https://doi.org/10.7840/kics.2024.49.7.947

군집 운영을 위한 PX4-ROS2 무선 통신 최적화 기술 연구

이 현 규◆, 문 성 태°

Research on PX4-ROS2 Wireless Communication Optimization Technology for Swarm Operation

Hyeon-gyu Lee◆, Sung-tae Moon°

요 약

최근 다양한 분야에서 무인 이동체의 서비스가 증가하면서, 무인 이동체를 활용한 서비스 시스템 개발을 위해 오픈소스 기반 플랫폼인 PX4-ROS2에 대한 관심이 증가하고 있다. 특히, PX4-ROS2의 DDS (Data Distribution Service)를 활용하면 군집 이동체를 위한 네트워크를 유연하게 구성할 수 있고 다양한 기능을 제공하여 주목받고 있다. 그러나 DDS를 활용한 군집 운영은 여러 노드간 초기 설정 데이터 교환 및 탐색을 위해 많은 트래픽이 발생하는 문제가 있다. 본 논문에서는 무인 이동체의 군집 운영 시 초기 탐색 단계에서 발생하는 네트워크 트래픽을 DDS 벤더별로 분석하고, 통신량을 최적화하기 위하여 사전에 필요한 통신 데이터를 내장하여 다수 무인 이동체 운영에 중복되는 데이터 통신을 방지하고 통신 데이터의 변화를 일정 시간마다 공유하여 네트워크 부하를 감소시키는 탐색 방식을 제안한다.

키워드 : ROS2, 데이터 분산 서비스, PX4, 군집 운영, 무선 통신
Key Words : ROS2, DDS, PX4, Swarm Operation, Wireless

ABSTRACT

Recently, there has been a growing interest in the development of service systems utilizing unmanned vehicles across various fields. In this context, the open-source platform PX4-ROS2 has garnered attention for facilitating the development of service systems leveraging unmanned vehicles. In particular, the utilization of PX4-ROS2 DDS (Data Distribution Service) enables the flexible configuration of networks for swarm mobility, garnering attention for its ability to provide diverse functionalities. However, there exists a challenge in swarm operations using DDS, as a substantial amount of traffic is generated during the initial exchange of configuration data and exploration among multiple nodes. This paper, we analyze the network traffic generated during the initial discovery phase of swarm operations of unmanned vehicles across various DDS vendors. We propose a discovery mechanism aimed at optimizing communication volume by embedding necessary communication data in advance to prevent redundant data transmission among multiple unmanned vehicles and by sharing changes in communication data at regular intervals to reduce network load.

※ 본 연구는 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 지역지능화혁신인재양성사업(IITP-2024-2020-0-01462)과 과학기술정보통신부의 재원으로 한국연구재단, 무인이동체원천기술개발사업단의 무인이동체원천기술개발사업(No. 2023M3C1C1A01098416)의 지원을 받아 수행되었습니다.
◆ First Author : Chungbuk University Department of Control and Robot Engineering, herbifors@cbnu.ac.kr, 정회원
° Corresponding Author : Chungbuk University Department of Control and Robot Engineering, stmoon@cbnu.ac.kr, 정회원
논문번호 : 202401-012-B-RN, Received January 9, 2024; Revised March 22, 2024; Accepted April 2, 2024

