

ORC-PG-ROS2-WEEK2

chapter 5-8

CH5 - WHY ROS 2?



Why ROS 2?



ROS 2 (Robot Operating System 2) is an open source software development kit for robotics applications. The purpose of ROS 2 is to offer a standard software platform to developers across industries that will carry them from research and prototyping through to deployment and production. ROS 2 builds on the success of ROS 1, which is used today in myriad robotics applications around the world.

» Shorten time to market

ROS 2 provides the robotics tools, libraries, and capabilities that you need to develop your applications, allowing you to spend your time on the work that is important for your business. Because it is open source, you have the flexibility to decide where and how to use ROS 2, as well as the freedom to customize it for your needs.

» Designed for production

Drawing on a decade of experience in establishing ROS 1 as the de facto global standard for robotics R&D, ROS 2 was built from the ground up to be industry-grade and used in production, including high reliability and safety critical systems. Design choices, development practices, and project governance for ROS 2 are based on requirements from industry stakeholders.

» Multi-platform

ROS 2 is supported and tested on Linux, Windows, and macOS, allowing seamless development and deployment of on-robot autonomy, back-end management, and user interfaces. The tiered support model allows for ports to new platforms, such as real-time and embedded OSs, to be introduced and promoted as they gain interest and investment.

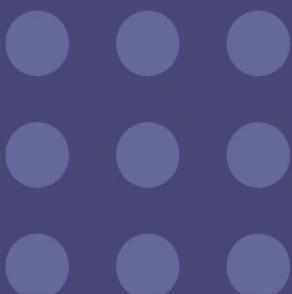
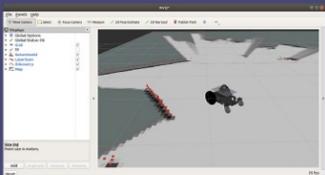
» Multi-domain

Like ROS 1 before it, ROS 2 is ready for use across a wide array of robotics applications, from indoor to outdoor, home to automotive, underwater to space, and consumer to industrial.



www.openrobotics.org www.ros2.org

Why ROS 2?



» No vendor lock-in

ROS 2 is built on an abstraction layer that insulates the robotics libraries and applications from the communication technologies. Below the abstraction are multiple implementations of the communications code, including both open source and proprietary solutions. Above the abstraction, core libraries and user applications are portable.

» Built on open standards

The default communications method in ROS 2 uses industry standards like IDL, DDS, and DDS-I RTPS, which are already widely deployed in a variety of industrial applications, from factories to aerospace.

» Permissive open source license

ROS 2 code is licensed under Apache 2.0 License, with ported ROS 1 code under the 3-clause (or “new”) BSD License. Both licenses allow permissive use of the software, without implications on the user’s intellectual property.

» Global community

Over 10+ years the ROS project has produced a vast ecosystem of software for robotics by nurturing a global community of hundreds of thousands of developers and users who contribute to and improve that software. ROS 2 is developed by and for that community, who will be its stewards into the future.

» Industry support

As demonstrated by the membership of the ROS 2 Technical Steering Committee, industry support for ROS 2 is strong. Companies large and small from around the world are committing their resources to making open source contributions to ROS 2, in addition to developing products on top.

» Interoperability with ROS 1

ROS 2 includes a bridge to ROS 1 that handles bidirectional communication between the two systems. If you have an existing ROS 1 application, you can start experimenting with ROS 2 via the bridge and port your application incrementally according to your requirements and available resources.

 ROS

www.openrobotics.org www.ros2.org

ROS 2는 ROS 1의 성공과 10년이 넘는 기간 동안 쌓은 노하우를 기반으로 하는 차세대 로봇 응용 프로그램을 위한 오픈소스 소프트웨어

개발 키트

ROS 2, 무엇이 특징일까?

1. 시장 출시 시간 단축

ROS 2는 로봇 응용 프로그램을 개발하는 데 필요한 도구, 라이브러리 및 기능을 제공하기 때문에 본연의 로봇 개발에 더 많은 시간을 할애할 수 있다는 큰 장점

[궁금한점] 처음 로봇 개발을 위한 프레임워크 생산, 통신 방법 설정, 디버깅&시각화 툴 설정이라는 비효율에서 벗어날 수 있다고 하는데 어떻게 효율적인지가 궁금

