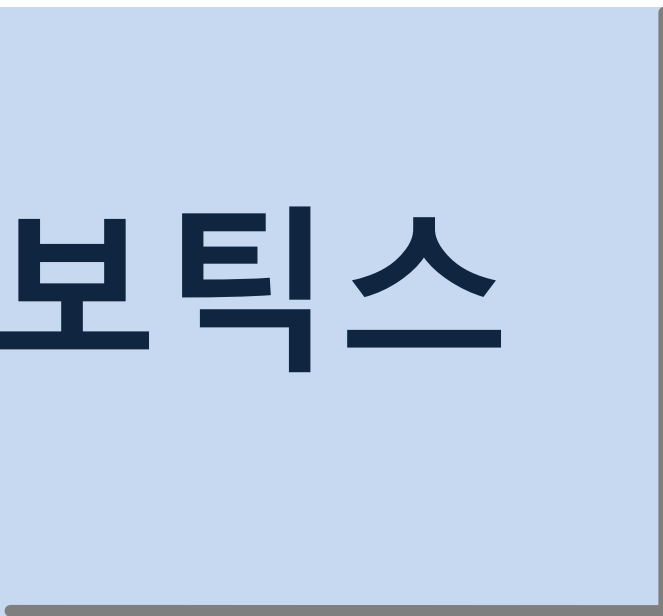




로보틱스



로보틱스(Robotics)란?



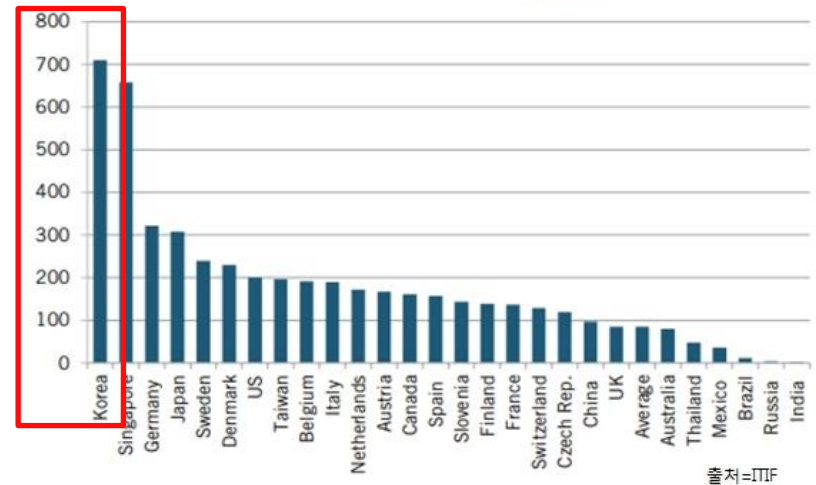
로봇의 종류

• 산업용 로봇

- 산업 자동화에 사용되는 로봇
- 주로 바디부(manipulator), 말단부(end effector)로 구성되어 있음
- 국제로봇연맹(IFR)에 따르면 전세계 산업용 로봇 밀도는 2015년 노동자 1만명당 66대에서 2017년 85대로 15% 증가함. 한국이 1만명당 710대로 8년째 압도적인 세계 1위를 차지함

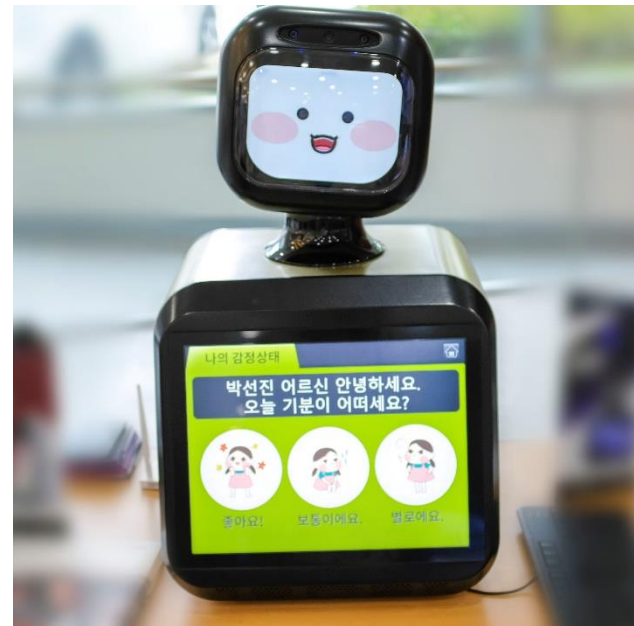


각국의 로봇 밀집도 (노동자 1만명당 로봇 대수)