Ⅰ. 서 론

최근 무인 이동체의 활용이 증가하면서 다양한 분야에서 무인 이동체를 활용한 연구가 활발하게 이루어지고 있다. 특히, 무인 이동체를 이용한 환경을 구현함에 있어 오픈소스 플랫폼인 PX4-ROS2[1]는 이동체를 하나의 노드(Node)로서 사용할 수 있는 DDS (Data Distribution Service)[2] 통신을 이용할 수 있고 노드 간의 네트워크 유연성을 가지고 군집 환경을 구성할 수 있어 많은 관심을 받고 있다. 군집 이동체는 DDS 환경을 통해 도메인에 자유롭게 참여와 탈퇴를 할 수 있고 한 부분에 문제가 생기면 전체 시스템에 영향을 끼칠 수 있는 SPOF (Single Point of Failure)가 없는 유동적인 네트워크 구성에 있어 필수적이며 경로추적과 자동비행 같은 플랫폼의 다양한 기능들을 이용하여 운용할 수 있다. 그 결과 PX4-ROS2는 군집 이동체 운영을 포함하여 다양한 임무 수행을 위한 연구에 많이 활용되고 있다. 하지만, 이동체간 통신에 DDS를 활용하게 되면 하나의 노드로서 동작하기 위한 초기 설정과 노드 간 통신을 위한 추가적인 기능들이 필요하다. 또한, SPOF가 없는 환경에서 상대 노드와의 정보 교환을 위해 모든 정보를 각 노드가 가지고 있어야 하므로 많은 데이터 교환이 이루어질 수밖에 없다. 특히, 군집 이동체의 운영에 있어 각 노드 간의 데이터 교환이 이루어지는 Discovery 과정에서 발생하는 데이터로 인해 네트워크 트래픽이 증가하게 된다. 따라서 무선 환경에서 운영이 되는 무인 이동체의 경우 유선 환경을 기준으로 수행되는 기존 DDS 환경과 다른 최적화가 필요하다.

본 논문에서는 다수의 무인 이동체 군집 운영을 위해 사용되는 PX4-ROS2 플랫폼의 네트워크 트래픽을 분석하였다. 우선, DDS를 구현한 벤더별 성능을 비교하여 Discovery 방식에 따른 네트워크 트래픽을 분석하였다. 그리고 복잡한 군집 환경에서의 트래픽을 분석하기 위해 시뮬레이션 환경에서 트래픽을 분석하고, 실제 환경에서 동일한 시험을 통해 군집 운영의 문제점을 확인하였다. 또한, 군집 이동체 운영 시 초기 Discovery에 의한 네트워크 트래픽을 줄일 수 있는 통신 방식을 제안한다.

본 논문에서 제안하는 방법의 개선점은 다음과 같다

- 초기 Discovery에 필요한 데이터를 내장하고 초기 메타데이터 변화를 공유하지 않아 중복 데이터의 공유를 제한하였다.
- 데이터의 변화가 있을 때 설정한 시간 동안 변화된 데이터를 모아 한 번에 공유하는 배치(Batch) 형식으로 변경하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 관해 기술하고 3장에서는 DDS Discovery 규약[3]을 통해 DDS Discovery에 필수 요건을 파악하며 Simple, Server, Static Discovery방식을 설명한다. 4장에서는 군집 환경을 기준으로 벤더별 Discovery방식에 따른 성능을 비교하고 시뮬레이션 환경과 실제 환경에서 테스트를 진행하였다. 5장에서는 본 논문에서 제안한 군집 이동체 운용 시 초기 부하를 줄일 수 있는 Discovery방식을 설명한다. 6장에서는 Discovery 방식별 성능과 제시한 Discovery 방식을 시뮬레이션 환경에서 시험한 결과에 관해 설명한다. 마지막으로 7장에서는 전체 결론을 말한다.

Ⅱ. 관련 연구

무인 이동체를 운용함에 있어 실시간 데이터 교환을 지원하며 상호 운용성을 갖추고 있는 DDS는 무인 이동체의 핵심적인 역할을 수행하고 있다. DDS는 표준으로 제시하고 있기 때문에 다수의 벤더가 표준에 맞춰 DDS를 개발한 결과 벤더마다 조금씩 다른 성능을 보여주고 있어서 다양한 연구가 진행되고 있다.

Maruyama, Y.는 DDS가 추가된 ROS2 환경에서의 벤더 별 성능을 측정하고 비교하였다[4]. 특히, QoS와 Data 크기에 따른 Capacity를 측정하였고 다양한 벤더에 대한 분석을 진행하여 ROS2 성능을 검증하였다. 또한, Bode, V.를 포함한 연구에서는 벤치마킹 도구인 DDS-Perf를 이용하여 신뢰성과 유연성에 대해 분석하고 QoS (Quality of Service), 패킷 손실, 통신 지연을 고려한 실시간성에 대한 평가를 진행하여 벤더별 최적의 성능을 구현하기 위한 조정을 통하여 성능을 평가하였고[5] Kronaueret al.는 ROS2 미들웨어를 구현하는 세 가지 벤더에 대하여 페이로드 크기, 변화하는 노드 수와 발행 빈도에 대한 성능 분석을 진행하였다[6]. Aartsen, M.은 ROS2에서 사용되는 벤더별 DDS의 설치 편의성과 사용 편의성을 연구하였고 벤더별 상호 운용성과 보안 기능 여부에 따른 성능 저하에 대한 분석을 진행하였다[7]. 하지만, 이러한 벤더별 연구는 DDS Discovery 방식에 따른 구조 분석은 진행되지 않았으며, 다양한 Discovery 방식에 따른 성능의 변화를 고려하지 않았다.

