1부 6장_ROS1과 2의 차이점으로 알아보는 ROS2의 특징

요약

- ROS2의 기능적 개선점
  - “DDS”
    - Real Time, Security, QoS가 보장됨

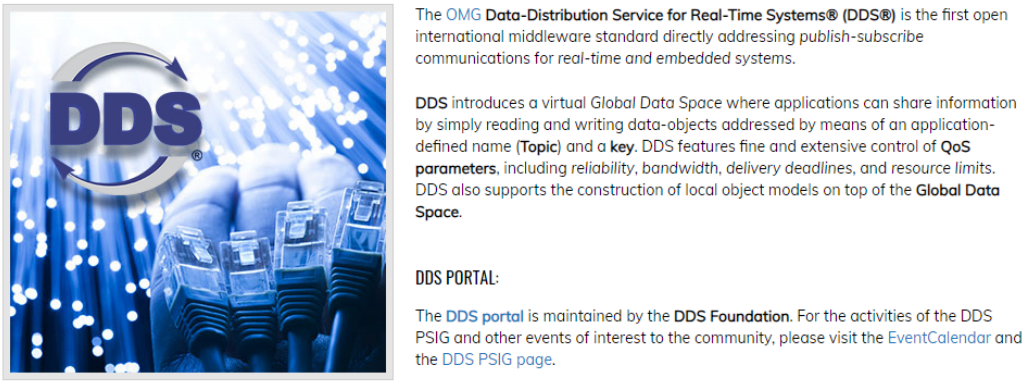
ROS2의 특징

- Platforms
  - 기존 리눅스에서 추가적으로 윈도우, macOS에서도 지원 가능하도록 테스트 중이다.
  - 리눅스 이용자를 위한 우분투 홈페이지의 참고 자료를 활용하자
    - https://ubuntu.com/robotics
    - https://ubuntu.com/blog/topics/robotics

- Real-time
  - 선별된 하드웨어, 리얼타임 지원 운영체제, DDS의 RTPS (Real-Time Publish-Subscribe Protocol)와 같은 통신 프로토콜, 리얼다임 코드 사용을 전제로 실시간성을 지원한다.
    - Doing real-time with ROS2: Capabilities and challenges

- Security
  - ROS1은 TCP/IP 기반의 통신을 사용하여 ROS master가 IP, Port가 노출되면 시스템이 쉽게 공격받는 형태였다.

- ROS2는 TCP/IP 통신 대신 OMG (Object Management Group)에서 산업용으로 사용 중인 DDS (Data Distribution Service)를 도입하였다. 그에 따라 DDS-Security라는 DDS 보안 사양을 적용할 수 있었다.
  - OMG (Object Management Group 객체 관리 그룹)는 분산 객체에 대한 기술 표준을 제정하기 위해 1989년에 설립된 비영리 단체로서 현재는 800여 개 이상의 업체들이 참여하고 있다. 이 단체에는 Oracle, Microsoft, NASA 등이 있다. 그리고 모델링을 위한 새로운 포커스와 모델 기반의 기술을 표준화 한다.
  - DDS (Data Distribution Service): The OMG Data Distribution Service (DDS™) is a middleware protocol and API standard for data-centric connectivity.