로봇의 종류

- 서비스 로봇
 - 산업용이 아닌 사람과 환경 개선에 도움을 주는 로봇
 - HRI(Human Robot Interaction) 기술



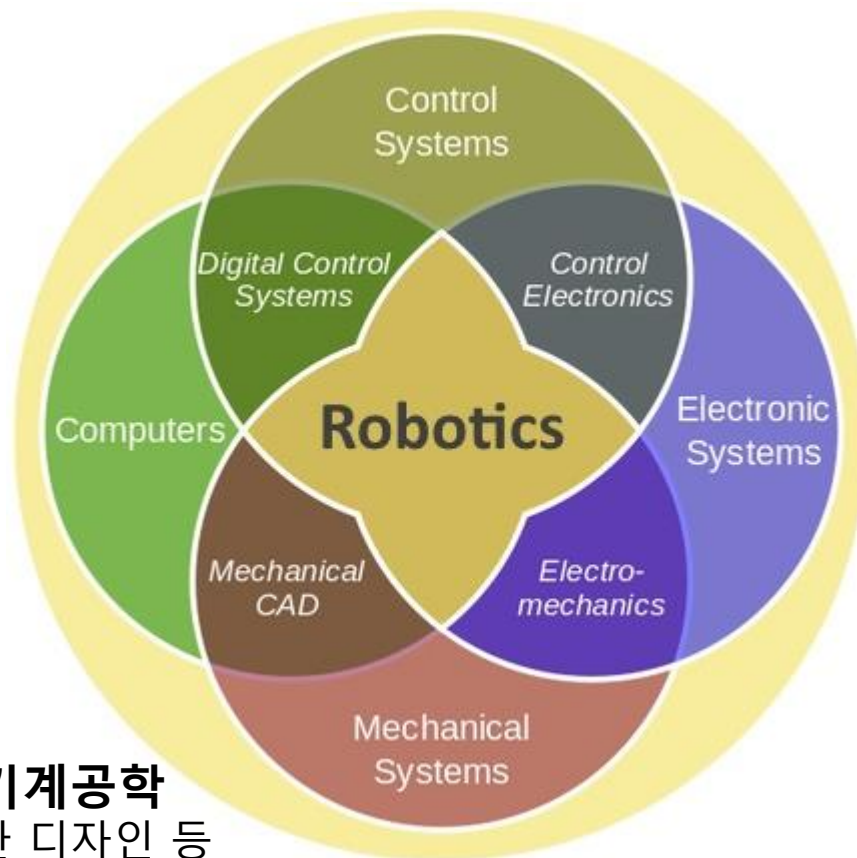
- 그 외 군사용 로봇, 우주탐사용, ...

로보틱스(Robotics)란?

로봇의 설계 및 응용을 위한 학문

Robotics is an interdisciplinary branch of computer science and engineering.

컴퓨터공학
동작 구현 등



기계공학
외관 디자인 등

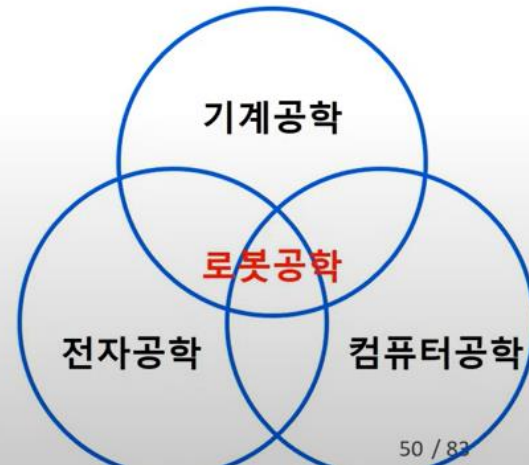
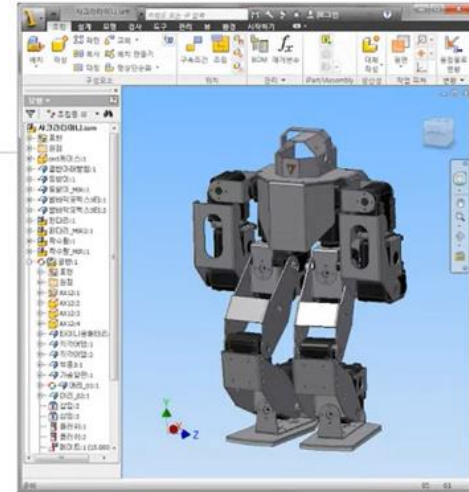
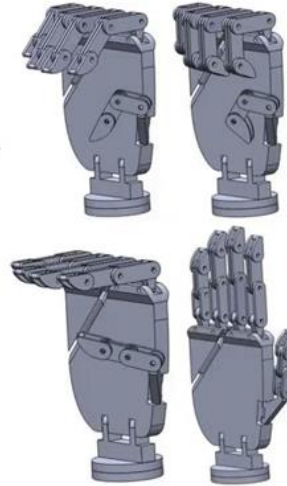


전자공학
모터 제어, 센싱 등

로보틱스(Robotics)란?

로봇 개발 단계

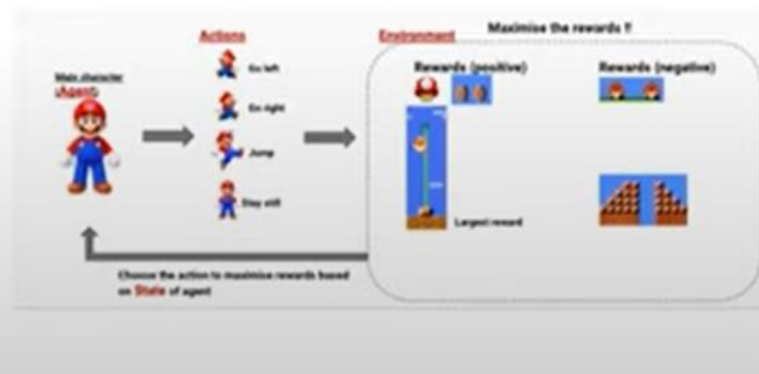
- 1) 기구 설계
 - 로봇 하드웨어 디자인
- 2) 시뮬레이션
 - 로봇의 행동 생성
 - 개발기술 확인
- 3) 전자부 설계
 - 모터, 센서 제어를 위한 회로 디자인
- 4) 시제품 제작
 - 프로토타입 만들기
- 5) 상품화 개발
 - 대량생산을 통해 원가 절감, 디자인 수정



로보틱스(Robotics)란?