Jeong, Y.는 DDS의 기본적인 Discovery 과정과 Static 방식 그리고 성능 향상을 위한 Discovery 방법을 제안하였다[8]. 제시한 Discovery 방식은 DDS 통신 과정에서 Group 정보를 추가적으로 설정함으로써 PDP (Participant Discovery Protocol) 단계에서 정보를 필

터링 후 제한된 부분만을 도메인으로 설정하여 Discovery 시간과 자원을 줄이는 부분에 중점을 두고 있다. 하지만 제시한 Discovery 방식은 Group이 정해질 수 없는 단일 군집 구조에서는 최적화를 할 수 없다.

Kwon, K.J.는 확장성과 효율성을 고려하여 DDS Discovery 기법을 분석하였다[9]. 분석과정에서 대규모 분산처리를 위한 계층별 Discovery 기법을 제시하여 대규모 분산 서비스에 대한 검증을 진행하였다. 대규모 분산처리를 위하여 블룸 필터를 이용하여 계층적 클러스터 방식으로 구성하였고 각 클러스터의 관리자가 하위 노드를 담당하는 구조로 Discovery 방식을 제안하였다. 하지만, 블룸필터를 이용하는 방식은 클러스터별 도메인 관리자 및 노드 관리자가 필요하고 이러한 방식은 DDS를 이용하는 데 개별 노드만으로 동작할 수 있는 SPOF가 없는 환경을 구할 수 없다.

Ⅲ. DDS Discovery 방식

DDS는 OMG (Object Management Group)에서 제정한 DCPS (Data-Centric Publish-Subscribe)[3]모델을 기준으로 DDS의 동작과 사양이 정의되어 있다. OMG는 벤더에 상관없이 DDS를 사용하는 노드들 사이에 통신할 수 있도록 RTPS (Real-Time Publish Subscribe)라는 프로토콜 표준을 제공하고 있다. DDS-RTPS의 규약 중 Discovery에 관한 부분을 보면 Discovery 모듈은 참여자(Participant), 주제(Topic), 구독(Subscription), 발행(Publication)이 정의되어 있어야 하며 주제와 연관된 QoS를 포함해야 한다. Discovery 과정에서는 PDP와 EDP (Endpoint Discovery Protocol)를 제공해야 한다. PDP는 멀티캐스트 또는 미리 지정된 유니캐스트를 통해 자신의 노드의 정보를 전송하고 PDP 과정에서 참여하고 있는 도메인 ID, 프로토콜 버전, DDS 벤더, 인라인 QoS, 로케이터(Locator) 등 데이터를 교환하여 도메인에 다른 참여자를 인식하고 자신의 존재를 알리는 기능을 한다. EDP는 노드에 정의되어 있는 엔드포인트를 연결하여 각 노드가 가지고 있는 구독과 발행 목록의 주제를 파악하고 상호 운용성을 위한 안정적인 통신을 제공한다.

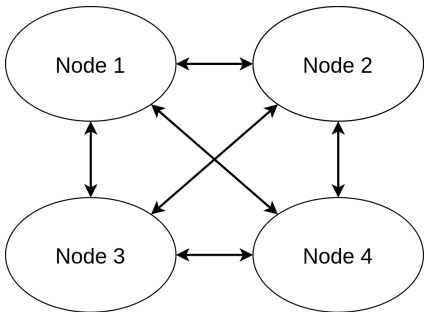

그림 1. Simple Discovery
Fig. 1. Simple Discovery

DDS Discovery의 PDP와 EDP 과정을 나눠 Simple, Server, Static 방식으로 구성할 수 있다. Simple Discovery 방식은 모든 벤더들이 기본적으로 구성되어 있는 Discovery 방식으로 각 노드가 다른 모든 노드들과 독립적으로 통신할 수 있지만 PDP와 EDP 정보 교환으로 많은 트래픽을 발생시키는 방식이다.

