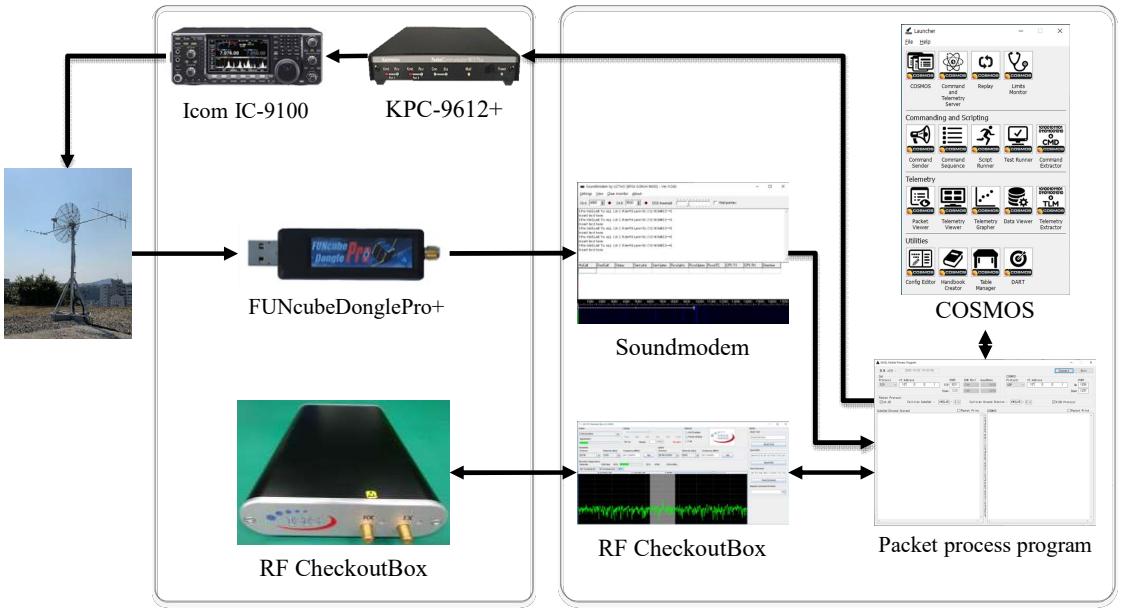
KMSL cubesat 구조

KMSL 큐브위성 avionics 부품 구성도는 아래 그림과 같고, 위성 Bus 간 harness cable을 최소화하기 위해서 PC104 connector를 최대한 활용할 수 있도록 avionics Bus를 위성 하단부에 배치하여 적층하였다.



KMSL avionics

2. KMSL 지상국



KMSL 큐브위성 관제를 위한 지상국 구성

KMSL 큐브위성을 관제하기 위한 지상국은 위 그림과 같이 구성되어 있다.

a. Hardware

- Icom IC-9100(transceiver) - 한 대의 장비에서 데이터 전송을 위한 송신과 수신이 가능한 단말 장치로 증폭기(power amplifier)가 내장된 아마추어 무선 장비이다.
- Kantronics KPC-9612+ (TNC - Terminal Node Controller) - 터미널에서 수신한 디지털 데이터를 아날로그 신호로 변조하여 트랜시버를 통하여 전송하고, 트랜시버를 통하여 수신한 아날로그 신호를 복조하여 디지털 데이터를 터미널로 전송을 담당하는 하드웨어 modem(변복조기)이다.
- ISIS RF checkout box(transceiver) - 위성과 신호를 송수신하기 위해 설계된 실내용 테스트 장비이며, 신호의 변복조는 소프트웨어에서 수행된다.
- FUNcube Dongle Pro+(receiver) - 150kHz ~ 1.9GHz의 무선 대역 수신이 가능한 SDR(Software Defined Radio)입니다. SDR은 신호를 수신하고, 기존 수신

기에서 처리하던 라디오의 복조, 디코딩 및 주파수 변환과 같은 많은 기능이 소프트웨어에서 수행된다. FUNcube Dongle Pro+는 안테나와 USB만 연결하면 되고, 자체에는 물리적 제어 기능이 없으며 모든 설정은 호스트 컴퓨터에서 소프트웨어로 제어한다.

b. Software

- ISIS RF Checkout Box - 하드웨어와 같이 제공되는 소프트웨어로 송수신을 위한 변복조 방식, 전송 속도, 주파수를 지정할 수 있다. 수신/송신된 패킷 및 스펙트럼에 대한 정보를 보여주는 패널과 위성에 명령을 보낼 수 있는 uplink 패널이 있다. TCP/IP를 통한 KISS(Port 3211)와 binary(Port 3210) 인터페이스 방식으로 다른 프로그램과 연결할 수 있다.
- Soundmodem - SDR을 통해 수신된 아날로그 신호를 디지털 데이터로 변환시켜 주는 software dual-port Packet-Radio TNC이며, Andrei(UZ7HO)에 의해 개발되었다. 다양한 복조 방식과 전송 속도를 제공한다. TCP/IP를 통한 AGWPE(Port 8000) 및 KISS(Port 8100 - 변경 가능) 인터페이스 방식으로 다른 프로그램과 연결할 수 있다.