AI x Robotics

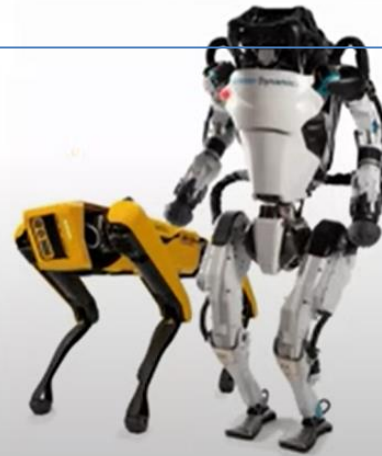
- 1) 로봇 제어(강화학습)
 - 움직임 생성, 작업 수행
 - 다관절 로봇의 행동 생성
 - 로봇손 컨트롤
- 2) 환경 인식(센서처리)
 - 이미지, 라이다, IMU, 초음파 센서 처리
- 3) SLAM
 - 위치추정 및 주변 환경 지도 작성
- 4) 자율주행
 - 경로 생성, 충돌 회피



로보틱스(Robotics)란?

로봇 구성 요소

- 1) 바디부(Manipulator)
 - 로봇 팔, 다리
- 2) 말단부(End-effector)
 - 손, 발
- 3) 구동기(Actuator)
 - 관절
- 4) 제어기(Controller)
 - 근육
- 4) 센서(Sensor)
 - 눈, 코, 입, 귀, 촉감
- 6) 소프트웨어(SW)
 - 머리
- 7) 동력장치(Power)
 - 소화기관



서보모터: 각도 제어
DC모터: 방향, 속도 제어

센서 값이 계속 들어오는 중에 데이터 분석, 추론 등이 동시에, 실시간으로 일어남



ROS



다양한 HW



Personal Computer

Galaxy S5 Tear Down



SAMSUNG TOMORROW

Personal Phone

다양한 OS



Personal Computer



Personal Phone

로봇 HW



TurtleBots
Burger



ROBOTIS OP
ROBOTIS OP

THORMANG3

로봇 OS



ROS 등장배경

■ 기존 로봇 개발 방식의 한계

로봇 개발 과정

- 하드웨어

하드웨어 설계 조립

하드웨어 제어기 설계

제어, 센싱 펌웨어 작성

센서, 모터와 PC 간 통신 구현

- 소프트웨어

통신 코드 작성

인식, 네비게이션 코드 작성

Task 실행 프로그램 작성

프로그램 검증, 디버깅

등등.....



ROS 등장배경

- 기존 로봇 개발 방식의 한계



API 마다 Interface가 다르고 적용하는데 학습이 필요함

하드웨어 의존적인 소프트웨어

- 로봇이 달라지면 소프트웨어를 재작성 하거나 많은 부분을 수정해야 함
- OS에 의존적 이어서 OS 변경이 어렵거나 불가능함
- 멀티 PC, 로봇을 구성하는 경우 통신 구축, 검증에 많은 시간을 소비함

ROS 등장배경

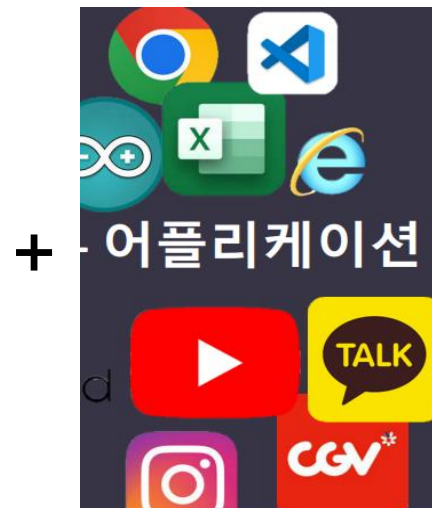
- 로봇 소프트웨어 플랫폼의 필요성
기존 SW의 생태계를 생각해보면..



하드웨어의 모듈화



운영체제가 하드웨어를
제어하여 앱 개발이 쉬워짐



개발자가 늘어나
다양한 앱 출시



유저의 증가로
피드백 생성

ROS 등장배경

- 로봇 소프트웨어 플랫폼의 필요성
로봇의 경우로 확장시켜 플랫폼을 만든다면?



가격 Down, 성능 UP

사용자 수요에 맞는 서비스
->선택의 자유도가 높아짐
사용자경험 질이 높아짐

하드웨어 추상화-규격화-모듈화

하드웨어 인터페이스의 통합

개발자를 하드웨어, 미들웨어(OS),
소프트웨어로 분리하여 전문성 향상

ROS란?

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

출처: <http://wiki.ros.org/Documentation>

- 운영체제와 비슷한 하드웨어 추상화를 포함하면서도 전통적 운영체제는 아님
- 독립된 운영체제는 아니며 우분투 등 기존 운영체제를 이용함

운영체제 역할

프로세스 관리 시스템, 파일 시스템, 유저 인터페이스, 프로그램 유틸리티
(컴파일러, 스레드 모델 등) ...

- 로봇 응용 프로그램에 필요한 필수 기능들을 라이브러리 형태로 제공
- 로봇 응용 프로그램을 개발하기 위해 로봇에 특화된 개발환경을 제공하는 로봇 SW 플랫폼이자 로봇 SW개발을 위한 툴

ROS(Robot Operating System)

ROS는 로봇을 위한 오픈 메타 운영 체제이다.

