JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 27(5): 610-620, Oct. 2023

FTP와 JSON을 활용한 대용량 미디어의 항공장비용 데이터 로드 프로세스

Data Load Process of large-sized media for avionics using FTP and JSON

최 지 환* · **최 낙 민** · **신 재 권**
LIG Nex1 항공연구소

Ji-Hwan Choi* · **Nak-Min Choi** · **Jae-Kwon Shin**
Aeronautical Research Institute of LIG Nex1, Daejeon 34127, Republic of Korea

[요 약]

4차 산업 혁명에 기반한 기술발전 및 항공사들의 고객 유치를 위한 경쟁으로 인해 항공기 인테리어 시장에 대한 관심이 증가하고 있으며, 그 일환으로 국내에서는 FAA Part.25급 민항기를 대상으로 한 CDS (Cabin Display System)이 개발되고 있다. CDS는 IDPM (Integrated Display Processing Module)로 제어되는 대형 Flexible 및 투명 OLED (Organic Light Emitting Diodes)를 활용하여 승객들에게 다양한 멀티미디어 서비스를 제공하는 시스템으로 고품질의 서비스 제공을 위해 대용량의 미디어 콘텐츠 활용이 필수적으로 요구된다. 본 논문에서는 대용량 파일들의 효율적인 Data Load Process를 수행하기 위한 새로운 방안을 제시하고 그 구현 및 성능을 다룬다. 본 연구 결과는 기존 ARINC-615A 대비 Data Load Process 개발 비용의 절감과 더불어 신뢰성 높은 대용량 파일 전송이 필요한 항공 장비의 Data Load Process 개발에 대체 적용이 가능할 것으로 기대된다.

[Abstract]

The interest in the aircraft interior market is gradually growing due to technological development based on the 4th industrial revolution and competition for airlines to attract customers, and as part of that, Cabin Display System (CDS) for FAA Part.25 civil aircraft is being developed in Korea. The CDS is a system that provides various multimedia services to passengers by utilizing Flexible and Transparent Organic Light Emitting Diodes (OLED) with Integrated Display Processing Module (IDPM). This paper presents a new method for efficient Data Load Process of large-sized files and deals with their implementation and performance. The results of this study are expected to be applied to Data Load Process development of avionics that require reliable large-capacity file transmission along with reducing the costs of development compared to existing ARINC-615A.

Key word : ARINC-615A, Data Load Process, FTP (File Transport Protocol), JSON (Java Script Object Notation), Smart Cabin.

http://dx.doi.org/10.12673/jant.2023.27.5.610

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-CommercialLicense(http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 11 September 2023; **Revised** 12 October 2023
Accepted (Publication) 25 October 2023 (30 October 2023)

***Corresponding Author: Ji-Hwan Choi**

Tel:***－****－****
E-mail: jihwan.choi@lignex1.com

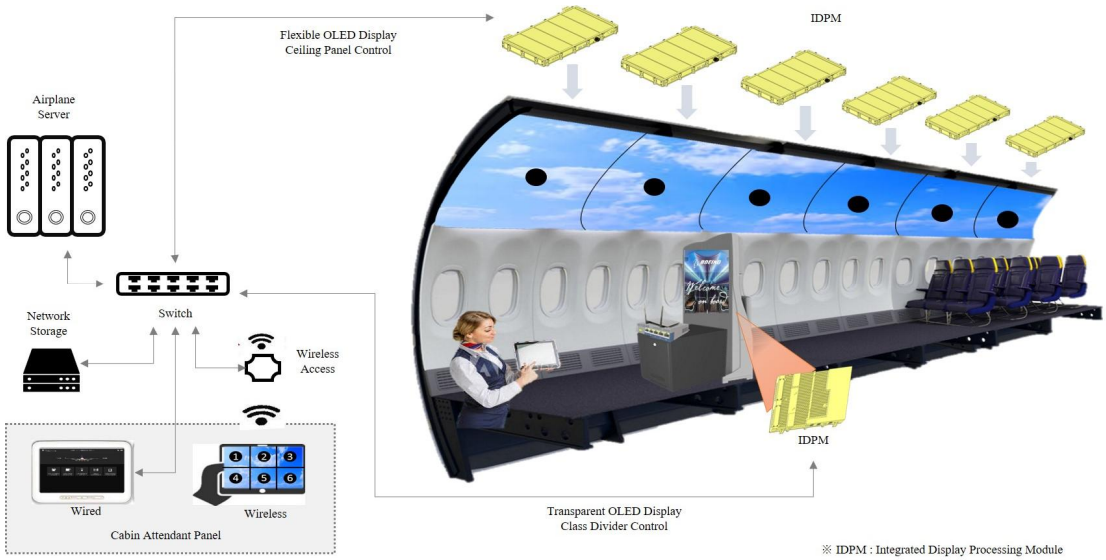

그림 1. Cabin Display System 개요
Fig. 1. Cabin Display System Overview

Ⅰ. 서 론

현대의 글로벌 항공 산업은 IOT (Internet of Things), ICT(Information & Communications Technology) 및 AI (Artificial Intelligence)와 같은 4차 산업 혁명을 중심으로 빠르게 발전[1]-[5]하고 있으며, 항공사들의 고객 유치에 대한 경쟁으로 인해 항공기 인테리어 산업 역시 빠른 속도로 발전하고 있다. 그 중에서도 과거 한 번에 많은 승객을 안전히 운반하는 수단에만 목적을 두었던 것에서 벗어나 Smart Cabin처럼 승객들에게 다양한 유저 경험을 제공하기 위한 기내 엔터테인먼트 (IFE: In-Fligth Entertainment) 시스템들에 대한 관심이 높아지고 있으며, 이러한 실정에 맞춰 현재 국내에서도 대형 Flexible 유기 발광 다이오드 (OLED: Organic Light Emitting Diodes) 및 투명 유기 발광 다이오드 (OLED: Organic Light Emitting Diodes) 패널을 활용한 CDS (Cabin Display System)가 개발되고 있다.

CDS는 Smart Cabin을 구성하는 시스템 중 하나로, 이더넷 인터페이스를 통해 객실에 설치된 55인치 OLED 패널들을 제어하는 IDPM (Integrated Display Processing Module)과 연동하는 방식으로 동작한다. 패널에 연결된 각 IDPM들은 별도의 그룹으로 나뉘어 대형 분할 영상을 시연할 수 있는 기능을 제공해 승객들에게 차별화된 영상 및 웹 서비스를 제공한다[6]. 이때, CDS가 고품질의 멀티미디어 서비스를 제공하기 위해서는 대용량의 고화질 미디어 파일들의 활용이 필수적이며 각 재생 파일들의 시나리오인 재생 목록들의 구성에 대한 업데이트도 빈번하게 요구된다. 이러한 시스템 특징에 따른 요구사항을 만족하기 위해서는 해당 항공장비의 파일 업데이트를 위한 ARINC-615A와 같은 Data Load Process가 필요한데 해당 프로세스를 본 시스템에 적용하기에는 한계점이 존재한다.