2. 생산을 위한 설계

ROS 2는 처음부터 산업용 수준으로 개발되었고 높은 신뢰성과 안정성 중시

ROS 2의 설계, 개발 및 프로젝트 관리는 업계 실사용자로부터 얻은 실질적인 요구 사항을 기반

3. 멀티 플랫폼

ROS 2는 리눅스, 윈도우, macOS 등 다양한 플랫폼에서 활용 가능

계층형 지원 모델을 통한 실시간 운영체제, 임베디드와 같은 새로운 플랫폼 포팅

4. 다중 도메인

ROS 2는 실내, 실외, 모바일 장치, 수중 및 우주 등등 다양한 로봇 응용 분야에서 사용 가능

5. 벤더 선택 가능

ROS 2는 로봇공학 라이브러리와 응용 프로그램을 통신 기능으로부터 분리하여 추상화 작업 된 상태

사용자 애플리케이션은 사용자가 원하는 형태로 개발, 수정이 가능함

6. 공개 표준 기반

ROS 2 기본 통신 방법은 IDL, DDS, DDS-I RTPS 등 모든 산업을 포함하는 표준 사용

7. 자유 재량 허용 범위가 넓은 오픈소스 라이센스 채택

ROS 2 코드 Apache 2.0 라이센스를 기본 라이센스로 사용하여 지적 재산권에 영향을 주지 않음

8. 활발한 글로벌 커뮤니티

9. 강력한 업계 지원 - ROS 2 Technical Steering Committee

10. ROS 1과의 상호 운용성 확보

CH6 ROS 1과 2의 차이점으로 알아보는 ROS 2

ROS 1의 한계

ROS 1 현재까지도 대학, 연구 기관, 산업계, 로봇 자작 활동까지 폭넓게 이용되고 있지만 2000년대 초반 PR2초기 콘셉트를 그대로 이어받아 다음과 같은 제한 사항이 있음

- 단일 로봇
- 워크스테이션급 컴퓨터
- 리눅스 환경
- 실시간 제어 지원 불가
- 안정된 네트워크 환경 요구됨
- 주로 대학이나 연구소와 같은 아카데믹 연구 용도

그렇다면 이러한 제한 사항에 따라 새로운 로봇 개발환경 및 요구되는 기능은 어떤 것들이 있었을까?

- 두 대 이상의 로봇
- 임베디드 시스템 ROS 사용
- 실시간 제어
- 불안정한 네트워크 환경에서도 동작할 수 있는 유연함
- 멀티 플랫폼 지원
- 최신 기술 지원(Zeroconf, Protocol Buffer, ZeroMQ, WebSockets, DDS 등)
- 상업용 제품 지원

ROS 1과 ROS 2의 차이점

ROS 1과 ROS 2 의 차이점을 표로 확인하면 아래와 같다. 다만 어떻게 보면 전혀 다른 새로운 플랫폼을 만든 것이 ROS 2기 때문에 각 Feature 별 특징이 곧 둘의 ROS 2의 설명이 된다.

(ROS 1 이 좋지 않다는 것은 아님을 강조)

Features	ROS 1	ROS 2
Platforms	Linux, macOS	Linux, macOS, Windows
Real-time	External frameworks like OROCOS	Real-time nodes when using a proper RTOS with carefully written user code
Security	SROS	SROS 2, DDS-Security, Robotic Systems Threat Model
Communication	XMLRPC + TCPROS	DDS (RTPS)
Middleware interface	–	RMW
Node manager (discovery)	ROS Master	No, use DDS's dynamic discovery
Languages	C++03, Python 2.7	C++14 (C++17), Python 3.5+
Client library	roscpp, rospy, rosjava, rosnodejs, and more	rclcpp, rclpy, rcljava, rcljs, and more
Build system	rosbuild → catkin (CMake)	Ament (CMake), Python setuptools (Full support)
Build tools	catkin_make, catkin_tools	Colcon
Build options	–	Multiple workspace, No non-isolated build, No devel space
Version control system	rosws → wstool, rosinstall (*.rosinstall)	vcstool (*.repos)
Life cycle	–	Node life cycle
Multiple nodes	One node in a process	Multiple nodes in a process
Threading model	Single-threaded execution or multi-threaded execution	Custom executors
Messages (topic, service, action)	*.msg, *.srv, *.action	*.msg, *.srv, *.action, *.idl
Command Line Interface	rosrun, roslaunch, ros topic ...	ros2 run, ros2 launch, ros2 topic ...
roslaunch	XML	Python, XML, YAML
Graph API	Remapping at startup time only	Remapping at runtime
Embedded Systems	rosserial, mROS	microROS, XEL Network, ros2arduino, Renesas DDS-XRCE(Micro-XRCE-DDS), AWS ARCLM

