2W_Ros

5. Ros 이유

1. 시장 출시 시간 단축 : 오픈 소스를 활용하여 로봇 개발의 프레임워크,통신,툴 개발의 시간 단축
2. 생산을 위한 설계 : ROS1 과 달리 실제 생산에 이르기 위해 업계 관계자들의 요구 사항 기반설계
3. 멀티플랫폼 : 리눅스, 윈도우, macOS 의 지원과 백엔드 관리, 사용자 인터페이스 개발, 배포 가능
4. 다중 도메인 : 실내,외 자동차, 수중, 우주등의 로봇 응용 분야에 사용 가능
5. 벤더 선택 기능 : 로봇 라이브러리, 통신 기능을 추상화하여 오픈,독점 솔루션의 다양한 방법 제공
6. 공개 표준 기반 : IDL , DDS , DDS-I RTPS(Real time r/sub) 과 같이 산업표준을 사용
7. 자유 재량 허용 범위가 넓은 오픈소스 라이브러리 채택 : Apache 2.0 기본 라이센스 사용
8. 글로벌 커뮤니티 : ROS S/W 기여, 개선의 자의의 개발자들로 구성된 에코시스템을 육성
9. 산업지원 : 전세계 기업들이 제품 개발에이어 오픈소스를 기여 및 자원 투입
10. ROS 1과의 상호 운용성 확보 : Ros1,2 간의 통신을 위해 Ros2에 브리지 포함

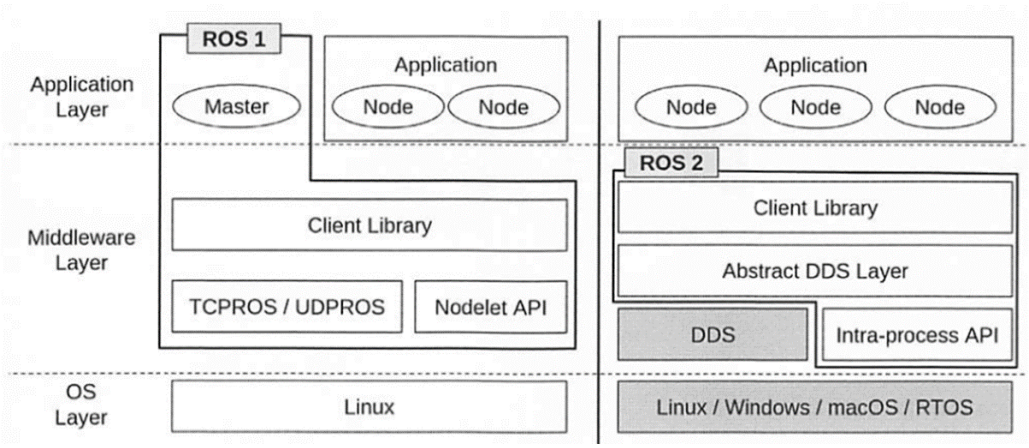
6. Ros1 vs Ros2 차이점 & 특징

- **Ros1 history :**

Willow Garage 사가 2000년대 초반 개인 서비스 로봇 PR2 개발용 미들웨어 형태의 로봇 개발 프레임워크의 개발 툴과 소스를 오픈 소스 공개가 시작의 시초