이에 본 논문은 Smart Cabin에 적용되는 CDS의 특징을 고려하여 FTP[7]와 JSON 파일을 활용해 대용량 파일들의 효율적인 Data Load Process를 수행할 수 있는 새로운 방안을 제안한다.

본 논문의 2장에서는 CDS의 특징과 그에 따른 Data Load Process에 적용된 이론적 배경들을 설명하며 3장에서는 그 상세 구현 내용을 다룬다. 마지막으로 4장에서는 본 논문에서 제안하는 방식의 실제 성능을 확인하고 기대효과를 기술한다.

Ⅱ. 선행 이론

2-1 Cabin Display System

Fig. 1과 같이 CDS는 항공기 객실에 설치된 OLED 패널들을 제어하는 IDPM들로 구성되며 각 IDPM 장비들은 자신과 연결된 OLED 패널을 1대1로 연동한다. 이때 각 IDPM들은 star topology 또는 daisy-chain topology를 구성해 이더넷 네트워크를 형성할 수 있으며 항공기 서버를 통해 CDS를 제어하는 CAP (Cabin Attendant Panel)과 MQTT (Message Queueing Telemetry Transport) 프로토콜로 연동된다[8]-[11].

CDS를 구성하는 각 IDPM들이 항공기 서버를 통해 CAP (Cabin Attendant Panel)과 연동하기 위해서는 MQTT Broker에 접속해야 하는데 이 과정을 수행하기 위해서는 몇 가지 조건이 필요하다. 최초 항공기 서버에 접속된 IDPM들은 항공기 서버에서 동작하는 MQTT Broker가 아닌 별도의 네트워크를 통해 Commissioning 이라는 프로세스를 수행해야 한다.

해당 프로세스에서 각 IDPM들은 SSL/TLS를 통한 인증 절차[12], [13]를 거쳐야하며, MQTT Broker를 통해 통신에 사용할 Topic을 구성하기 위한 장비 고유 식별자인 '<dev-id>'를 할당 받는다. 이 외에도 장비 연동을 위한 다양한 설정을 진행하여 Commissioning 프로세스를 성공한 경우에만 항공기 서버

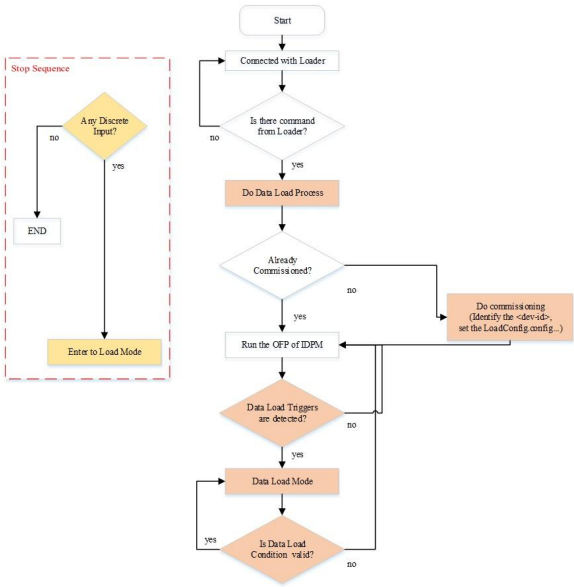

그림 2. Data Load Mode 진입 순서도
Fig. 2. Flow chart of Data Load Mode Entrance

에서 동작하는 MQTT Broker에 접속이 가능해진다.

최초 MQTT Broker를 통해 접속된 IDPM들은 사용자가 맞춤 제작한 시나리오인 재생 목록이 존재하지 않기 때문에 해당 콘텐츠들을 다운로드하기 위한 Data Load Process를 필수적으로 수행해야 한다. IDPM이 Data Load Process를 수행하기 위한 순서도는 Fig. 2와 같으며, 각 IDPM들은 장비 정비를 위한 Shop 환경과 실제 장비가 항공기에 장착되어 운용되는 Onboard 환경 모두에서 Data Load Process를 지원할 수 있도록 설계되어 있다.

이러한 과정을 거쳐 정상적으로 Data Load Process까지 마친 IDPM들은 CAP와 연동하여 객실 멀티미디어 서비스를 제공하게 된다.

2-2 FTP 프로토콜 활용

항공장비가 Data Load Process를 수행하기 위한 방법에는 앞서 1장에서 설명한 ARINC-615A라는 TFTP 기반의 항공기용 이더넷 Data Load Process[14], [15]에 대한 가이드 문서가 존재한다.

하지만 해당 프로세스는 UDP 기반의 TFTP를 활용해 동작하기 때문에 SSL/TLS를 활용하는 CDS의 특징과 맞지 않고, UDP에 적용할 수 있는 DTLS[16]-[18]의 사용에 대한 고려도 ARINC-615A에서 옵션으로 제공하지 않기 때문에 적용할 수 없다.

또한, TFTP는 아주 간단한 구조의 명령어 세트[19]만 지원할 수 있고 폴더 구조에 대한 접근이 불가능하기 때문에 다양한 미디어 콘텐츠들의 시나리오를 구성하는 CDS 재생 목록 폴더를 관리하기에도 까다롭다. 특히, UDP 기반으로 동작하기 때문에 Flow Control을 제공하는 TCP기반 통신과 비교해

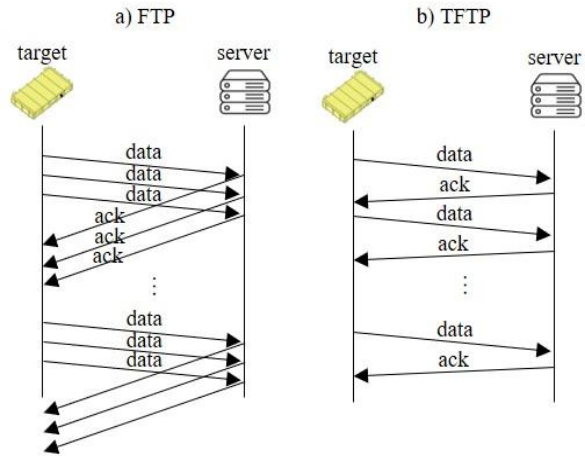

그림 3. FTP와 TFTP의 데이터 전송 진행
Fig. 3. Data transfer sequence of TFTP and FTP

daisy-chain topology로 구성된 네트워크 환경에서 패킷 Loop에 의한 데이터 유실 가능성이 높아 신뢰성 있는 데이터 전송을 보장하기 어렵다 [20], [21].

반면, FTP는 대부분의 시스템에서 호환이 가능하면서 TFTP를 포함한 다른 프로토콜들과 비교[19], [22]-[24]해 대용량 파일 전송에 있어 TFTP가 가졌던 단점을 보완할 수 있다는 이점도 가진다.

FTP는 기본적으로 ID/Password를 통한 인증 이후에 통신이 가능하며 SSL/TLS를 활용한 데이터 라인 암호화[25], [26]도 쉽게 할 수 있어 본 시스템이 사용하는 인증 ID/Password와 SSL/TLS를 동일하게 사용할 수 있다. 그리고 FTP는 파일 전송을 위해 TFTP보다 많은 명령어 세트를 지원[7], [19], [27]해 전송 실패 시 해당 지점부터 다시 다운로드 할 수 있으며 폴더에 대한 접근도 가능하기 때문에 다양한 폴더와 대용량 파일들로 구성된 CDS의 데이터를 관리하기에 매우 적합하다.