ROS 2 특징

1. Platforms (플랫폼)

- ROS 2부터 리눅스, 윈도우, macOS 모두 지원!
- 22년 4월 기준 현재 Ubuntu 22.04.2 LTS & ROS 2 Humble 호환 가능

2. Real-time (리얼타임)

- 실시간성을 지원! 단, 선별된 하드웨어, 리얼타임 지원 운영체제, DDS의 RTPS(Real-time Publish-Subscribe Protocol)와 같은 통신 프로토콜, 리얼타임 코드 사용을 전제로 함

3. Security (보안)

- TCP 기반의 통신은 OMG(Object Management Group)에서 산업용으로 사용 중인 DDS(Data Distribution Service)를 도입하였고, 자연스럽게 DDS-Security라는 보안 사양을 ROS에 적용하여 보안에 대한 이슈를 통신단부터 해결
- SPOS 2 (Secure Robot Operating System 2) 개발하여 보안 관련 RCL 서포트 강화

4. Communications (통신)

- ROS 2는 리얼타임 퍼블리시와 서브스크라이브 프로토콜인 RTPS를 지원하는 통신 미들웨어 DDS사용
- DDS에서는 IDL(Interface Description Language)을 사용하여 메시지 정의 및 직렬화를 더 쉽게, 더 포괄적으로 다룰 수 있음
- DDS는 노드 간의 자동 감지 기능을 지원하고 있어서 기존 ROS 1의 ROS Master가 없어도 여러 DDS 프로그램 간에 통신이 가능
- 노드 간의 통신을 조정하는 Qos(Quality of Service)설정 가능하여 TCP처럼 데이터 손실을 방지하여 신뢰도를 높이거나, UDP처럼 통신 속도를 최우선하여 사용 가능
- 즉, “**실시간 데이터 전송, 불안정한 네트워크 대응, 보안 강화**”

5. Middleware interface (미들웨어 인터페이스의 변화)

- ROS 2 의 DDS는 다양한 기업에서 통신 미들웨어 형태로 제공
- ROS 2 를 지원하는 DDS Vendor : ADLink, Eclipse Foundation, Eprosima, Gurum Network, RTI
- ROS 2에서는 이러한 벤더들의 미들웨어를 유저가 원하는 사용 목적에 맞게 선택하여 사용할 수 있도록 ROS Middleware (RMW) 형태로 지원하고 있음

6. Node manager - Discovery (노드 매니저 - 디스커버리)

- ROS 2에서는 ROS 1의 roscore가 없어지고 3가지 프로그램이 각각 독립적으로 수행 되도록 바뀌었다. 특히, ROS Master의 경우 완전히 삭제되었고, 이는 DDS의 기능들로 대체
- 노드는 DDS의 Participant로 취급, Dynamic Discovery를 이용하여 DDS 미들웨어를 통해 노드를 직접 검색하여 연결 가능

7. Languages (다양한 언어 지원)

- 다양한 언어 지원 예정 (계속 업데이트)
- ROS 1 : C++03, 파이썬 2.7
- ROS 2 : C++14(C++17), 파이썬 3.5+

8. Build System (빌드 시스템의 변화)

- ROS 2에서는 새로운 빌드 시스템인 ament을 사용
- ament는 CMake를 사용하지 않는 파이썬 패키지 관리도 가능
- ROS 2에 와서는 파이썬 패키지가 완전한 독립을 이루게 되었음 ROS 2에서 파이썬 패키지는 `setup.py` 파일의 모든 기능을 순수 파이썬 모듈과 동등한 수준으로 사용할 수 있게 되었음
- **ROS 2 : ament(CMake), Python setup(x)ls(Full support)**

9. Build Tools (빌드 툴 지원 범위 확대)

- ROS 2에서는 알파, 베타 그리고 Ardent 릴리스까지 빌드 도구로 amentools가 이용되었고 지금에 와서는 colcon(collective construction)을 추천
- Colcon : ROS 2 패키지 흐름을 향상시키는 CLI 타입의 명령어 도구. 터미널 창에서 colcon test, colcon build와 같은 명령어 실행 가능