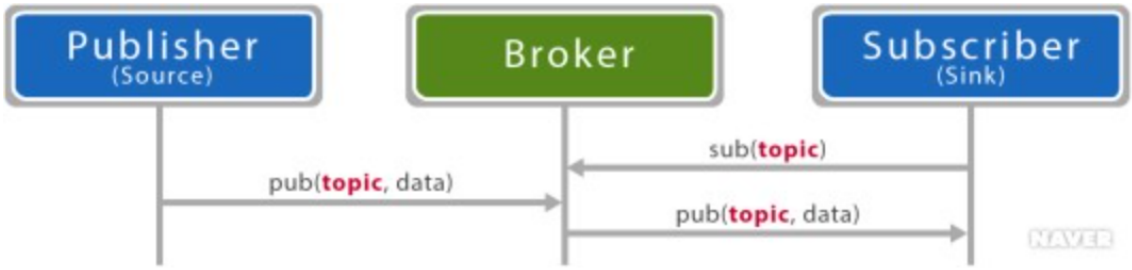
Ros 1 제한사항	제한 사항의 개발 필요	REMARK
단일 로봇 가능	두대 이상의 로봇	1. 제한 사항 개발시 API 수정 필요
워크 스테이션급 Computer 필요	임베디드 시스템에서 사용	2. Ros1 분리 개발
리눅스 환경	멀티 플랫폼((Win, Mac OS)	3. 기존 Ros1 메세지 통신 가능한
실시간 제어 한계	실시간 제어	브릿지 프로그램 제공
네트워크의 안정성 필요	유연성 확보 (불완전한 네트워크 환경)	
연구소, 아칻데믹 연구 용도	상업성 제품 , 최신기술 지원	

- **DDS 사용에 따른 ROS의 구조 변화**



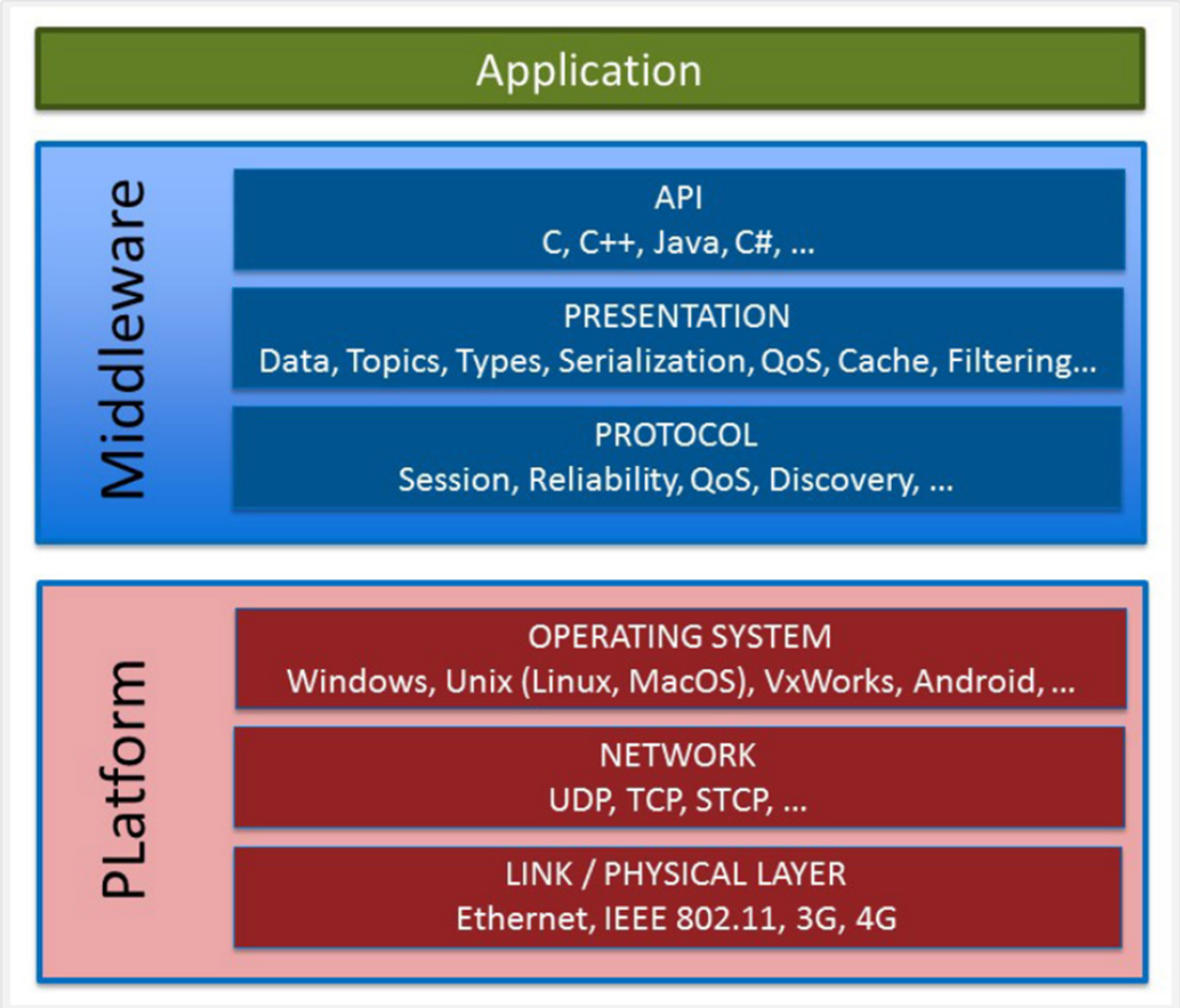
ROS1은 독자적인 TCPROS를 통해 통신을 하고 ROS1에서 사용했던 방식인 MQTT는 TCP 기반으로 브로커를 통해 메시지를 전달에서 ROS2는 DDS의 UDP 기반으로 중간 매개체가 없는 구조로 변