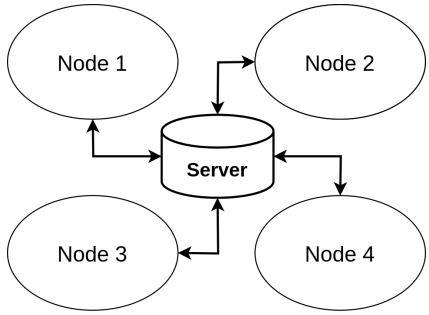

그림 2. Server Discovery
Fig. 2. Server Discovery

Server Discovery 방식은 PDP 부분을 담당할 서버 노드를 생성하여 각 노드의 PDP 통신을 서버 노드가 처리하는 방식이다. 이 방식은 노드 간의 PDP 통신을 서버 노드가 처리하여 트래픽을 감소시킬 수 있지만, 서버 노드 자체가 SPOF가 되는 문제가 존재한다.

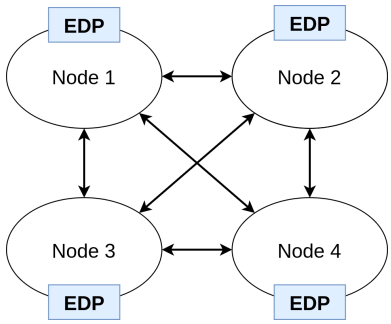

그림 3. Static Discovery
Fig. 3. Static Discovery

마지막으로, Static Discovery 방식은 통신에 필요한 데이터를 사전에 인지하고 있는 환경에서 적용할 수 있다. 노드들이 사전에 필요한 EDP 방식을 공유하면

EDP 과정을 제외하고 PDP 과정만을 수행할 수 있기 때문에 트래픽 감소에 효과적이다.

Ⅳ. Discovery 성능 비교

4.1 시험 환경

다수 무인 이동체 운영 환경에서 Disovery 성능을 검증하기 위해 PX4-ROS2 시스템을 기반으로 성능을 측정해보았다.

시뮬레이션 시험 환경을 구축하기 위해 그림 4와 같이 각 시스템은 PX4-ROS2을 구축하였다. 이때 ROS2는 Humble 버전을 사용하였으며, 개별 노드간의 독립성을 유지하기 위해 도커 컨테이너 기반으로 시뮬레이션 시스템을 구축하였다. 각 컨테이너에서는 도커 내부에 1Hz 간격으로 무인 이동체의 위치 정보인 Odometry[10] 데이터를 전송하는 노드가 무인 이동체 수(N) 만큼 동시에 통신에 참여하고 각 참여자가 20번의 랜덤한 좌표 데이터(Odometry)를 전송 후 종료되는 형식으로 진행하였다. Discovery의 부화를 시험하기 위해 송신자 기준 30개의 노드까지 순차적으로 참여자를 늘려 전체 트래픽 양을 DDS 벤더에 따라 성능을 측정하였다. 이 때 패킷 재전송을 하는 Reliable QoS에 비해 패킷 손실에 대해 재전송을 하지 않아 통신 시 패킷이 적게 발생하는 Best-Effort 방식으로 QoS를 설정하였다. 시험 과정에서 나온 패킷은 PCAP을 이용하여 측정

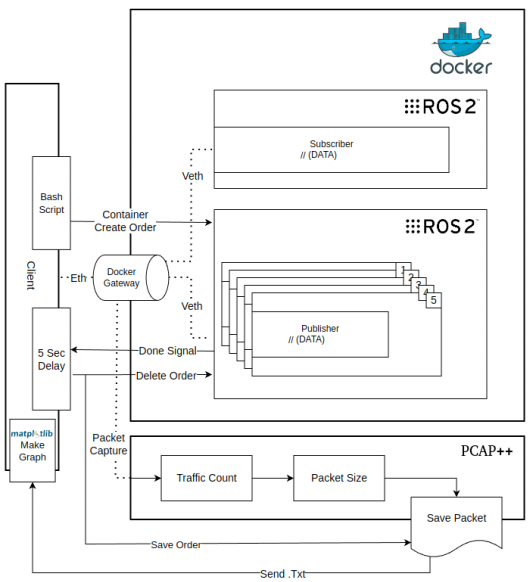

그림 4. DDS 시뮬레이션 환경
Fig. 4. DDS simulation environment

표 1. 시뮬레이션을 위한 DDS 설정
Table 1. DDS Setting for simulation

Publisher frequency	1 Hz
Number of Nodes	1, 2, ..., 29, 30
DDS Back-end	Connext DDS, Cyclone DDS, Fast DDS
Reliability	best-effort
Default Data	Random Odometry x,y,z