- 시스템 개발을 위한 오픈 소스 프레임워크이다.
- 메타 운영 체제란, 일반적인 운영 체제처럼 작동하지만, 실제로는 운영 체제 위에서 동작하는 소프트웨어 플랫폼을 의미한다.
- ROS는 로봇을 제어하고 개발할 수 있는 소프트웨어환경을 제공하며, 로봇 개발자들이 로봇의 다양한 기능을 쉽게 구현할 수 있도록 돕는다.
- ROS 2는 ROS(로봇 운영 체제)의 다음 세대 버전이다.
- ROS 2는 로봇 소프트웨어 개발자들에게 더욱 안전하고 확장 가능하며 분산된 시스템을 구축할 수 있는 도구와 라이브러리를 제공한다.

ROS의 목적 및 특징

- Open Source Robotics Foundation (ROS 개발 재단) CEO가 말하는 ROS의 목적

ROS의 목적은 기존의 PC와 Phone의 역사를 Robot에 반영시키기 위한 생태계 형성이다.



- 로보틱스 소프트웨어 개발을 전 세계 레벨에서 공동 작업이 가능하도록 환경 구축
- 로보틱스 연구, 개발에서의 코드 재사용을 극대화
 - > 노드 단위의 분산 프로세스 지원
 - > 공유 및 재배포를 쉽게하는 패키지 단위 관리
 - > 다양한 프로그래밍 언어 지원

ROS 1 vs ROS 2

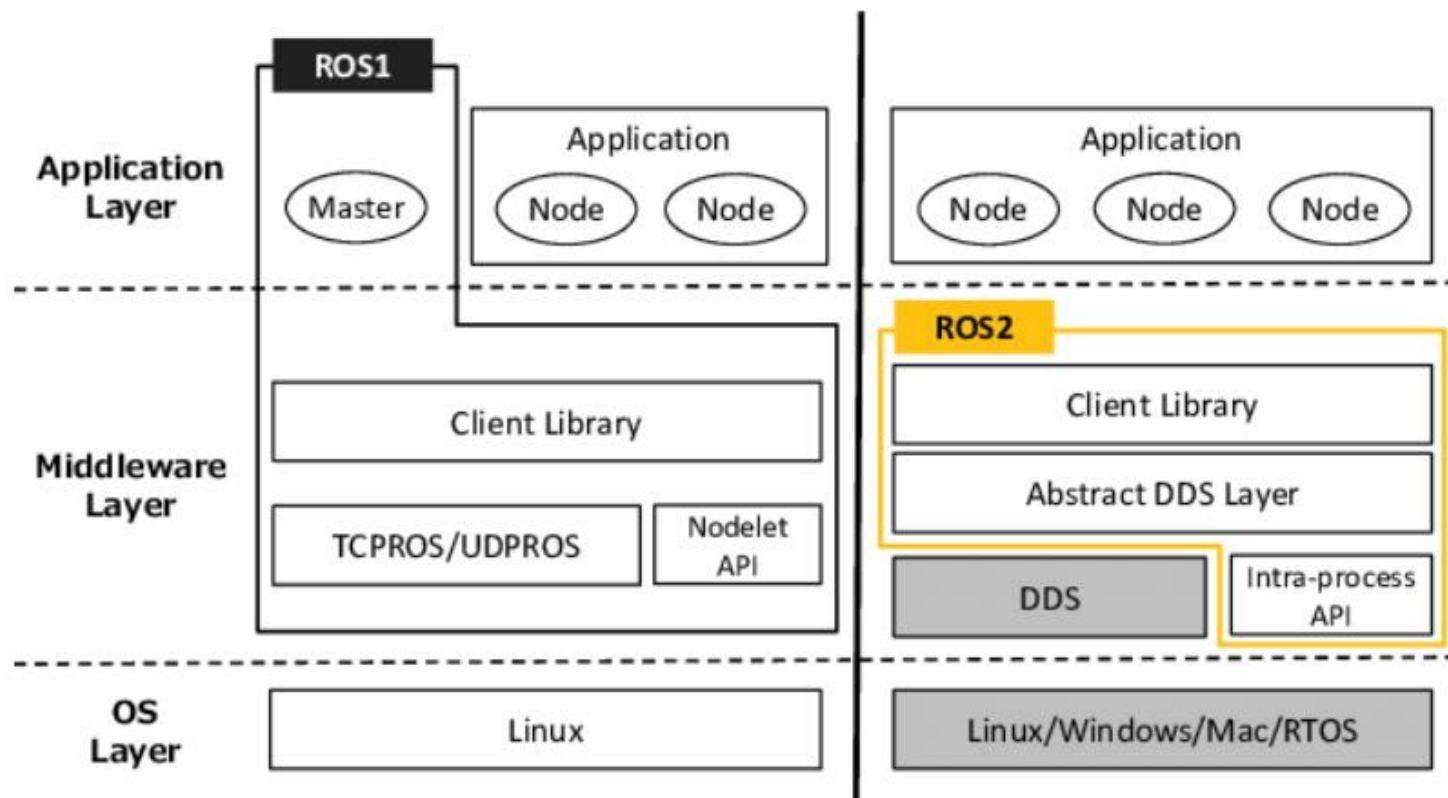
- ROS 1은 2007년을 개발을 시작으로 **개인 서비스 로봇**인 PR2 개발에 필요한 미들웨어 형태의 로봇 개발 프레임워크를 다양한 개발 툴과 함께 오픈 소스로 공개하며 시작되어 초기 컨셉트가 그대로 적용됨