- .MQTT (Message Queueing Telemetry Transport)



MQTT의 Publish/Subscribe 개념

- **미들웨어의 DDS**



DDS는 데이터 분산 서비스라는 개념을 나타내는 단어이다.

-. 데이터를 중심으로 연결성을 갖는 미들웨어의 프로토콜(DDSI-RTPS)과 같이 DDS 사양을 만족하는 미들웨어 API가 그 실체.

-. 미들웨어는 레이어에서 호스트 계층(Host layers)에 해당되는 4~7계층에 해당되고 .ROS2에서는
운영체제와 사용자 애플리케이션 사이에 있는소프트웨어 계층이며 통신으로 데이터를 공유.

- **Ros1 vs Ros 2 차이점**

-

구성	ROS1	ROS 2	특 징
Platforms	Linux, mac OS	Linux, mac OS, Windows	Windows OS가능
Real-time	External frameworks / OROCOS	Real -time nodes when using a proper RTOS with cafefully written user code	
security	SROS	SROS 2, DDS-Security, Robotic System Threat Modeel	
Communication	XMLRPC+TCPTOS	DDS(RTPS)	DDS 사용Real-time pub/sub
Middleware interface	-	RMW	Middleware layer 활용
Node manager	ROS Master	No, use DDS's dynamic discovery	

Languages	C++03, PYTHON 2.7	C++14, Python 3.5+	
Client library	roscpp, rospy, rosjava, rosnodejs,	rclcpp, rclpy, rcljava, rcljs and more	
Build system	rosbuild → catkin(CMake)	Ament(CMake), python setuptools	
Build tools	catkin_make, catkin_tools	Colcon	Build 변경
Build options	-	Mutiple workspaces, No non-isolated. 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/Multi-threaded execution	Custom executors	
Messages (topic,service,action)	*.msg , *srv, *action	*.msg , *srv, *action, *idl	idl : 이종 언어 통신의 인터페이스 추가,
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 System	rosserial, mROS	microROS, ros2arduino,Renesas . .	

Ros 2와 DDS

- 산업표준

  DDS 는 분산객체 기술 표준 제정 단체인 OMG (Object Management Group , 객체 관리 그룹)가 관리함으로 산업 표준으로 자리잡고 있음.