특히, 작은 파일들을 전송하는 경우와 다르게 대용량 파일 전송에 있어 FTP가 TFTP보다 유리한 측면을 가진다[28]. Fig. 3처럼, UDP에 기반한 TFTP는 패킷 전송 유무를 확인하기 위해 별도의 ACK 메시지를 각 송신 패킷하다 확인해야 하지만 FTP는 TCP의 flow control[29]를 통해 한번 채널이 형성된 이후부터는 빠르게 다수의 데이터 패킷을 전달할 수 있다[30].

2-3 JSON 포맷 데이터 파일 활용

앞서 언급된 ARINC-615A의 경우 Data Load Process를 수행하기 위해서 ARINC-665 라는 별도 문서[15], [31]에 정의된 포맷의 데이터 파일을 활용한다. 정의된 포맷에 따라 Data Load를 수행하기 때문에 해당 포맷의 모든 파일들은 ARINC-615A를 통해 Data Load Process를 수행할 수 있지만, 정해진 필드 크기와 순서에 맞춰서 다양한 데이터를 구성하도록 되어있어 사람이 보기 어렵고 구현이 복합하다는 단점이 존재한다[31].

표 1. Data Load에 사용되는 JSON 포맷 프로토콜 파일
Table 1. JSON formatted Protocol files for Data Load

File Name	Target Process	Description
LoadConfig.config	All	Configuration Parameter for Data Load
<dev-id>_software.list	Refresh	Information about Software file structure that are loaded in Target
<dev-id>_media.list	Refresh	Information about Media file structure that are loaded in Target
<package>.info	Upload	Information data of <package>
<dev-id>_software.dlist	Download	Software file list that Loader wants to download from Target
<dev-id>_media.dlist	Download	Media file list that Loader wants to download from Target
<dev-id>_software.rlist	Remove	Software file list that Loader wants to remove from Target
<dev-id>_media.rlist	Remove	Media file list that Loader wants to remove from Target

그렇기 때문에 본 논문에서 제안하는 Data Load Process들은 ARINC-665 대신 Table 1에 기술된 JSON 포맷의 프로토콜 파일을 사용한다.

JSON 포맷의 데이터는 이름과 값이 한 쌍을 이루는 데이터들이 모인 객체의 형태로 구성되어 사람이 쉽게 읽고 쓰는 것이 가능하며 다양한 타입의 데이터들을 손쉽게 관리할 수 있다[32]-[34]. 해당 데이터는 텍스트 문자로 구성되기 때문에 XML[35]과 같은 데이터 포맷과 비교해 언어와 시스템 환경에 종속적이지 않아 호환성이 높고, 사람이 쉽게 데이터를 확인할 수 있기 때문에 개발 기간의 단축을 통한 비용 절감도 가능하다.

Ⅲ. Data Load Process

3-1 제안 프로세스의 개요

Table 1에 기술된 JSON 포맷의 파일들은 본 논문에서 제안하는 Data Load Process의 5가지 기능들을 수행하는데 사용된다. 각 기능들은 Refresh, Upload, Download, Remove와 Stop으로 정의되며 각 기능에 대한 명령어는 Table 2와 같다.

표 2. Data Load에 사용되는 기능별 TCP 명령어
Table 2. TCP Commands of Data Load Functions

Command	Function
refresh	Update Target's internal file list
upload <package>	Upload the <package> to Target
download	Download selected file from Target
remove	Remove selected file from Target
stop	Stop Load process that currently performed

각 프로세스들이 수행되는 네트워크 구조와 연동 방식은 Fig. 4와 동일하다.

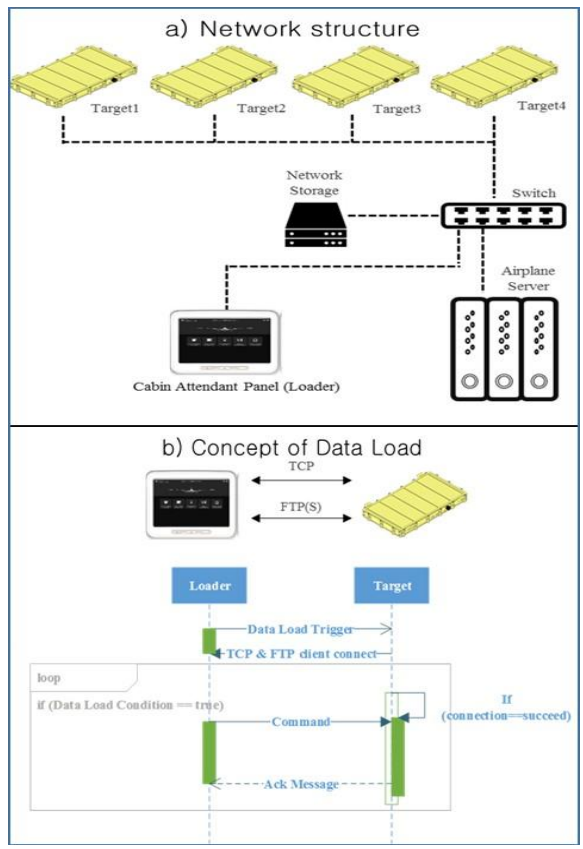

그림 4. Loader와 대상 장비 간 네트워크 연동 구조
Fig. 4. Network structure between Loader and Target

Fig. 4에서처럼 각 대상 장비인 IDPM들은 Loader로부터 명령을 받기위한 TCP 한 채널과 파일 데이터를 전송하기 위한 FTP 한 채널을 구성하여 Data Load Process를 수행한다.

표 3. LoadConfig.config 파일
Table 3. LoadConfig.config file

Name	Type	Value
url	string	Loader URL
Port	numeric	Port number of Loader
encryption	boolean	true or false
Id	string	user identifier (if FTP)
Pw	string	user password (if FTP)
protocol	string	tftp or ftp

대상 IDPM들은 'LoadConfig.config' (Table. 3) 파일에 저장된 정보를 기반으로 Data Load Process를 수행하기 위한 설정을 수행하며 해당 파일은 최초에 소프트웨어와 함께 장비 내부에 설치된 상태로 제공된다.

3-2 Refresh 기능

Refresh 기능은 대상 IDPM 내부에 구성된 미디어 콘텐츠 파일 구조와 설치된 소프트웨어 패키지 파일 정보를 확인하는 기능으로 Fig. 5와 같이 수행된다.

대상 IDPM들은 Loader로부터 'refresh' 명령을 받으면 자신의 내부에 설치된 소프트웨어 패키지 목록이 담긴 '<dev-id>_software.list'(Table. 4) JSON 파일과 자신의 미디어 콘텐츠 구성에 대한 정보가 담긴 '<dev-id>_media.list'(Table. 5) JSON 파일을 생성하여 FTP를 통해 Loader로 전달한다.

여기서 '<dev-id>'는 앞서 설명한 Commissioning 단계에서 장비마다 할당받는 고유 식별자이며 Loader는 해당 정보를 기반으로 어떤 장비의 내부 데이터 목록인지 확인할 수 있다.