하였다. 시험을 위해 AMD Ryzen 5 CPU @ 3.7 GHz와 Ubuntu Linux - Focal(20.04)의 운영체제를 가진 데스크톱에서 진행하였다. 시뮬레이션은 각 벤더의 Simple 방식에 따라 측정하였고 추가적으로 Fast DDS의 Server 방식과 Static 방식의 발생 패킷의 개수와 전체 패킷의 용량을 수집해 결과로 표현하였다.

4.2 DDS 밴더 성능 시험

벤더 별 DDS 성능 시험을 위해 RTI의 Connext DDS[11], Eclipse의 Cyclone DDS[12], Eprosima의 Fast DDS[13]를 사용하였다. 시험을 위해 Simple Discovery 방식을 사용하였다.

수행 결과, 그림 5와 같이 노드의 수가 적은 경우 차이가 미비하지만, 노드의 수가 증가함에 따라 Discovery로 인한 부화가 매우 커지는 것을 볼 수 있었다. 특히, Fast DDS의 Simple 구조의 경우 추가적인 노드가 인식되었을 때 기존 노드의 연결을 확인하는 과정이 존재하여 전체 패킷 양이 다른 벤더에 비해 높았고, 노드가 30개 이상인 경우 정상적인 운영이 어려웠다.

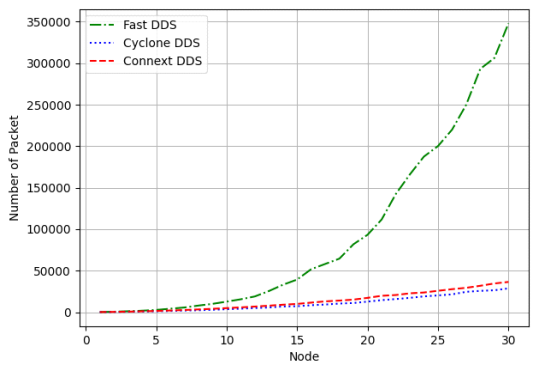

그림 5. DDS 벤더별 Simple Discovery 수행 시 패킷량 비교 실험 결과
Fig. 5. Comparison experiment of packets in Simple Discovery performed by DDS vendors

4.3 Discovery 방식 성능 시험

3장에서 언급한 Discovery 방식에 대한 성능 시험을 수행하였다. 성능 시험을 위해 본 논문에서 제시하는 모든 Discovery 구조를 지원하는 Fast-DDS를 이용하여 Discovery 구조별 시험을 진행하였고, 30개 노드까지 점차적으로 늘려가면서 DDS 수행을 위해 사용된 패킷의 수를 측정하였다. 수행 결과 그림 6과 같이 노드의 수가 증가함에 따라 DDS에 활용되는 패킷 량이 증가함을 확인할 수 있다. 특히, Simple 방식의 경우 Sever와 Static 방식에 비해 급격하게 증가함을 확인할 수 있었다.

그림 7의 부화 테스트 결과를 분석하기 위해 6개의 노드가 각 20개의 랜덤한 Odometry 데이터를 발행하였을 때의 패킷 구조를 분석하였다. 그림 7과 같이 Simple 방식은 PDP, EDP 작업으로 인해 패킷 량이 많은 것을 확인할 수 있었다. 한편, Server 방식의 경우 PDP 작업을 위해 필요한 패킷(PDP) 양이 적어져 전체 패킷 수가 줄어든 것을 확인할 수 있었다. Static 방식의 경우 전체 패킷 수는 EDP 작업을 하지 않아 현저하게