- 운영체제 독립

  DDS는 Linux, Windows, macOS, Androiid등 운영체제를 지원 하므로 사용하던 운영체제를 변경할 필요가 없다

- 언어 독립

  DDS는 프로그래밍 언어를 변경할 필요가 없고 DDS를 RMW(Ros middleware)로 추상화하여 벤더 별로 FastRTPS , RTI Connext , RTI Connext Dynamic을 지원하는 유연성 있음

- UDP 기반의 전송 방식

  UDP의 멀티케스트는 브로드캐스트 처럼 특정 소속된 도메인 그룹(ROS_DOMAIN_ID)에 대해서만 일괄적으로 데이터를 전송 (UDP : 전송 목적 , TCP : 수신 Check)

- 데이타 중심적 기능

  DDS 사양에도 DCPS(Data-centric-publish-subscribe)라는 적절한 수신자에게 적절한 효율적으로 전달하는 것을 목표로 하는 퍼블리시, 서브스크라이브 방식이다

- 동적검색

지정된 네트크네 도메인 내에서 어떤 노드가 어떤 토픽을 발행하는지 수신하는지 DDS의 동적 검색으로 알수있는 기능 이고

- 확장 가능한 아키텍쳐

  DDS의 참가자 형태의 노드는 확장 가능한 형태로 제공 및 사용 가능하여 복잡성을 분산 시스템으로 단순화 하여 다량의 노드 관리가 필요한 곳에 적합하다

- 상호 운용성

  DDS의 표준 사양을 지키고 있으면 A제품에서 B사 제품 변경시 사용이 가능하고 A제품과 B제품의 상호 통신도 지원한다

- 서브스 품질

데이타의 송수신 관련 설정을 유저가 통신 설정을 직접할수 있게 했고 Reliablity , Best effort 등의 기능들이 사용되고 있다.

- 보안

DDS - security 보안 사양을 적용한 SROS 2(Secure R O System) 툴을 개발하여 보급 사용되고 있다

- **RMW의 상호 운영성 테스트**

RMW_IMPLEMENTATION=rmw_cyclones_cpp test 안될시 Eclipse cylcone 설치가 안된 상태

```
Error getting RMW implementation identifier / RMW implementation not installed (expected identifier of 'rmw_connext_cpp'), exiting with 1.
```

Eclipse Cyclone DDS (설치SITE)

Eclipse Cyclone DDS — ROS 2 Documentation: Foxy documentation

- DOMAIN_ID 변경

.bashrc file DOMAIN과 다름 (ID=7 사용) or nano ~/..bashrc 에서 ID=11 수정

DDS의 QoS

- DDS의 서비스 품질 (QoS)

DDS의 QoS(Quality of Service)는 데이터 통신 옵션으로 ROS2 에서는 Publish & Subscribe을 선언시 QoS내에서 유저가 테이터 통신 옵션을 설정할수 있다.

Type	ROS 1	ROS 2
Prootocol	자체 프로토콜 TCPROS	TCP : 신뢰성 중심 (수신 CHECK) UDP : 통신속도 중심 ,

- **QoS 의 종류 (설정 옵션)**

1. History

History	데이터를 몇 개나 보관할지를 결정하는 옵션
KEEP_LAST	정해진 메세지 큐 크기만큼의 데이타 보관 (@ depth : 메세지 큐의 크기 / 설정시만 유효)
KEEP_ALL	모든 데이타를 보관

1-1. Examples

[rclcpp] : rclcpp::QoS(rclcpp::KeepLast(10));

[rclpy] : qos_profile = QoSProfile(history=QoSHistory=QoSHistoryPolicy.KEEP_LAST, depth=10)

2. Reliabilty

Reliabilty	데이터 전송에 있어 속도를 우선시 하는지 신뢰성을 우선시 하는 옵션
BEST_EFFORT	데이터 송신에 집중, 전송 속도를 중시하며 네트워크 상태에 따라 유실이 발생할수 있음
RELIABLE	데이터 수신에 집둥, 신뢰성을 중시하며 유실이 발생하면 재전송을 통해 수신을 보장