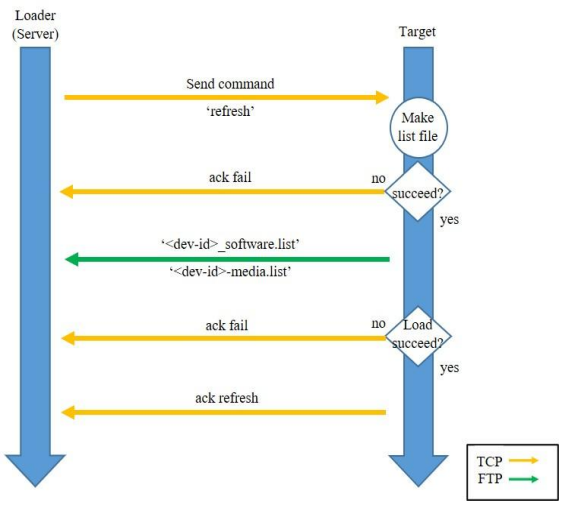

그림 5. Refresh 수행 절차
Fig. 5. Refresh Process

표 4. <dev-id>_software.list 파일
Table 4. <dev-id>_software.list file

Name	Type	Value
flie-lists	string array	[installed software files, ...]

표 5. <dev-id>_media.list 파일
Table 5. <dev-id>_media.list file

Name	Type	Value
flie-lists	string array	[installed media content files, ...]

3-3 Upload 기능

Upload 기능은 사용자가 Loader를 통해 대상 IDPM 내부에 구성된 미디어 콘텐츠 파일 구조와 설치된 소프트웨어 패키지 파일을 업데이트하는 기능으로 Fig. 6과 같이 수행된다.

사용자가 새롭게 업데이트를 수행할 패키지를 선택해 Upload 요청을 수행하면 Loader는 사용자가 선택한 패키지 이름을 담아 'upload <package>' 라는 명령어를 생성해 대상 IDPM에 전달한다. 명령을 수신한 IDPM들은 Loader의 FTP 서버로부터 '<package>.info'(Table. 6) 파일을 받아와 해당 패키지에 대한 정보를 획득한다.

표 6. <package>.info 파일
Table 6. <packet>.info file

Name	Type	Value
version	string	version of package
model	string array	[compatible aircraft model IDs, ...]
file-list	JSON array	[{"filename": CRC_value}, ...]
hw-info	string	compatible Hardware part number

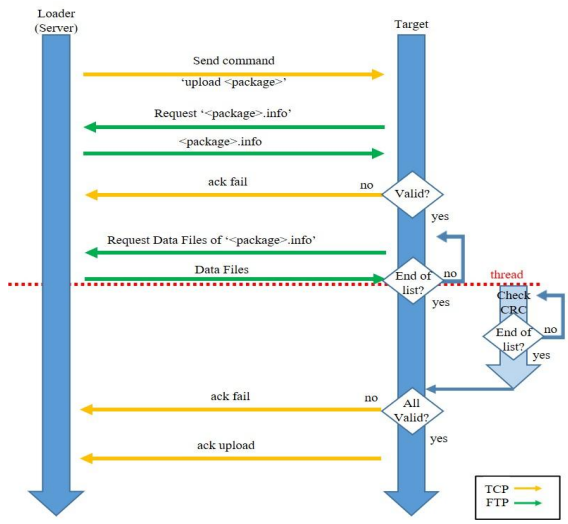

그림 6. Upload 수행 절차
Fig. 6. Upload Process

'<package>.info' 파일을 성공적으로 수신한 IDPM들은 호환 항공기 모델, 소프트웨어 버전을 통해 호환성 검사를 수행하여 Upload 기능 수행 여부를 판단한다. 만약, 해당 단계에서 비호환성이 탐지된다면 IDPM은 Loader에 'ack fail' 메시지를 통해 Upload 기능 실패를 알린다.

호환성 검사에 성공한 경우 각 IDPM들은 '<package>.info' 파일에 담긴 파일 리스트 정보를 통해 Loader의 FTP 서버에서 해당하는 파일들을 받아온다. 이때. 해당 파일이 FTP 서버에 존재하지 않거나 '<package>.info'의 CRC 정보와 다르면 'ack fail' 메시지를 통해 Upload 기능 실패를 알리고, 모든 파일들이 CRC 검사를 통해 유효성이 검증된 경우 'ack upload'를 반환해 Upload 기능의 성공적인 수행을 알린다.

추가적으로 본 장절의 Upload 기능은 ARINC-615A의 Short Load[14]와 같은 기능을 지원할 수 있기 때문에 현재 IDPM 장비에 설치된 패키지 구성 파일이 새롭게 요청받은 패키지 파일과 동일한 경우 해당 파일을 중복으로 Load하지 않고 이미 설치된 파일을 활용해 Data Load 시간을 줄일 수 있다.

3-4 Download 기능

Download 기능은 사용자가 Loader를 통해 대상 IDPM 내부에 구성된 미디어 콘텐츠 파일 구조와 설치된 소프트웨어 패키지 파일을 가져오는 기능으로 Fig. 7과 같이 수행된다. 해당 기능은 IDPM에 설치된 파일들을 받아오는 것이기 때문에 Refresh 기능을 선행해야 진행할 수 있다.

Loader는 사용자가 선택한 파일들의 종류에 따라 '<dev-id>_software.dlist' (Table. 7) 또는 '<dev-id>_media.dlist' (Table. 8)를 FTP 서버 루트 경로에 생성하고 대상 <dev-id>를 가진 IDPM에게 'download' 명령을 보낸다.

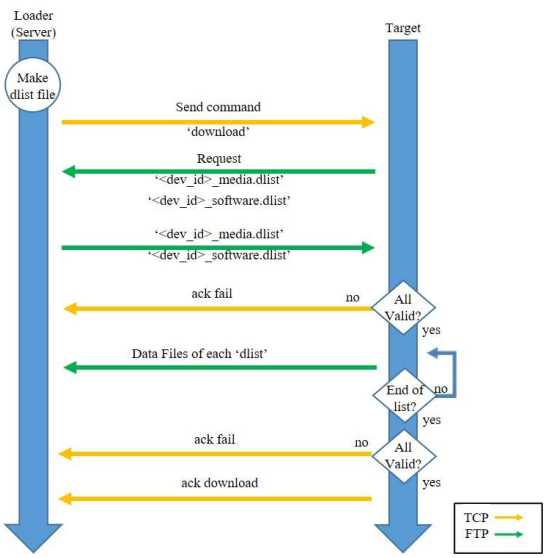

그림 7. Download 수행 절차
Fig. 7. Download Process

표 7. <dev-id>_software.dlist 파일
Table 7. <dev-id>_software.dlist file

Name	Type	Value
flie-lists	string array	[software files selected by user, ...]

표 8. <dev-id>_media.dlist 파일
Table 8. <dev-id>_media.dlist file

Name	Type	Value
flie-lists	string array	[media files selected by user, ...]