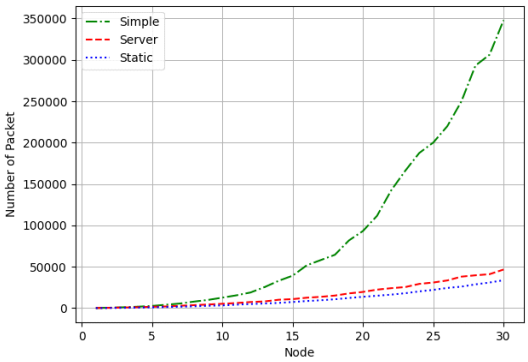

그림 6. Fast DDS의 Discovery 방식에 따른 패킷량 실험 결과
Fig. 6. Experimental results on packets according to discovery mechanism of Fast DDS

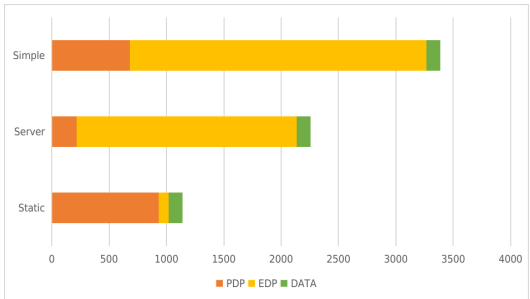

그림 7. Fast DDS의 Discovery 방식에 따른 패킷 구조 분석
Fig. 7. Analysis results of packet types according to discovery mechanism in Fast DDS

패킷량이 줄어들었다. 하지만 PDP을 위해 필요한 패킷량은 오히려 Simple 방식 보다 증가한 것을 확인할 수 있었다. 이는 Static으로 이루어진 Discovery 방식에서 사전에 설정된 EDP 데이터를 로드하면서 노드 내부의 데이터가 변화할 때마다 PDP로 상태 변화를 공유하였기 때문이다[14].

V. 제안하는 Discovery 메커니즘

군집 환경이라는 기준에서 다수의 무인 이동체를 운영하면서 필요한 데이터를 미리 파악할 수 있는 장점이 있다. 따라서, 본 논문에서 제안하는 Discovery 방법은 그림 8과 같이 공동으로 사용되는 EDP에 필요한 데이터를 사전에 노드 내부에 저장하여 EDP 과정 없이 PDP만을 이용해 Discovery를 수행하는 Static 방식 기반으로 탐색하고 통신 제약을 설정할 수 있는 파라미터를 추가하여 Static 방식의 한계점을 극복하였다.

Static 방식의 한계점을 극복하기 위해 제안한 방식은 다음과 같다.

- EDP 데이터 로드가 완료되기 전까지 데이터 공유를 제한한다.
- 모든 노드에 동일하게 로드되는 데이터는 공유를 제한한다.
- 데이터의 변화가 있을 때 설정한 파라미터 시간 동안 변화된 데이터를 모아 한 번에 공유하는 배치(Batch) 형식으로 변경한다.

본 논문에서 제안한 Discovery 메커니즘을 통해 기존 Static 방식의 벤더별 성능측정에서 발생하였던 문제인 노드 내부의 변화에 따른 지속적인 공유를 제한하기 위해 데이터 공유를 제한할 시간을 설정하는 파라미터를 구성하여 데이터의 변화가 존재할 시 설정한 시간 동안 여러 번의 변화가 발생하여도 네트워크 트래픽에

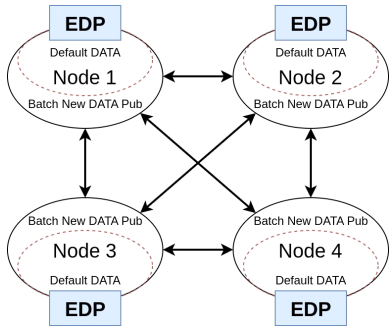

그림 8. 본 논문에서 제안하는 Discovery 방식
Fig. 8. Proposed Discovery mechanism in this paper

부하를 적게 발생할 수 있도록 Discovery 방식을 변경하였다. 제안한 Discovery 메커니즘의 데이터 보유 시간이 증가할수록 패킷량이 감소하지만, 메모리 사용량과 통신 지연 늘어나고 반대로 설정 시간이 감소할수록 패킷량이 증가하지만, 메모리 사용량과 통신 지연이 줄어든다. 본 논문에서 제안하는 Discovery 구조는 1초를 기준으로 동작하고 있다.