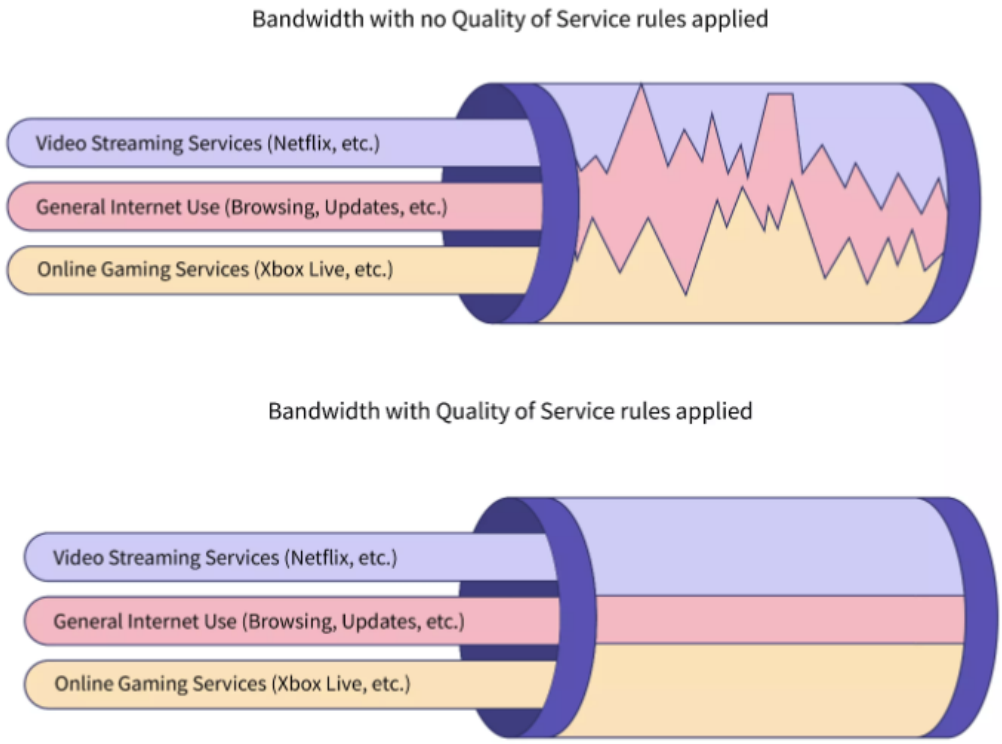


https://www.omg.org/omg-dds-portal/

- SROS2 (Secure ROS2)는 ROS 커뮤니티에서 개발되어 보안 관련 RCL 서포트를 강화했다.
  - 보안용 툴킷이 제공된다.

- Communicaiton
  - ROS1는 TCPROS 통신 라이브러리를 사용하였다.
  - ROS2에서는 DDS를 사용하였다.
    - DDS에서는 RTPS를 지원하여 실시간 데이터 전송이 가능하다.
    - DDS는 노드 간의 자동 감지 기능을 지원하고 있어 ROS1에서 노드를 관리하던 ROS master가 없어도 여러 DDS 프로그램 간에 통신이 가능하다.
    - DDS는 노드 간의 통신을 조정하는 QoS (Quality of Service)를 설정할 수 있다.

- TCP처럼 데이터 전달을 최우선으로 할지, UDP처럼 통신 속도를 최우선으로 할지 선택할 수 있다.
- 네트워크 자원의 우선 순위를 정하여 데이터 전송 시 일정한 성능을 보장해준다.

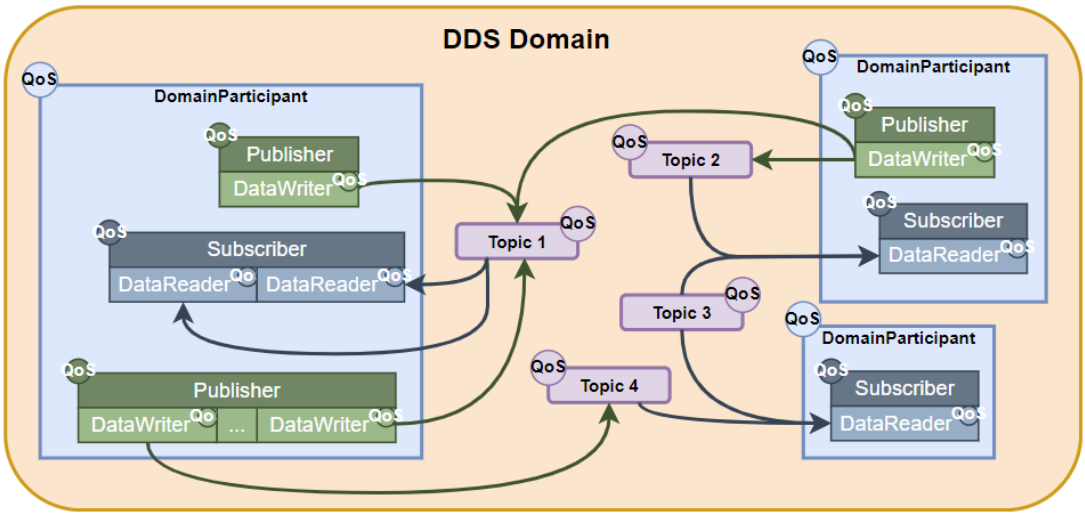


https://www.scaler.com/topics/computer-network/qos/

- Middleware interface
  - DDS는 다양한 기업에서 통신 미들웨어 형태로 제공하고 있다.
  - ROS2에서는 ROS Middleware (RMW) 형태로 DDS 미들웨어를 유저가 원하는 사용 목적에 맞게 선택하여 사용할 수 있도록 지원한다.
    - RMW는 각 벤더들의 DDS API에 대한 추상화 인터페이스를 지원하고 있다.