해당 명령을 수신한 IDPM은 FTP 서버에서 자신에 해당하는 '.dlist' 파일을 가져와 정보를 확인하고 FTP 기능을 통해 자신의 <dev-id> 폴더를 FTP 서버에 생성하고 그 안에 사용자가 요청한 파일을 Load한다.

이때, 사용자가 요청한 파일이 IDPM 내부에 없는 파일이거나 전송 과정 중 문제가 발생한 경우 'ack fail'을 통해 실패를 반환하며, Download 기능이 정상 수행된 경우 'ack download'를 반환하여 성공적인 기능 수행을 알린다.

3-5 Remove 기능

Remove 기능은 사용자가 Loader를 통해 대상 IDPM 내부에 구성된 미디어 콘텐츠 파일 구조와 설치된 소프트웨어 패키지 파일을 제거하는 기능으로 Fig. 8과 같이 수행된다. 해당 기능은 IDPM에 설치된 파일이 손상되어 정상 동작이 불가한 경우 수행될 수 있으며 마찬가지로 IDPM 내부 파일을 대상으로 수행되기 때문에 Refresh 기능이 선행되어야 한다.

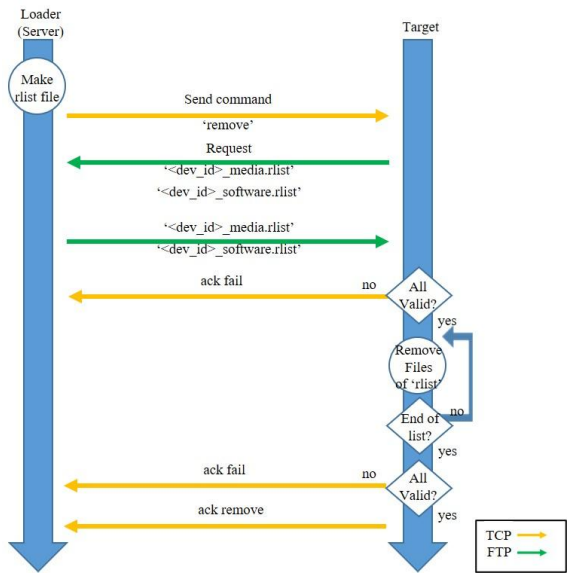

그림 8. Remove 수행 절차
Fig. 8. Remove Process

표 9. <dev-id>_software.rlist 파일
Table 9. <dev-id>_software.rlist file

Name	Type	Value
flie-lists	string array	[media files selected by user, ...]

표 10. <dev-id>_media.rlist 파일
Table 10. <dev-id>_media.rlist file

Name	Type	Value
flie-lists	string array	[media files selected by user, ...]

Remove 기능은 Download 기능과 유사하며 Loader는 사용자가 선택한 파일들의 종류에 따라 '<dev-id>_software.rlist' (Table. 9) 또는 '<dev-id>_media.rlist'(Table. 10)를 FTP 서버 루트 경로에 생성하고 대상 <dev-id>를 가진 IDPM에게 'remove' 명령을 보낸다. 해당 명령을 수신한 IDPM은 FTP 서버에서 자신에 해당하는 '.rlist' 파일을 가져와 정보를 확인하고 사용자가 요청한 파일을 IDPM 내부에서 삭제한다.

이때, 사용자가 요청한 파일이 IDPM 내부에 없는 파일이거나 파일 삭제 과정 중 문제가 발생한 경우 'ack fail'을 통해 실패를 반환하며, Remove 기능이 정상 수행된 경우 'ack remove'를 반환하여 성공적인 기능 수행을 알린다.

3-6 Stop 기능

Stop 기능은 사용자가 Loader를 통해 현재 진행 중인 Data Load Process를 중단하는 기능으로 Fig. 9와 같다.

각 IDPM들이 특정 Data Load Process 기능을 수행하는 도중 TCP 채널을 통해 'stop' 명령을 수신하면, 현재 수행하던 기능 수행을 취소한다. 이때, 해당 기능을 통해 취소된 기능은 정상 종료된 기능이 아니고 사용자가 명시적으로 취소한 기능이기 때문에 이후 일어나는 Data Load Process에서 FTP의 기능을 활용한 이어받기를 수행하지 않는다.

명령을 수신한 IDPM이 진행 중인 기능을 정상적으로 취소하지 못하면 'ack fail'을 통해 Stop 기능 수행 실패를 알리며, 정상적으로 진행 중인 기능 취소에 성공한 경우 'ack stop'을 반환하여 성공적인 기능 수행을 알린다.

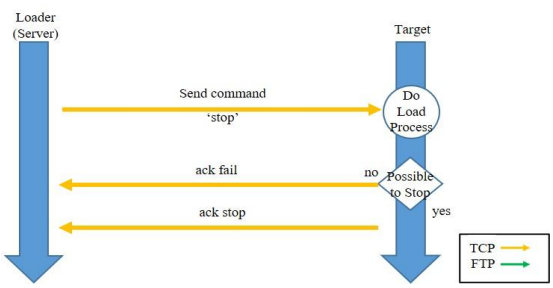

그림 9. Stop 수행 절차
Fig. 9. Stop Process

Ⅳ. 구현 및 성능 분석

4-1 시험 환경

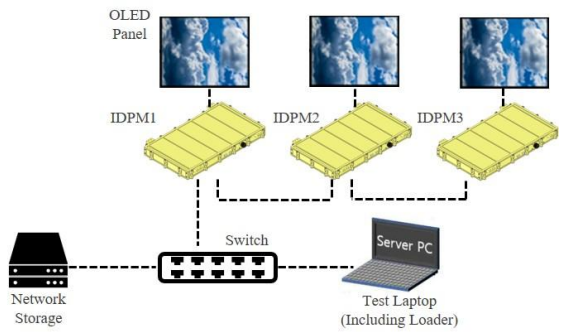

그림 10. 시험 환경 구성
Fig. 10. Configuration of Test Environment

본 장절에서는 제안하는 Data Load Process 기능 중 핵심 기능인 Upload와 Download에 대한 구현과 그 성능을 확인한다.

시험 환경은 Fig. 10에서와 같이 3대의 IDPM이 daisy-chain topology로 구성되어 Loader용 프로그램이 동작하는 노트북 PC 장비와 이더넷 인터페이스로 연동된다.

이때, 항공기에서 연동되는 CDS는 대용량의 패키지를 활용하기 때문에 별도 NAS (Network Access Storage)를 구축하는데 해당 저장장치를 노트북 PC의 FTP 서버에서 바인딩하는 방식으로 연동한다. 상세 구성 환경의 정보는 Table 11과 같다.

표 11. 시험 환경 정보
Table 11. Information of Test Environment

Name	Type	Value
url	string	Loader URL
Port	numeric	Port number of Loader
encryption	boolean	true or false
Id	string	user identifier (if FTP)
Pw	string	user password (if FTP)
protocol	string	tftp or ftp

Data Load Process의 대상 장비인 IDPM에 적용된 CPU는 NXP 사의 i.MX8MQaud[36]이며, MQTT 연동을 위해 mosquitto 2.0.10 라이브러리[37]를 사용했다.