Ⅵ. 시험 결과

그림 9의 기존 Fast DDS의 Discovery 방식별 패킷 발생량보다 제안하는 방식의 패킷 발생량이 상당 부분 감소하였고 그림 10을 보면 기존 Fast DDS의 Simple Discovery에 비해 제안하는 방식은 모든 부분에서 데이터가 감소함을 확인할 수 있고 통신에 필요한 데이터를

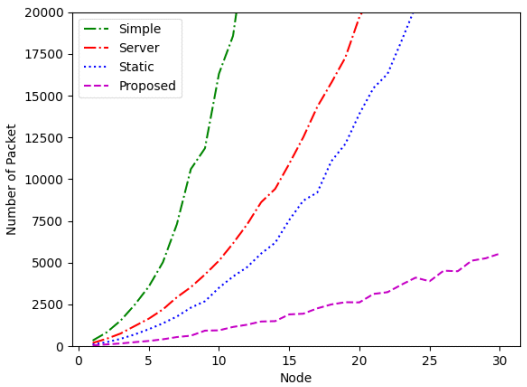

그림 9. 제안한 Discovery 방식의 패킷량 비교 실험 (Fast DDS Discovery 방식 비교)
Fig. 9. Packets comparison experiment of Proposed Discovery mechanism (comparison with Fast DDS discovery mechanism)

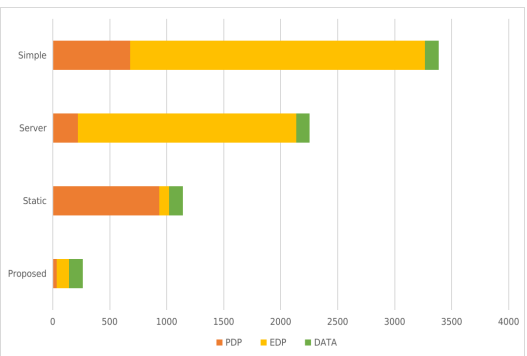

그림 10. 제안한 Discovery 방식의 패킷량 분석 (Fast DDS Discovery 방식 비교)
Fig. 10. Packets analysis of Proposed Discovery mechanism (comparison with Fast DDS discovery mechanism)

Static Discovery와 같이 내장하였기에 EDP 데이터에 있어서 유리함을 확인하였다. 또한 Static Discovery와 EDP 데이터는 비슷하지만 PDP 부분에서 노드 내부 데이터 변화에 따른 데이터를 공유하지 않고 파라미터로 설정된 시간 동안 배치형식으로 데이터를 모아서 전송할 수 있도록 제한하였기에 Static Discovery에 비하여 PDP 패킷의 발생량도 큰 폭으로 감소하였다.

그림 11의 Simple 구조에서 동작하는 벤더별 패킷 발생량을 수집한 패킷 그래프에서도 제안하는 방법이 전체적으로 기존 벤더에 비해 패킷 발생량이 큰 폭으로 감소한 것을 확인할 수 있다.

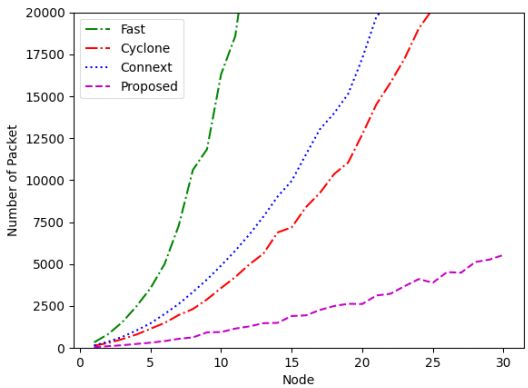

그림 11. 제안한 Discovery 방식의 패킷량 비교 시험 (벤더별 Simple Discovery 방식 비교)
Fig. 11. Packets comparison experiment of Proposed Discovery mechanism (comparison with simple discovery mechanism from vendors)

Ⅶ. 결 론

본 논문에서는 최근 주목받고 있는 PX4-ROS2 군집 운용을 위한 DDS 벤더들의 Discovery 방식에 따른 성능을 분석하고 군집 운영에 최적화된 Discovery 방식을 제안한다. 제안하는 Discovery 방식은 시뮬레이션을 통한 실험을 통해 그 성능을 검증하였다. 다음 연구는 군집 운영을 위한 통신 성능 향상 연구를 수행할 예정이다.