ROS 1	WE WANT MORE!
단일 로봇	여러 대의 로봇
워크스테이션급 컴퓨터	임베디드 시스템에서의 ROS사용
실시간 제어 불가	실시간 제어
안정된 네트워크 환경이 요구됨	불안정한 네트워크 상황에서도 동작할 수 있는 유연함
대학이나 연구소 같은 아카데미 연구용	상업 제품 지원
리눅스 환경	멀티 플랫폼(리눅스, 윈도우, Mac OS)
	최신 기술 지원

참고: <http://design.ros2.org/articles/changes.html>
<https://www.theconstructsim.com/infographic-ros-1-vs-ros-2-one-better-2/>

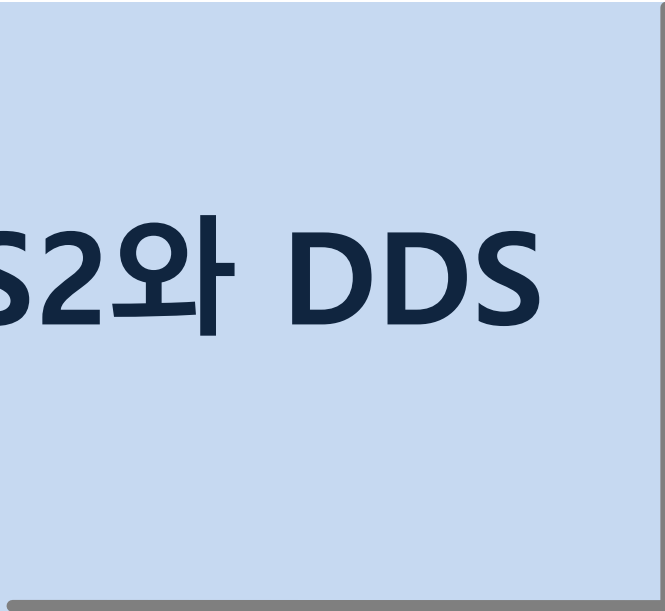
ROS 1 vs ROS 2

- DDS(Data Distribution Service): 데이터 통신을 위한 MW



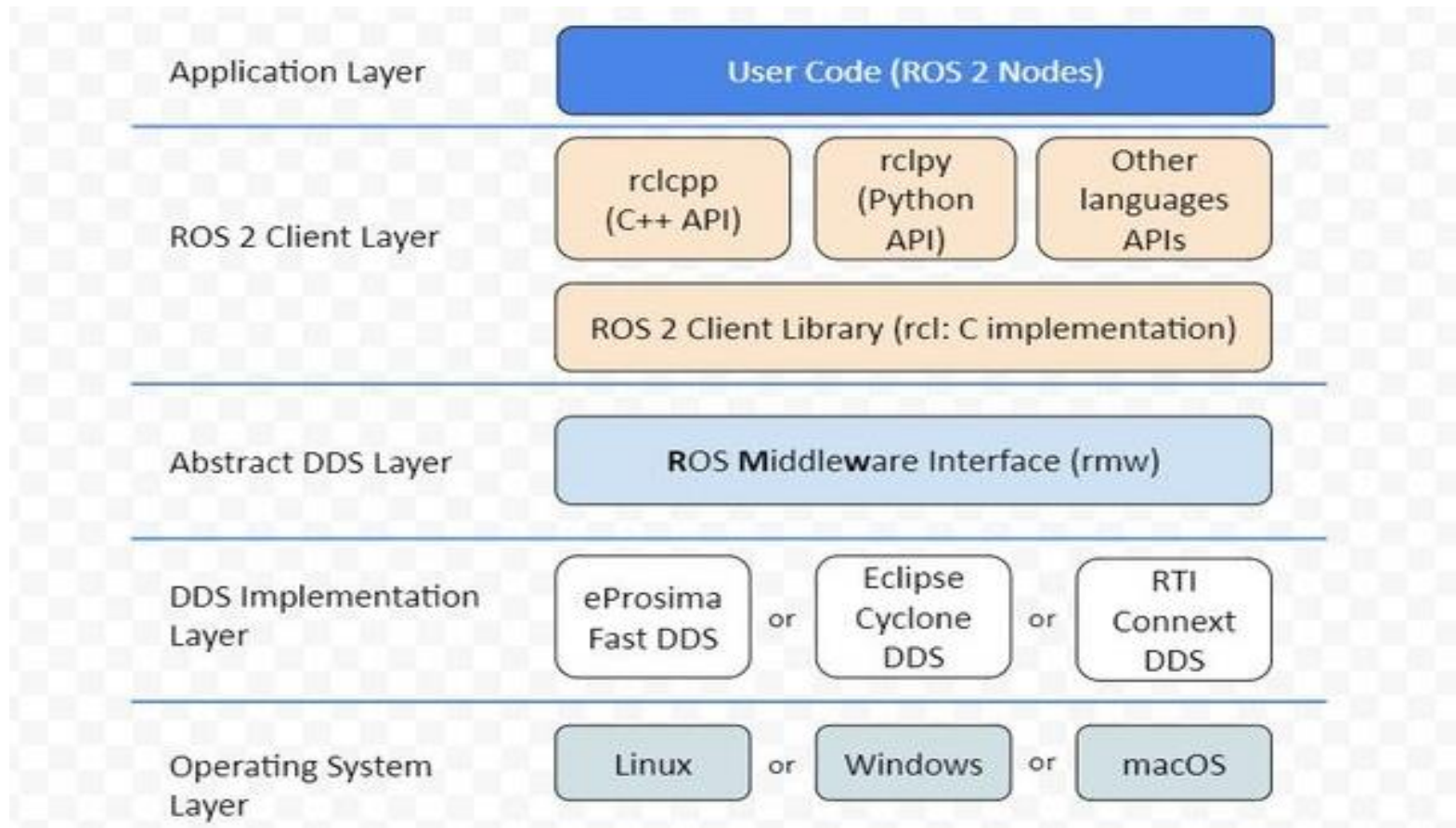


ROS2와 DDS

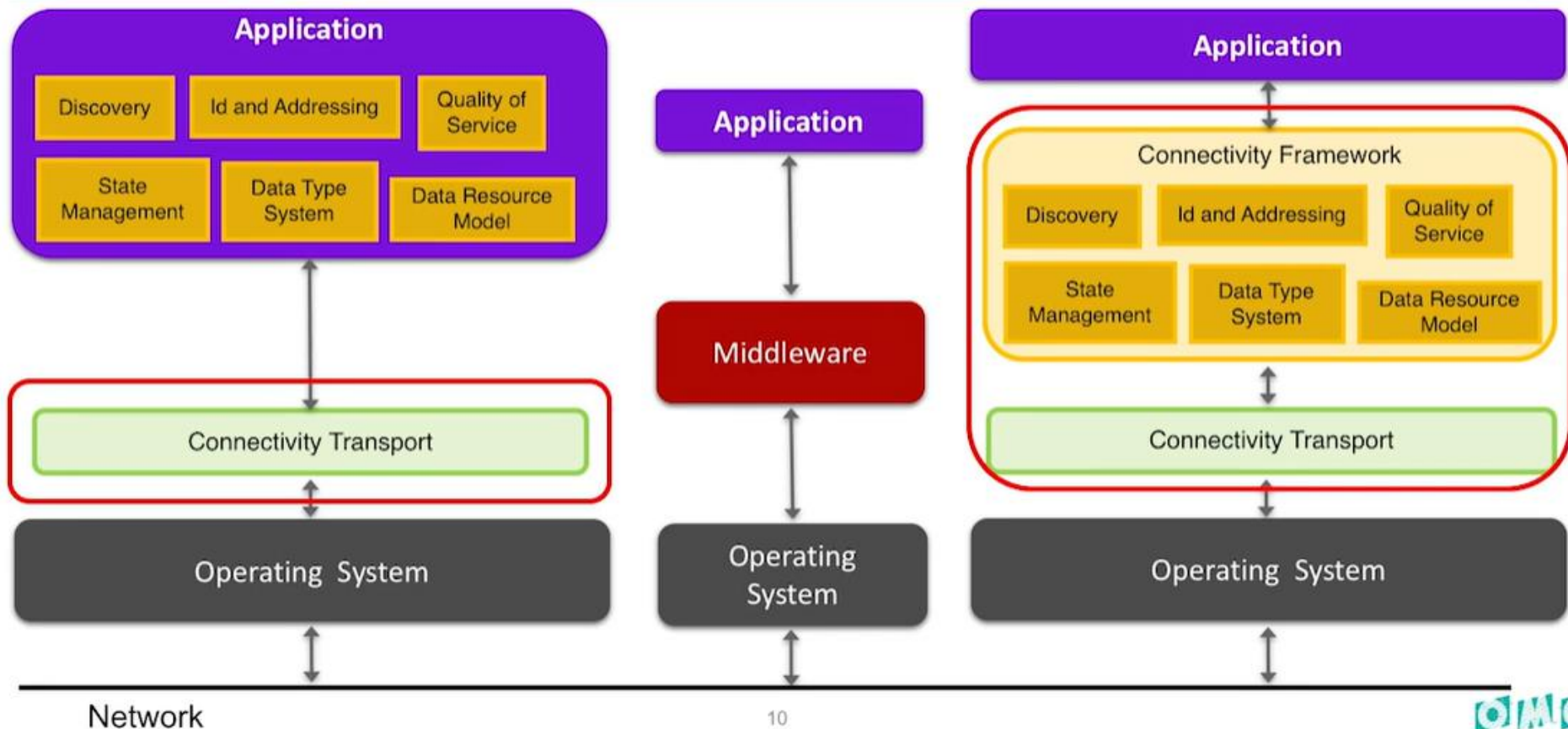