Loader 장비를 구현한 노트북 PC의 경우 Intel 사의 i5-9400 CPU를 사용하며 MQTT 연동을 위해 MQTTnet 4.1.4.563 라이브러리를 사용했다.

이더넷 스위치를 포함한 시험 환경을 구성하는 모든 장비들은 1Gbps의 네트워크 대역폭을 지원한다.

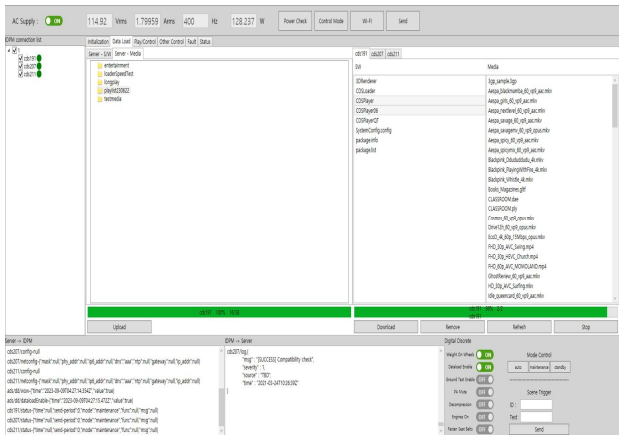

그림 11. CDS Data Load 시험용 프로그램 유저 인터페이스
Fig. 11. Program UI for CDS Data Load Test

본 논문에서 제안된 방법으로 구현된 Data Load Process 소프트웨어를 시험하기 위해 구성한 시험용 프로그램은 Fig. 11과 같다.

해당 프로그램은 Loader를 포함하여 CDS 시스템을 구성하는 IDPM 장비들을 제어하고 관리하는 프로그램으로 Fig. 11에 보이는 인터페이스는 Data Load Process 수행을 담당하는 화면이다.

좌측 트리에 구성된 3대의 장비들 내부에 구성된 미디어 및 소프트웨어 패키지를 메인 화면 우측에서 확인할 수 있으며, 좌측에서는 현재 Loader에서 구성하는 패키지 목록들을 확인할 수 있으며 Refresh, Upload, Download, Remove, Stop 명령 생성을 위한 버튼들과 Upload, Download 프로세스의 진행도를 확인할 수 있는 Progress Bar가 존재한다.

하단에는 디버깅을 위한 메시지 창과 함께 Fig. 2에 기술된 것처럼 IDPM을 Data Load Mode에 진입시키기 위한 Digital Discrete 신호를 생성할 수 있는 인터페이스가 위치한다.

4-2 Upload 성능 측정

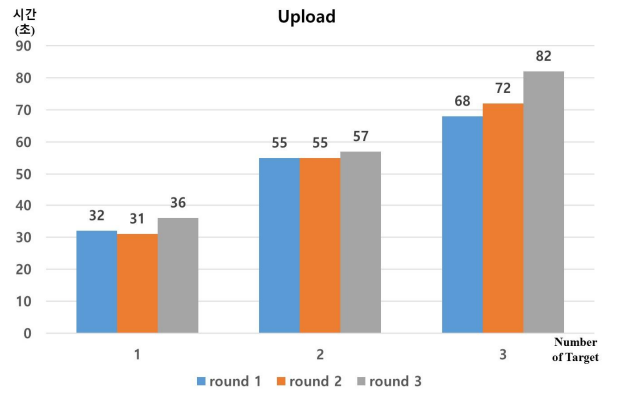

그림 12. Upload 성능 측정 결과
Fig. 12. Test results of Uplaod Performance

Upload 성능 측정을 위해 Fig. 10처럼 3대의 IDPM이 연결된 상태에서 각 1대, 2대, 3대를 동시에 연동하는 상황에서 3Gbytes의 패키지를 Upload할 때 장비에서 걸린 시간들의 평균값을 계산했으며 그 결과는 Fig. 12와 같다.

측정에 걸린 시간은 각 1대, 2대, 3대가 동시에 연동된 상황에서 각 장비들이 Upload 프로세스를 수행하는 시간은 평균 33초, 55.6초, 74초가 걸렸으며 속도로는 약 90.9MB/s, 53.9MB/s, 40.5MB/.s 정도임을 알 수 있다.

이는 1Gbps 대역의 네트워크 환경에서 거의 모든 네트워크 대역을 활용해 고속의 다운로드가 가능함을 보여주며, 각 장비들의 시간 편차가 크지 않은 것을 통해 균일한 네트워크 분배도 적절히 이루어졌음을 알 수 있다. 다만, 3회 차 시험에서는 지속된 Load 수행으로 IDPM에 설치된 nvme SSD (Solid State Disk)의 발열이 속도 저하를 유발한 것을 확인할 수 있다.

설치된 패키지의 무결성 검증을 위해 노트북 PC에 설치된 시험용 프로그램으로 IDPM의 미디어 재생을 수행했으며 파일 손상 없이 모두 정상 동작함을 확인했다.

4-3 Download 성능 측정

Download 성능 측정은 Download 기능 수행을 대상으로 4-2 장절과 동일한 방식으로 진행했으며 그 결과는 Fig. 13과 같다.

측정에 걸린 시간은 각 1대, 2대, 3대가 동시에 연동된 상황에서 각 장비들이 Upload 프로세스를 수행하는 시간은 평균 33.6초, 75초, 94.3초가 걸렸으며 속도로는 약 89.2MB/s, 40MB/s, 31.8MB/.s 정도임을 알 수 있다.

Upload 대비 속도가 약간 줄어든 것을 확인할 수 있으나 마찬가지로 1Gbps 대역의 네트워크 환경에서 거의 모든 네트워크 대역을 활용해 고속의 다운로드가 가능함을 보여주며, 각 장비들의 시간 편차가 크지 않은 것을 통해 균일한 네트워크 분배도 적절히 이루어졌음을 알 수 있다. 다만, 3회 차 시험에서는 앞선 Upload 시험을 포함한 지속적인 Load 수행으로 IDPM에 설치된 nvme SSD (Solid State Disk)의 발열이 속도 저하를 유발한 것을 확인할 수 있다.

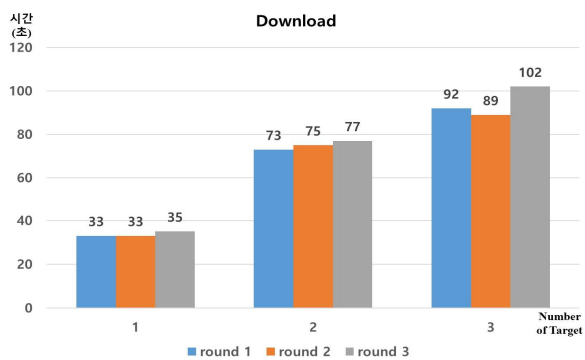

그림 13. Download 성능 측정 결과
Fig. 13. Test results of Download Performance

가져온 패키지의 무결성 검증을 위해 노트북 PC에서 IDPM에서 받아온 미디어 재생을 수행했으며 Upload 결과와 동일하게 파일 손상 없이 모두 정상 동작함을 확인했다.