10. Build options (새롭게 변화된 빌드 옵션 3가지)

- **Multiple workspace** : ROS 2는 복수의 독립된 워크스페이스를 사용할 수 있기 때문에 작업 목적 및 패키지 종류별로 이를 관리할 수 있게 되었음
- **No non-isolated build** : ROS 2에서는 이전 빌드 시스템인 catkin의 일부 기능으로 사용되었던 `catkin_make_isolated` 형태와 같은 격리 빌드만을 지원함으로써 모든 패키지를 별도로 빌드 가능. 이 기능 변화를 통해 설치용 폴더를 분리하거나 병합 가능
- **No devel space** : ROS 2에서는 패키지를 빌드한 후 설치해야 패키지를 사용할 수 있도록 변경됨. colcon 사용 시에 “`--symlink-install`”과 같은 옵션을 제공하여 심볼릭 링크 설치가 가능하도록 함

11. Version Control System (버전 관리 시스템의 변화)

- ROS 1 : rosdep — wstool, rosinstall(.rosinstall)
- **ROS 2 : vcstool(.repos)**
- vcstool은 여러 리포지터리 작업을 보다 쉽게 관리할 수 있도록 설계된 버전 관리 시스템 툴

```
wget https://raw.githubusercontent.com/ros2/ros2/foxy/ros2.repos  
vcs import src < ros2.repos
```

12. Client library (클라이언트 라이브러리)

- ROS 2에서는 ROS 클라이언트 라이브러리(ROS Client Library)를 제공. 이는 프로그래밍 언어별로 rclcpp, rclc, rclpy, rcljava, rclobjc, rclada, rclgo, rclnodejs 등으로 제공

13. Lifecycle (라이프사이클)

- 로봇 개발에 있어 로봇의 현재 상태를 파악하고 현재 상태에서 다른 상태로 변경되는 상태전이 제어하는 수십 년간 주요 연구 주제로 다루었던 중요한 부분
- ROS 2는 기존 ROS 1에서 사용한 SMACH와 같은 상태전이를 관리하는 독립적인 패키지에 패키지의 각 노드들이 현재 상태를 모니터링하고 상태 제어가 가능한 Lifecycle을 클라이언트 라이브러리에 포함 시켰음. 이를 통해 ROS 시스템 상태를 보다 효과적으로 제어할 수 있게 되었음
- Lifecycle을 이용하면 기존 ROS 1에서는 할 수 없었던 노드의 상태를 모니터링하고 상태를 천이시킬 수 있고, 노드의 상태에 따라 재시작하거나 교체할 수도 있음

14. Multiple nodes

- ROS 2에서는 기존 ROS 1에서 여러 노드를 실행하기 위한 Nodelet 라이브러리를 사용하지 않고 이 기능을 RCL에 포함
- 이를 컴포넌트(Components)라고 부르며, ROS 2에서는 이 컴포넌트를 사용하여 동일한 실행 파일에서 복수의 노드를 실행시킬 수 있게 되었음.
- 장점 : 노드의 실행 파일 수준을 보다 세분화시킬 수 있으며 프로세스 내 통신 기능을 이용하여 ROS 2의 통신 오버헤드를 제거할 수 있어서 더욱 효율적인 ROS 2 응용 프로그램의 작성이 가능

15. Threading model (스레딩 모델)

- ROS 1에서 사용자가 단일 스레드 실행 또는 다중 스레드 실행 중 하나만 선택할 수 있었다면, ROS 2에서는 더 세분화된 실행 모델(Executor)을 C++와 파이썬에서 사용할 수 있으며 사용자가 정의한 RCL API를 이용하여 쉽게 구현할 수 있음

16. Messages (Topic, Service, Action)

- ROS 2에서도 ROS 1와 동일하게 단일 데이터 구조를 메시지로 정의. 사용자가 정의한 메시지 사용 가능

- 차별점 : OMG(Object Management Group)에서 정의된 IDL(Interface Description Language)을 사용하여 메시지 정의 및 직렬화를 더 쉽게, 더 포괄적으로 다룰 수 있음

17. Command Line Interface (CLI 타입 명령어 사용법)

- ROS 1 : rostopic list
- ROS 2 : ros2 topic list

18. Launch (런치-프로그램 실행 방법의 변화?)

- run : 단일 프로그램 실행. launch : 사용자 지정 프로그램 실행
- ROS 2는 XML 형식 이외에 파이썬을 지원하여 복잡한 논리와 기능을 자유롭게 사용할 수 있게 되었음