- Node manager
  - ROS1에서는 roscore를 실행하여 ROS Master, ROS Parameter Server, rowout logging node가 자동적으로 실행되었다.
  - ROS2에서는 roscore가 없어지고 위 3가지 프로그램이 독립적으로 수행되도록 변경되었다.

- ROS Master DDS의 기능들로 대체되었다.
- 노드는 DDS의 Participant로 취급하게 되어 Dynamic Discovery를 이용하여 DDS를 통해 노드를 직접 검색하여 연결할 수 있게 되었다.



https://fast-dds.docs.eprosima.com/en/latest/fastdds/getting_started/definitions.html

- Languages
  - 모던 C++ (C++14), Python 3.5+

- Build system (단일 패키지를 빌드)
  - ROS1에서는 catkin을 사용하였다.
  - ROS2에서는 ament를 사용한다.
    - CMake를 사용하지 않는 파이썬 패키지 관리도 가능하다.
    - C++에서는 ament_cmake, 파이썬에서는 ament_python을 사용한다.

- Build tools (전체 패키지를 빌드)
  - ROS1에서는 여러가지 빌드 도구가 지원되었다.
  - ROS2에서는 ament_tools가 이용되었고 colcon이 추천된다.

- Build options

    - Multiple workspace: 복수의 독립된 워크스페이스를 사용
    - No non-isolated build: 모든 패키지를 별도로 빌드
    - No devel space: 패키지 설치 필요없이 패키지를 사용할 수 있는 devel 공간을 사용하지 않고 패키지를 빌드 후 설치해야 사용할 수 있도록 함

- Version control system
    - 여러 버전 관리 시스템(Git, Mercurial..)을 사용할 수 있도록 vcstool이라는 버전 관리 시스템 툴로 통합하였다.

- Client library
    - ROS Client Library (RCL)로 클라이언트 구성이 가능하다.
        - rclcpp, crlc, rclpy, rcljava, rclobjc, rclada, rclgo, rclnodejs 등

- Lifecycle
    - 패키지의 각 노드들의 현재 상태를 모니터링하고 상태 제어가 가능한 Lifecycle을 클라이언트 라이브러리가 포함되어 있다.

- Mutiple nodes
    - Components (RCL 기능)
        - 동일한 실행 파일에서 복수의 노드를 실행시킬 수 있다.

- Threading model
    - 사용자가 정의한 실행기도 RCL API를 이용하여 스레드를 쉽게 구현할 수 있다.

- Messages (Topic, Service, Action)
    - ROS1과 비슷한 구조의 데이터 구조 (Topic, Service, Action)를 사용할 수 있다. 콘셉트는 변하지 않으나 사용 방법이 많이 바뀌었다.
    - 추가적으로 OMG에서 정의한 IDL (Interface Description Language)를 사용하여 메시지 정의 및 직렬화를 더 쉽고 포괄적으로 다룰 수 있게 되었다.

- Command Line Interface (CLI)
- Launch
  - ROS1에서는 Launch 파일이 특정 XML 형식을 가지고 있었다.
  - ROS2에서는 XML, Python을 지원하여 복잡한 논리와 기능을 자유롭게 사용할 수 있다.

- Graph API
- Embedded systems
  - 시리얼 통신, 블루투스 및 와이파이 통신, RTOS (Real-Time Operating System), DDS-XRCE (DDS-eXtremely Resource Constrained Environments)를 지원하여 임베디드 보드에서 직접 ROS 프로그래밍을 하여 펌웨어로 구현된 노드를 실행할 수 있게 되었다.

참고 사이트

- DDS
  - https://www.dds-foundation.org/
  - https://fast-dds.docs.eprosima.com/en/latest/fastdds/getting_started/definitions.html
- QoS
  - https://www.koit.co.kr/news/articleView.html?idxno=79996