ROS2(Robot Operating System) Architecture

- ROS는 전 세계 레벨에서 공동 작업이 가능하도록 코드 재사용을 극대화 하는 것을 목적으로 함

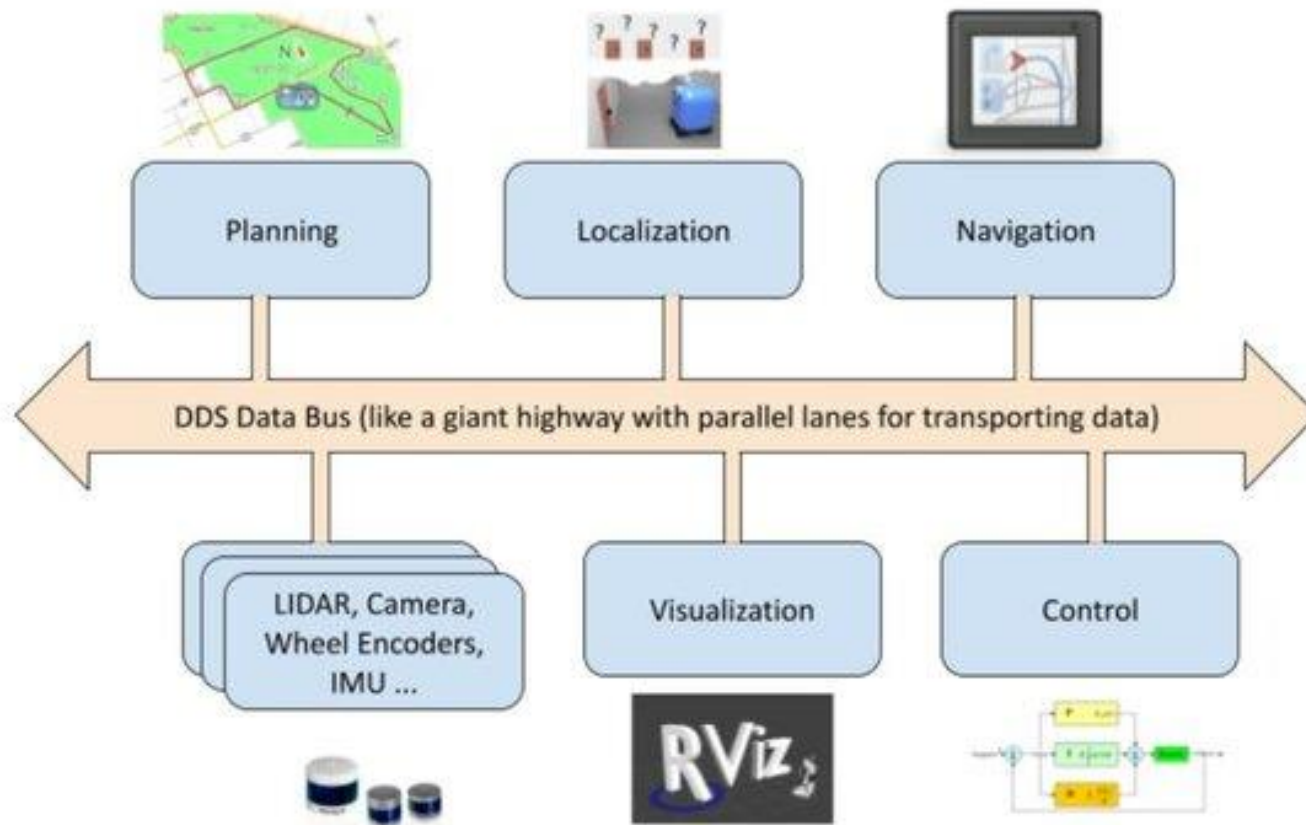


Complexity of the Application Code



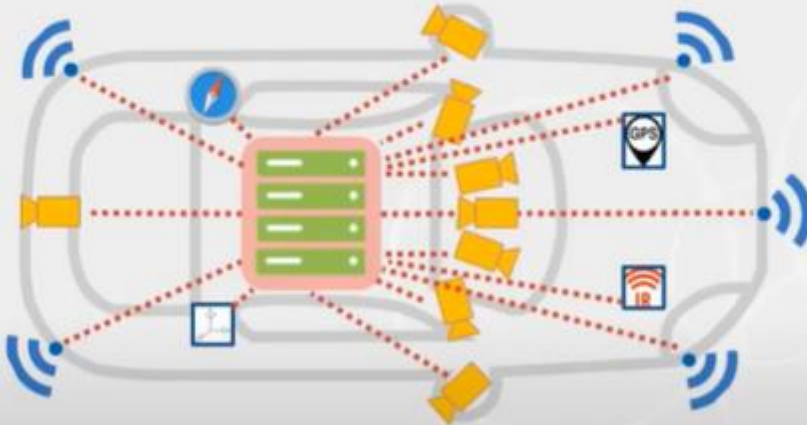
DDS(Data Distribution Service)

- OMG에서 정의한 발간-구독(Publish-Subscribe) 방식의 실시간 데이터 분배 서비스 규격



로보틱스 - DDS - ROS 2

Modern Autonomous vehicles



The diagram illustrates a top-down view of a vehicle chassis. In the center is a pink rectangular box representing the main computer system. Radiating from this central box are numerous red dotted lines, each connecting to a different sensor or component. These components are represented by various icons: yellow camera icons, blue LiDAR icons, blue radar icons, blue compass icons, blue accelerometer icons, blue gyroscope icons, blue environmental sensor icons, blue ultrasonic range-finder icons, blue road ice sensor icons, blue wheel sensors, blue GPS/GNSS location icons, and blue driver control/monitoring icons. The vehicle is also shown with blue Wi-Fi signal icons at the front and rear, indicating connectivity.