1. Graph API

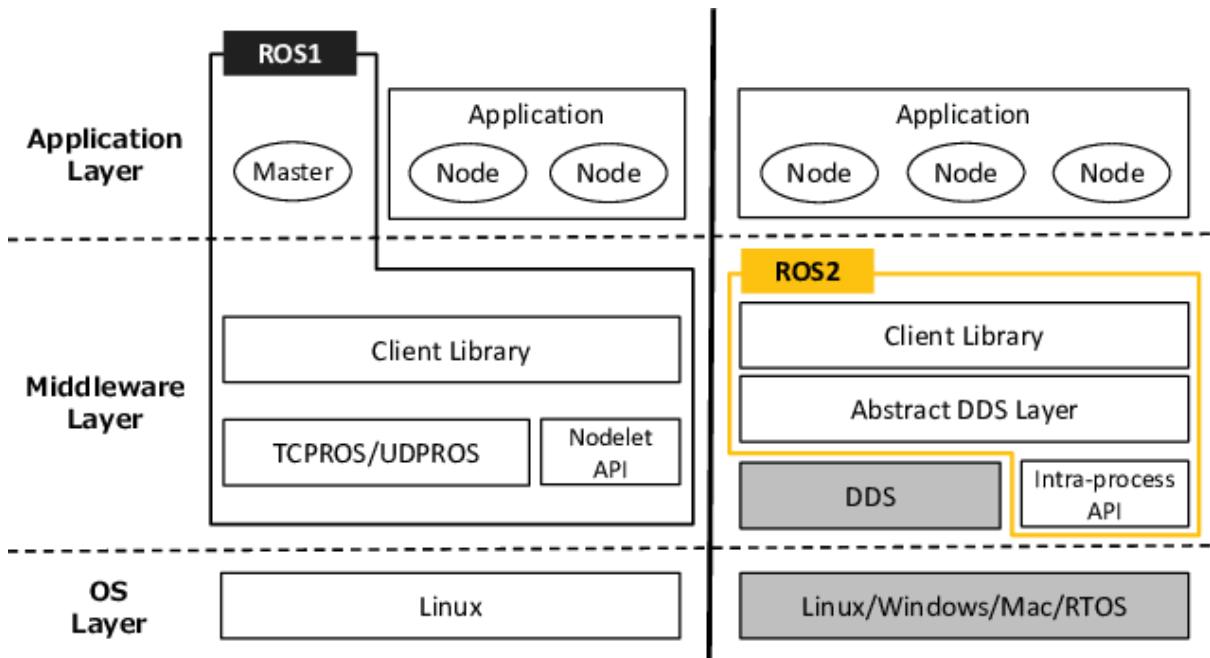
- ROS 1에서는 rqt_graph 사용. 각 노드와 메시지 등이 고유의 이름을 가지고 있고 사로 매핑되어 있어, 노드 간의 메시지 통신 관계를 그래프화 가능 **BUT, ROS 1 에서는 프로그램 시작 시에만 노드와 토픽 간의 매핑이 이루어져서 그 관계를 그래프로 표현할 수 있었기 때문에 실시간으로 변화되는 그래프를 볼 수 없었음**
- ROS 2는 이를 개선하여 노드가 시작할 때뿐만 아니라 실행 도중 재매핑이 가능하며, 그 결과를 바로 그래프로 표현할 수 있게 하려하고 있음 (개발 콘셉트만 나온 상태)

1. Embedded System (임베디드 시스템)

- ?

CH7 ROS 2와 DDS

- ROS 1 은 자체 개발한 TCPROS와 같은 통신 라이브러리 사용
- ROS 2 에서는 OMG(Object Management Group)에 의해 표준화된 DDS(Data Distribution Service)의 RTPS(Real Time Publish Subscribe)를 사용
- DDS를 통신 미들웨어로써 사용하기로 결정한 이유는 산업용 시장 진입을 위한 표준 방식 사용을 중요하게 여겼기 때문