4-4 전송 속도 비교

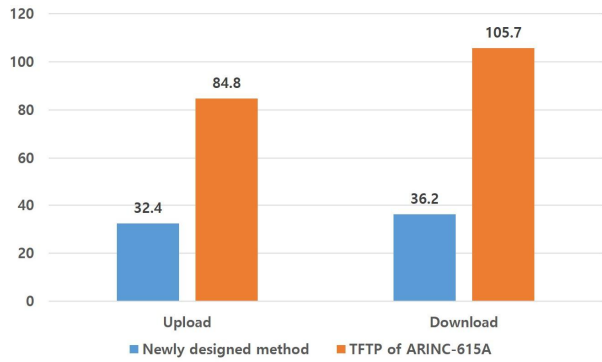

그림 14. 대용량 파일 전송 속도 비교
Fig. 14. Comparison of Large-sized file transfer speed

본 장절에서는 대용량 파일 전송에 있어 TFTP를 사용한 방식과 FTP를 사용한 방식의 속도 차이를 확인한다. 시험에 사용된 패키지 파일은 앞선 시험들과 동일하게 3Gbytes이며 1Gbps 네트워크 환경에서 1대의 IDPM 장비에서 Upload와 Download를 수행할 때 걸린 시간의 3회 측정 평균값을 측정했다.

TFTP의 대용량 파일 전송 성능 향상을 위해 TFTP 확장 옵션들을 적용[38], [39] 했음에도 Fig. 14와 같이 FTP보다 느린 것을 확인할 수 있다. TFTP의 경우 최대 확장 가능한 block size가 65464 bytes[40]이고 Fig. 3처럼 각 패킷마다 ACK 메시지를 기다려야하는 특징으로 인해 대용량 파일 전송이 필요한 경우 FTP 대비 속도가 더 느린 것을 확인할 수 있다.

게다가 TFTP는 서버 종류에 따라 전송 가능한 최대 파일 사이즈가 정해져있으며 최대 4Gbytes의 파일 전송 사이즈 제약[41]이 존재해 수십 Gbytes로 구성되는 다양한 재생 목록을 사용하는 CDS에 적용하기 적합하지 않음을 알 수 있다.

Ⅴ. 결 론

앞서 설명한 선행 이론을 통해 본 논문에서 제안하는 대용량 파일 전송을 위한 Data Load Process가 TFTP 기반으로 동작하는 ARINC-615A보다 빠르고 신뢰성 있는 파일 전송이 가능한 것을 알 수 있으며, ARINC-665 대비 JSON으로 구성된 프로토콜 파일을 사용한 방식이 구현에 용이하여 그에 따른 개발 비용의 단축 가능성을 살펴보았다. 또한, 4장절을 통해 3장절에서 제안한 Data Load Process 이론의 실제 구현을 확인했으며 그 성능에 대한 시험 및 측정을 통해 제안한 방식의 장점이 실제로 드러나는 것을 확인할 수 있었다.

본 논문에서 제안한 Data Load Process는 다양한 타입의 데이터를 쉽게 구성할 수 있는 JSON과 파일 데이터 전송에 다양한 기능을 제공하는 FTP를 활용하기 때문에 앞으로 보완 및 개선과 기능 확장이 매우 용이할 것으로 판단되며 기존 방안 대비 저비용의 빠른 장비 개발에 도움이 될 것으로 기대된다.

특히, 현재 빠르게 발전하는 항공 산업에서 고속의 네트워크와 대용량 데이터 연동, 보안 등에 대한 수요가 증가할 것으로 예상되는데 본 논문에서 제안하는 Data Load Process가 이러한 요구사항들을 만족할 수 있는 하나의 솔루션으로써 충분히 활용될 수 있을 것으로 기대된다.

Acknowledgments

본 연구는 산업통상자원부 재원으로 한국산업기술평가관리원(KEIT)의 지원을 받아 수행된 연구 결과입니다. [사업명: 스마트 캐빈 기술개발사업, 과제명: 항공기용 대형 Flexible OLED 디스플레이 시스템 개발/과제고유번호:20011845].

References

[1] G. Torkashvand, “Optimization of cabin design for enhanced passenger experience”, *Florida Institute of Technology ProQuest Dissertations Publishing*, 2019.28216828, 2019.

[2] B. K. Shin, H. S. Ji, “Smart Cabin System Technology Development Trend for single-aisle medium-sized aircraft”, *The Korean Society for Aeronautical & Space Sciences 2021 Fall Conference*, pp. 444-444, 2021.

[3] L. Mládková, “Industry 4.0: Human-Technology interaction: Experience learned from the aviation industry”, *In European Conference on Knowledge Management*, Academic Conferences International Limited, pp. 571-XXIII, 2018.

[4] J. Wang, J. Feng, X. Jiang, J. Tang, “A Intelligent Cabin Design Based on Aviation Internet of Things”, *Proceedings of the 5th China Aeronautical Science and Technology Conference. Lecture Notes in Electrical Engineering* [Online], vol 821. Springer, Singapore, 2022, Available: https://doi.org/10.1007/978-981-16-7423-5_95.

[5] D. Buhalis, “eAirlines: strategic and tactical use of ICTs in the airline industry”, *Information & Management*, 41(7), pp. 805-825, 2004.

[6] J. K. Shin, J. H. Choi, N. Min. Choi, J. H. Kim, J. Y. Kim, “Multi-Division Video Player Design and Implementation with Standard Interface for Aircraft in Embedded Environments”, *JOURNAL OF THE KOREA CONTENTS ASSOCIATION*, 22(11), pp. 80-91, 2022.

[7] J. Postel, J. Reynolds, File transfer protocol [Online], RFC 959, doi: 10.17487/RFC959, 1985, Avaliable: https://www.rfc-editor.org/info/rfc959.

[8] ISO/IEC. Information technology Message Queuing Telemetry Transport (MQTT) v3.1.1. [Internet], 20922:2016, Available: https://www.iso.org/standard/69466.html.

[9] U. Hunkeler, H. L. Truong, A. Stanford-Clark, "MQTT (2008) MQTT-S — A publish / subscribe protocol for Wireless Sensor Networks", *3rd International Conference on Communication Systems Software and Middleware and Workshops (CMOSWARE '08)*, Bangalore, India, pp. 791-798, doi: 10.1109/COMSWA.2008.4554519, 2008.

[10] OASIS. MQTT Version 3.1.1 [Internet], 2014, Avaliable: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1-os.pdf

[11] OASIS. MQTT Version 5.0 [Internet], March 2019, Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[12] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3 [Online], RFC 8446, doi: 10.17487/RFC8446, 2018, Avaliable: https://www.rfc-editor.org/info/rfc8446.

[13] R. Oppliger, SSL and TLS: Theory and Practice, Second Edition, *Artech House*, 2016.

[14] Airlines Electronic Engineering Committee, "Software Data Loader using Ethernet Interface. ARINC Report 615A", *Aeronautical Radio, Inc.*, 2551 Riva Road, Annapolis, Maryland 21401-7435, 2007.

[15] R. Rashmi, R. Srinidhi, S. Shriharsha, K. Ashwani, R. Byra, "ARINC 615A and 665-3 based Data Loader for Aircrafts", *International Journal of Engineering Trends and Technology,* 37, (5):260-264, doi: 10.14445/22315381/IJETT-V37P244, 2016.