2-1. RxO(Requested by Offered)

Pub(ㅣ)/Sub(-)	BEST_EFFORT	RELIABLE
BEST_EFFORT	BEST_EFFORT	불가
RELIABLE	BEST_EFFORT	RELIABLE

2-2. Examples

[rclcpp] : rclcpp::QoS(rclcpp::KeepAll.best_effort());

[rclpy] : qos_profile = QoSProfile(reliability=QoSReliabilityPolicy.BEST_EFFORT)

3. Durability

Durability	데이터를 수신하는 서브스크라이버가 생성되기 전의 데이터를 사용할 것인지에 대한 옵션
TRANSIENT_LOCAL	Subscription 이 생성되기 전의 데이터도 보관(Publish만 가능)
VOLATILE	Subscription 이 생성되기 전의 데이터는 무효

3-1. RxO(Requested by Offered)

Pub(ㅣ)/Sub(-)	TRANSIENT_LOCAL	VOLATILE
TRANSIENT_LOCAL	TRANSIENT_LOCAL	VOLATILE
VOLATILE	불가	VOLATILE

3-2. Examples

[rclcpp] : rclcpp::QoSrclcpp::KeepAll.transient_local();

[rclpy] : qos_profile = QoSProfile(durability=QoSDurabilityPolicy.TRANSIENT_LOCAL)

4. Deadline

Deadline	정해진 주기ㅇ 안에 데이터가 발신 ,수신 되지 않을 경우 EventCallback을 실행 하는 옵션
deadline_duration	Deadline을 확인하는 주기

4-1. RxO(Requested by Offered)

Pub(ㅣ)/Sub(-)	1000ms	2000ms
1000ms	가능	가능
2000ms	불가	가능

4-2. Examples

[rclcpp] : rclcpp::QoS(10).deadline(100ms);

[rclpy] : qos_profile = QoSProfile(depth=10, deadline=Duration(0.1))

5. Lifespan

Lifespan	정해진 주기 안에서 수신되는 데이터만 유효하고 그외에는 삭제하는 옵션
lifespan_duration	Lifespan을 확인하는 주기

5-1. Examples

[rclcpp] : rclcpp::QoS(10).reliable().transient_local().lifespan(10ms);

[rclpy] : qos_profile = QoSProfile(lifespan=Duration(0.01))

6. Liveliness

Liveliness	정해진 주기 안에서 노드 혹은 토픽의 생사를 확인하는 옵션
liveliness	자동/메뉴얼로 할지 지정 3종류 (AUTOMATIC, MANUAL_BY_NODE, MANUAL_BY_TOPIC)
lease_duration	Liveliness을 확인하는 주기

6-1. RxO(Requested by Offered)

Pub(ㅣ)/Sub(-)	AUTOMATIC	MANUAL_BY_NODE	MANUAL_BY_TOPIC
AUTOMATIC	가능	불가	불가
MANUAL_BY_NODE	가능	가능	불가
MANUAL_BY_TOPIC	가능	가능	가능

automatic : 생존성을 관리하는 주체가 DDS 미들웨어

manual : 생존성을 관리하는 주체가 사용자

6-2. Examples

[rclcpp] : rclcpp::QoS qos_profile(10);

qos_profile

.liveliness(RMW_QOS_POLICY_LIVELINESS_AUTOMATIC)

.liveliness_lease_duration(1000ms)

[rclpy] : qos_profile = QoSProfile(liveliness=AUTOMATIC,

liveliness_lease_duration=Duration(1.0))

- **rmw_qos_profile 사용과 유저 QoS 프로파일**

	Default	Sensor Data	Service	Action Status	Parameters	Pram Event
Reliabbilty	RELIABLE	RELIABLE	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
Durabilty	VOLATILE	VOLATILE	VOLATILE	TRANSIENT LOCAL	VOLATILE	VOLATILE