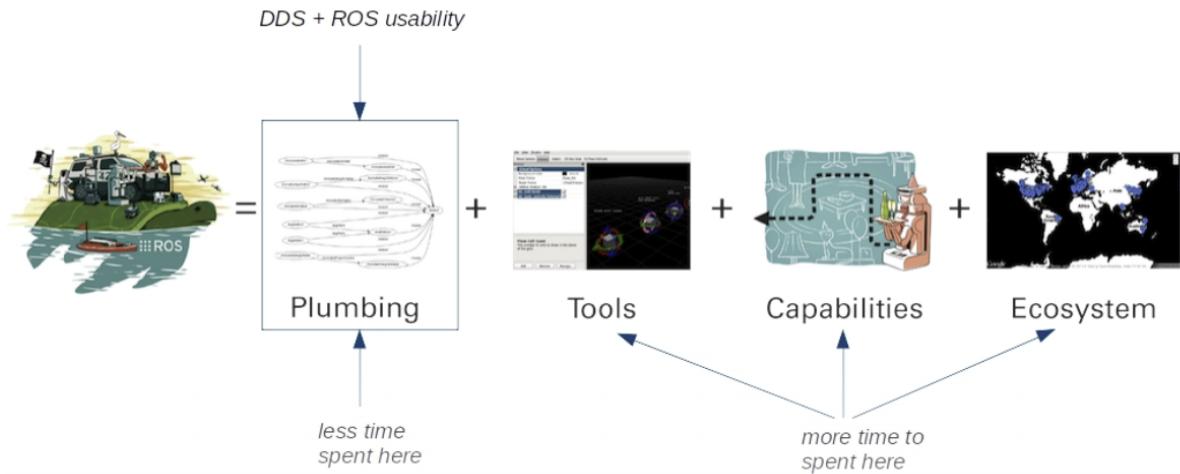


DDS 사용에 따른 ROS의 구조 변화

DDS 사용에 따른 BENEFITS

- ROS 1과 같은 메시지 전달 기능
- 실시간 데이터 전송
- 불안정한 네트워크에 대한 대응
- 보안

“개발자 및 사용자로 하여금 통신 미들웨어에 대한 개발 및 이용 부담을 줄여 주어 더욱 집중 해야 할 부분에 더 많은 시간을 쓸 수 있게 한 아주 고마운 DDS”



DDS란 무엇인가?

DDS는 데이터 분산 시스템의 줄임말로 OMG에서 표준을 정하고자 만든 트레이드 마크.

“데이터 통신을 위한 미들웨어”

- 데이터를 중심으로 연결성을 갖는 미들웨어의 프로토콜(DDSI-RTPS)과 같이 DDS 사양을 만족하는 미들웨어 API
- 이 미들웨어는 ISO 7 계층 레이어에서 호스트 계층(Host layers)에 해당되는 4~7계층에 해당
- ROS 2에서는 운영체제와 사용자 애플리케이션 사이에 있는 소프트웨어 계층으로 이를 통해 통신하면서 데이터를 공유할 수 있음

7계층	응용	HTTP, SMTP, IMAP, POP, SNMP, FTP, TELNET, SSH
6계층	표현	SMB, AFP, XDR
5계층	세션	NetBIOS
4계층	전송	TCP, UDP, SPX
3계층	네트워크	IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IPX, DDP
2계층	데이터 링크	이더넷, 토큰링, PPP, HDLC, 프레임 릴레이, ISDN, ATM, 무선랜, FDDI
1계층	물리	전선, 전파, 광섬유, 동축케이블, 도파관, PSTN, 리피터, DSU, CSU, 모뎀

OSI 7 계층 모델의 계층별 프로토콜

DDS 의 10가지 특징

- 산업 표준 (Industry Standards)
- 운영체제 독립 (OS Independent)
- 언어 독립 (Language Independent)
- UDP 기반의 전송 방식(Transport on UDP / IP)
- 데이터 중심적 기능(Data Centricity)
- 동적 검색(Dynamic Discovery)
- 확장 가능한 아키텍처(Scalable Architecture)
- 상호 운용성 (Interoperability)
- 서비스 품질 (Quality of Service(QoS))
- 보안 (Security)

ROS에서의 사용법

아래의 실행 명령어를 터미널 창 두 곳에서 각각 실행이 잘 된다면 이미 지정된 DDS, 즉 RMW(ROS Middleware)를 사용하고 있다는 뜻.

```
$ ros2 run demo_nodes_cpp listener
```

```
$ ros2 run demo_nodes_cpp talker
```

```
$ ros2 run demo_nodes_cpp listener
[INFO]: I heard: [Hello World: 1]
[INFO]: I heard: [Hello World: 2]
[INFO]: I heard: [Hello World: 3]
[INFO]: I heard: [Hello World: 4]
[INFO]: I heard: [Hello World: 5]
(생략)
```

```
$ ros2 run demo_nodes_cpp talker
[INFO]: Publishing: 'Hello World: 1'
[INFO]: Publishing: 'Hello World: 2'
[INFO]: Publishing: 'Hello World: 3'
[INFO]: Publishing: 'Hello World: 4'
```

** 아래 기능을 몰라 수 분 동안 그냥 두었음...

터미널상에서 각 노드를 종료시키기 위해서는 “Ctrl + c”를 누르면 된다.