References

[1] PX4, *PX4-ROS2 Documentation*, Retrieved Jan. 8, 2024, from https://docs.px4.io/main/en/ros/

[2] OMG Group, *Data Distribution Services (DDS)* v1.4(2015), Retrieved Jan. 8, 2024, from http://www.omg.org/spec/DDS/1.4/

[3] OMG Group, *The Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (DDSI-RTPS) Specification Version 2.5*.(2022), Retrieved Jan. 8, 2024, from https://www.omg.org/spec/DDSI-RTPS

[4] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in *Proc. 13th Int. Conf. Embedded Softw.*, Oct. 2016. (https://doi.org/10.1145/2968478.2968502)

[5] T. Kronauer, J. Pohlmann, M. Matthé, T. Smejkal, and G. P. Fettweis, "Latency overhead of ROS2 for modular time-critical systems," *arXiv 2021, arXiv preprint arXiv:2101.02074,* 2024.

[6] V. Bode, C. Trinitis, M. Schulz, D. Buettner, and T. Preclik, "DDS implementations as real-time middleware-A systematic evaluation," *IEEE 29th Int. Conf. Embedded and Real-Time Comput. Syst. and Appl.(RTCSA)*, pp. 186-195, Niigata, Japan, Aug. 2023. (https://doi.org/10.1109/RTCSA58653.2023.00030)

[7] M. Aartsen, K. Banga, K. Talko, D. Touw, B. Wisman, D. Meïnsma, and M. Björkqvist "Analyzing interoperability and security overhead of ROS2 DDS middleware," *IEEE 30th Mediterranean Conf. Control and Automat.(MED)* pp. 976-981, Vouliagmeni, Greece, Jun. 2022. (https://doi.org/10.1109/MED54222.2022.9837282)

[8] Y. Jeong, "A study on an improved DDS discovery method for a large-scale system," *J. Korea Soc. Comput. and Inf.*, vol. 25, no. 10, pp. 51-58, Oct. 2020. (https://doi.org/10.9708/jksci.2020.25.10.051)

[9] K. J. Kwon, Y. D. You, and H. Choi, "A scalable and effective DDS participant discovery mechanism," *J. Korea Inst. Inf. and Commun. Eng.*, vol. 13, no. 7, pp. 1344-1356, 2009. (https://doi.org/10.6109/JKIICE.2009.13.7.1344)

[10] ROS Documentation, "*Odometry Data*," Retrieved Jan. 8, 2024, from https://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html

[11] RTI, *Connext DDS*, Retrieved Jan. 8, 2024, from https://www.rti.com/products/connext-dds-professional

[12] Eclipse, *Cyclone DDS*, Retrieved Jan. 8, 2024, from https://cyclonedds.io

[13] eProsima, *Fast DDS*, Retrieved Jan. 8, 2024, from https://www.eprosima.com/index.php/products-all/eprosima-fast-dds

[14] H. G. Lee, G. H. Lim, Y. H. Shin, and S. T. Moon, "Improvement of PX4-ROS2 communication system for drone swarm," in *Proc. Korean Soc. for Aeronautical & Space Sci. Conf.*, vol. 93, Jeju Island, Korea, 2023.

이 현 규 (Hyeon-gyu Lee)

2023년 8월 : 한국기술교육대학교 컴퓨터공학과 졸업
2023년 9월~현재 : 충북대학교 제어로봇공학과 석사
<관심분야> 무인 이동체, 실시간 운영체제
[ORCID:0009-0005-5593-2782]

문 성 태 (Sung-tae Moon)

2005년 2월 : 전남대학교 컴퓨터정보학부 졸업
2007년 2월 : 광주과학기술원 석사
2021년 : KAIST 항공우주공학과 박사
2007년~2010년 : 국방과학연구소 연구원
2010년~2011년 : 국가보안기술 연구소 연구원
2012년~2022년 : 한국항공우주연구원 선임연구원
2022년~2023년 : 한국기술교육대학교 조교수
2023년~현재 : 충북대학교 조교수
<관심분야> 무인항공기 위치 예측, 군집비행, 딥러닝
[ORCID:0000-0002-1638-6898]