[16] N. N. Mohamed, H. Hashim, Y. M. Yussoff, A. M. Isa, "Securing TFTP packet: A preliminary study", in *2013 IEEE 4th Control and System Graduate Research Colloquium*, pp. 158-161, doi: 10.1109/ICSGRC.2013.6653295, 2013.

[17] M. A. M. Isa, N. N. Mohamed, H. Hashim, S. F. S. Adnan, R. Mahmod, "A lightweight and secure TFTP protocol for smart environment", in *2012 International Symposium on Computer Applications and Industrial Electronics (ISCAIE)*, pp. 302-306, doi: 10.1109/ISCAIE.2012.6482117, 2012.

[18] M. A. M. Isa, H. Hashim, S. F. S. Adnan, J. L. A. Manan, R. Mahmod, "A secure TFTP protocol with security proofs", *arXiv preprint*, arXiv:1409.0060, 2014.

[19] K. Sollins, The TFTP Protocol (Revision 2) [Online], RFC 1350, doi: 10.17487/RFC1350, 1992, Avaliable: https://www.rfc-editor.org/info/rfc1350.

[20] G. Yoon, D. H. Kwon, S. C. Kwon, Y. O. Park, Y. J. LEE, "Ring topology-based redundancy Ethernet for industrial network", *SICE-ICASE International Joint Conference*, pp. 1404-1407, doi: 10.1109/SICE.2006.315661, 2006.

[21] S. K. Chin, R. Braun, "A survey of UDP packet loss characteristics", in *Conference Record of Thirty-Fifth Asilomar Conference on Signals on System and Computers*, Cat. No. 01CH37256, Vol. 1, pp. 220-204, doi: 10.1109/ACSSC.2001.986905, 2001.

[22] Computer Hope. Linux scp command [Internet]. Nov. 2019, Avaliable: https://www.computerhope.com/unix/scp.htm

[23] SSH. SSH File Transfer Protocol (SFTP): Get SFTP client & server [Online]. Avaliable: https://www.ssh.com/academy/ssh/sftp-ssh-file-transfer-protocol

[24] M. Lottor, Simple File Transfer Protocol [Online], RFC 0913, doi: 10.17487/RFC0913, 1984, Avaliable: https://www.rfc-editor.org/info/rfc913.

[25] M. Horowitz, S. Lunt, FTP Security Extensions [Online], RFC 2228, doi: 10.17487/RFC2228, 1997, Avaliable: https://www.rfc-editor.org/info/rfc2228.

[26] P. Ford-Hutchinson, Securing ftp with tls [Online], RFC 4217, doi: 10.17487/RFC4217, 2005, Available: https://www.rfc-editor.org/info/rfc4217.

[27] J. WOŁOSZYN, "Trivial file transfer protocol (TFTP)", *Edukacja-Technika-Informatyka*, 2(3), pp. 247-156, 2012.

[28] T. P. Cavaiani, "Tools and techniques for simplifying the analysis of captured packet data", *Journal of Information Systems Education*, 19(4), pp. 375-378, 2008.

[29] K. C. Leung, V. O. Li, D. Yang, "An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges", *IEEE transactions on parallel and distributed systems*, 18(4), pp. 522-535, doi: 10.1109/TPDS.2007.1011, 2007.

[30] I. Strelkovskaya, R. Zolotukhin, J. Strelkovskaya, "Comparative analysis of file transfer protocols in low-bandwidth radionetworks", *Proceedings of International Conference on Applied Innovation in IT*, Anhalt University of Applied Sciences, Vol. 9, No. 1, pp. 27-32, 2021.

[31] Airlines Electronic Engineering Committee, "Loadable Software Standards, ARINC Report 665-3", *Aeronautical Radio, Inc.*, 2551 Riva Road, Annapolis, Maryland 21401-7435, 2005.

[32] L. Bassett, "Introduction to JavaScript object notation: a to-the-point guide to JSON", *O'Reilly Media, Inc.*, 2015.

[33] D. Crockford, The application/json media type for javascript object notation (json) [Online], RFC 4627, doi: 10.17487/RFC4627, 2006, Available: https://www.rfc-editor.org/info/rfc4627.

[34] T. Bray, Ed. The JavaScript Object Notation (JSON) Data Interchange Format [Online], RFC 8259, doi: 10.17487/RFC8259, 2017, Available: https://www.rfc-editor.org/info/rfc8259

[35] N. Nurseitov, M. Paulson, R. Reynolds, C. Izurieta, "Comparison of JSON and XML data interchange formats: a case study", *Caine*, 9, pp. 157-162, 2009.

[36] NXP Semiconductors. i.MX 8M Dual/8M QuadLite/8M Quad Applications Processors Reference Manua. *NXP*, 2020, Avaliable: https://www.nxp.com/docs/en/data-sheet/IMX8MQLQCEC.pdf

[37] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol", *Journal of Open Source Software*, 2(13), 265, doi: 10.21105/joss.00265, 2017.

[38] G. Malkin, A. Harkin, TFTP Option Extension [Online], RFC 2347, doi: 10.17487/RFC2347, 1998, Available: https://www.rfc-editor.org/info/rfc2347

[39] G. Malkin, A. Harkin, TFTP Timeout Interval and Transfer Size Options [Online], RFC 2349, doi: 10.17487/RFC2349, 1998, Available: https://www.rfc-editor.org/info/rfc2349

[40] G. Malkin, A. Harkin, TFTP Blocksize Option [Online], RFC 2348, doi: 10.17487/RFC2348, 1998, Available: https://www.rfc-editor.org/info/rfc2348

[41] N. N. Mohamed, H. Hashim, Y. M. Yussoff, M. A. M. Isa, S. F. S. Adnan, "Compression and encryption technique on securing TFTP packet", *IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE)*, Penang, Malaysia, pp. 198-202, doi: 10.1109/ISCAIE.2014.7010237, 2014.

최 지 환 (Ji-Hwan Choi)
2020년 08월 : 명지대학교 전자통신공학과 공학사
2020년 08월 – 현재 : LIG Nex1 항공연구소 선임연구원
※관심분야 : 항공전자, 실시간 운영체제, 임베디드 시스템, 스마트캐빈 시스템, 데이터 통신 및 네트워크

최 낙 민 (Nak-Min Choi)
2009년 02월 : 충남대학교 컴퓨터공학과 공학사
2011년 02월 : 충남대학교 컴퓨터공학과 공학석사
2011년 01월 – 현재: LIG Nex1 항공연구소 수석연구원
※관심분야 : 항공전자, 실시간 운영체제, 임베디드 시스템, 스마트캐빈 시스템, 데이터 통신 및 네트워크

신 재 권 (Jae-Kwon Shin)
2017년 08월 : 충남대학교 컴퓨터공학과 공학사
2020년 02월 : 충남대학교 컴퓨터공학과 공학석사
2020년 01월 – 현재 : LIG Nex1 항공연구소 선임연구원
※관심분야 : 항공전자, 실시간 운영체제, 임베디드 시스템, 스마트캐빈 시스템, 멀티미디어