RMW 변경 방법

기본적으로 rmw_fastrtps_cpp가 사용되지만(ros2 foxy 기준), 만약 RMW변경을 원하면 ROS 2를 지원하는 RMW 중에서 하나를 RMW_IMPLEMENTATION 환경변수로 지정한 후 노드 실행

(알파벳 순으로 나열)

- rmw_connex_cpp
- rmw_cyclonedds_cpp
- rmw_fastrtps_cpp
- rmw_gurumdds_cpp

** Fast RTPS는 2.0부터 Fast DDS로 명칭이 변경되었으나 현재 버전에서는 아직 Fast RTPS 이름을 사용하고 있음. 차후 변경

RMW 상호 운영성 테스트

위 지원 RMW 중 두 개 선택하여 진행

```
$ export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp  
$ ros2 run demo_nodes_cpp listener
```

```
$ export RMW_IMPLEMENTATION=rmw_fastrtps_cpp  
$ ros2 run demo_nodes_cpp talker
```

Domain 변경 방법

```
$ export ROS_DOMAIN_ID=11  
$ ros2 run demo_nodes_cpp talker
```

```
$ export ROS_DOMAIN_ID=11
$ ros2 run demo_nodes_cpp talker
[INFO]: Publishing: 'Hello World: 1'
[INFO]: Publishing: 'Hello World: 2'
[INFO]: Publishing: 'Hello World: 3'
(생략)
```

```
$ export ROS_DOMAIN_ID=12
$ ros2 run demo_nodes_cpp listener
```

```
$ export ROS_DOMAIN_ID=12
$ ros2 run demo_nodes_cpp listener
(서로 다른 도메인을 사용하고 있기에 아무런 반응이 없을 것이다.)
```

```
$ export ROS_DOMAIN_ID=11
$ ros2 run demo_nodes_cpp listener
```

```
$ export ROS_DOMAIN_ID=11
$ ros2 run demo_nodes_cpp listener
[INFO]: I heard: [Hello World: 13]
[INFO]: I heard: [Hello World: 14]
[INFO]: I heard: [Hello World: 15]
(생략)
```

QoS 테스트 - PASS

CH8 DDS의 QoS

DDS의 서비스 품질(QoS, Quality of Service)

QoS ? - 데이터 통신 옵션

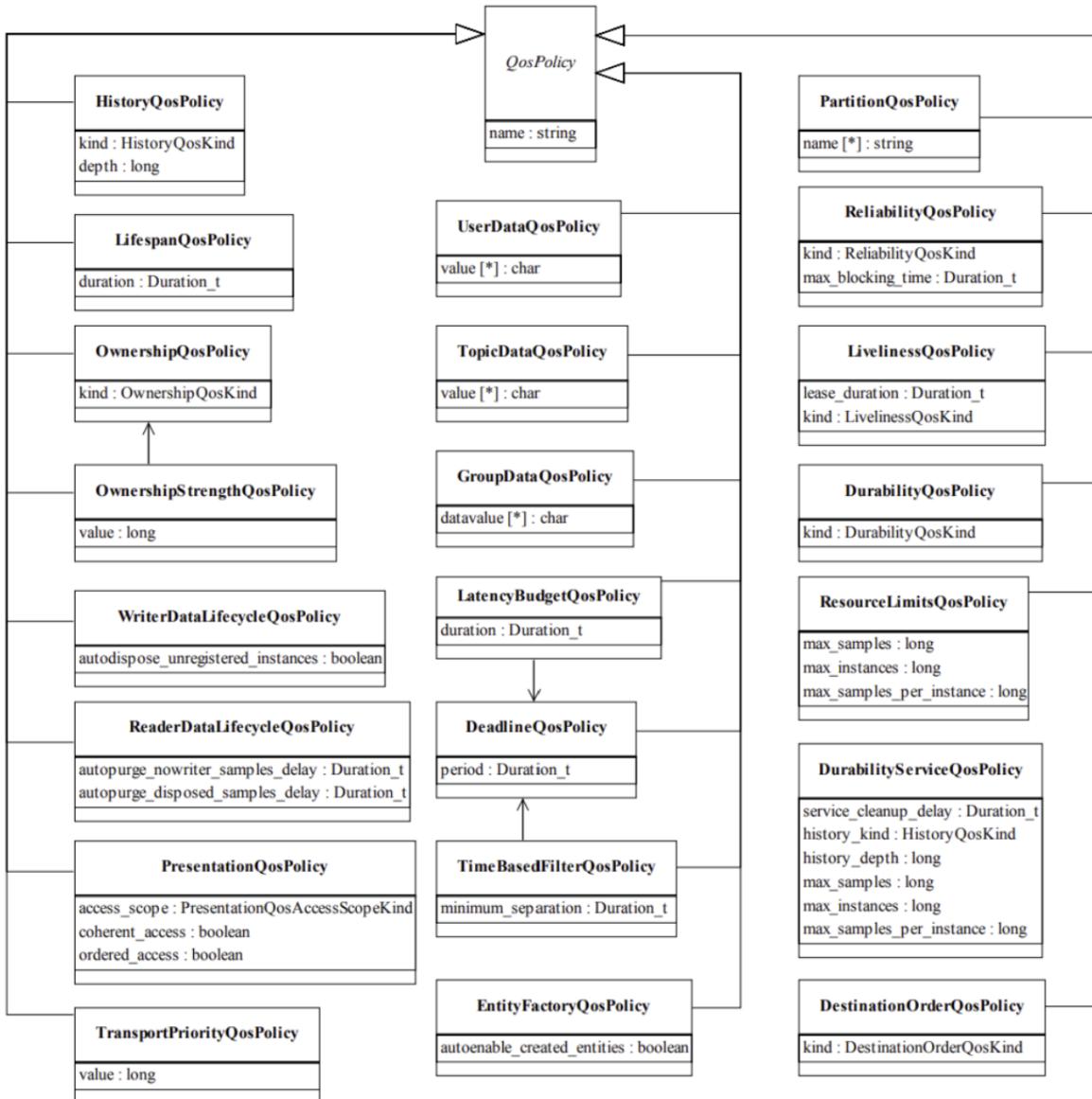
- ROS 2에서는 TCP처럼 신뢰성을 중시 여기는 통신 방식과 UDP처럼 통신 속도에 포커스를 맞춘 통신 방식을 선택적으로 사용할 수 있음

- 이를 위해 ROS 2에서는 DDS의 QoS를 도입하였고 퍼블리셔나 서브스크라이브 등을 선언할 때 QoS를 인자로 지정할 수 있어서 원하는 데이터 통신의 옵션 설정을 유저가 직접 할 수 있게 되었음
- QoS를 통해 변경 가능한 것들 — 데이터 전송 시 실시간성, 대역폭, 지속성, 중복성과 관련된 옵션 등

QoS의 종류 및 ROS 2에서 사용되는 QoS 옵션

22가지의 종류가 있지만 ROS 2에서는 Reliability가 대표적으로 사용

- **Reliability** : 데이터 전송에 있어 속도를 우선시 하는지 신뢰성을 우선시 하는지를 결정하는 옵션
- **History** : 데이터를 몇 개나 보관할지 결정하는 옵션
- **Durability** : 데이터를 수신하는 서브스크라이버가 생성되기 전의 데이터를 사용할 것인지에 대한 옵션
- **Deadline** : 정해진 주기 안에 데이터가 발신 및 수신 되지 않을 경우 Eventcallback 을 실행시키는 옵션
- **Lifespan** : 정해진 주기 안에서 수신되는 데이터만 유효 판정, 나머지는 삭제 옵션
- **Liveliness** : 정해진 주기 안에서 노드 혹은 토픽의 생사 확인 옵션



RMW_qos_profile 사용과 유저 QoS 프로파일 사용

- ROS 2의 RMW에서 QoS 설정을 쉽게 사용할 수 있도록 가장 많이 사용하는 QoS 설정을 세트로 표현해둔 것이 있는데 이를 RMW QoS Profile이라고 표현
- 목적에 따라 Default, Sensor Data, Service, Action Status, Parameters, Parameter Events와 같이 6가지로 구분하며 Reliability, History, Depth (History Depth), Durability를 설정

	Default	Sensor Data	Service	Action Status	Parameters	Parameter Events
Reliability	RELIABLE	BEST EFFORT	RELIABLE	RELIABLE	RELIABLE	RELIABLE
History	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST
Depth (History Depth)	10	5	10	1	1,000	1,000
Durability	VOLATILE	VOLATILE	VOLATILE	TRANSIENT LOCAL	VOLATILE	VOLATILE

QoS 테스트 -PASS