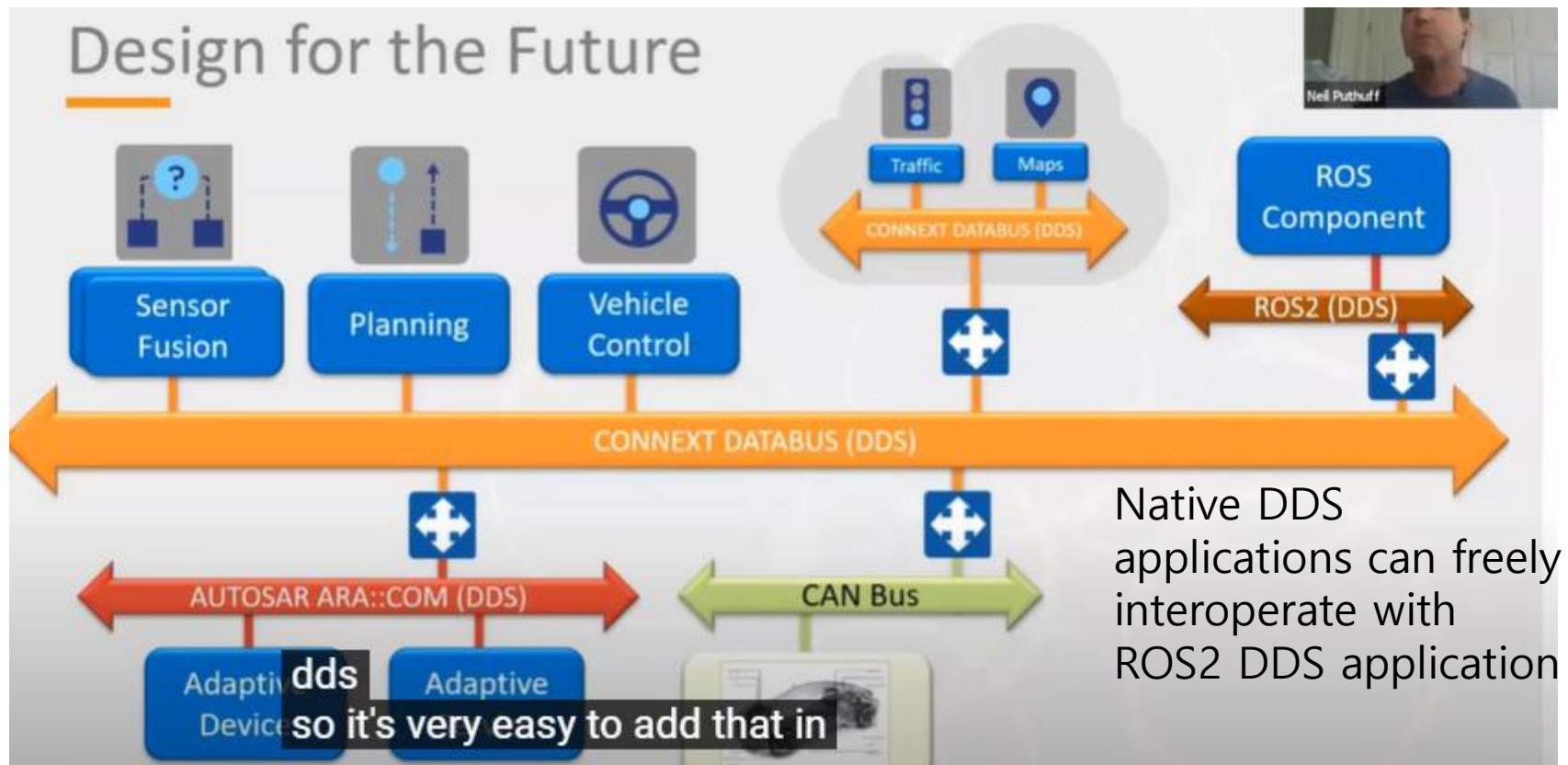
- Cameras
- LiDAR
- RADAR
- Compass
- Accelerometer
- Gyroscope
- Environmental
- Ultrasonic Range
- Road Ice
- Wheel Sensors
- Location (GPS, GNSS)
- Driver Controls
- Driver Monitoring

and since people's lives depend on it it needs to

자율주행 자동차처럼 자동화되는 영역이 늘어남

자동화 레벨이 높아질수록 communication에 필요한 데이터와 처리량이 증가

로보틱스 - DDS - ROS 2



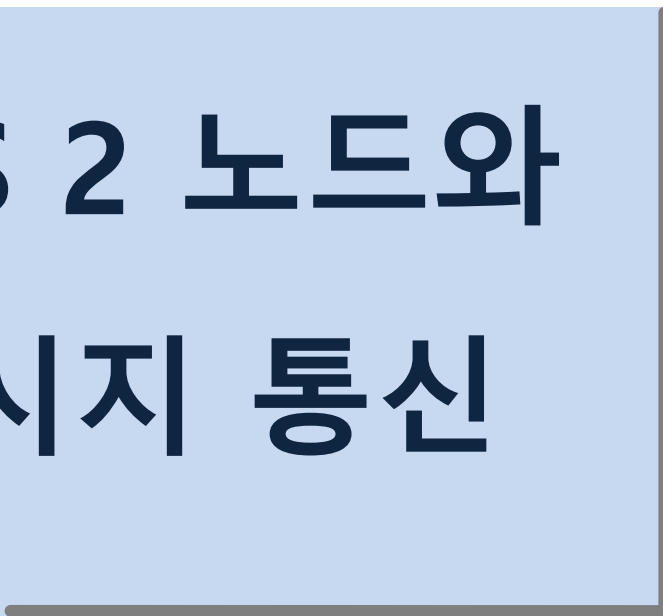
DDS를 통한 DATABUS 형태를 구성한다는 것은 다양한 센서, 기능을 언제든지 연결할 수 있는 유연함(flexibility)을 의미함

DDS 도입을 통한 ROS 2 특징

- 산업 표준으로 자리잡음
- 리눅스, 윈도우, Mac OS, 안드로이드, VxWorks등 다양한 OS 지원
- 사용자 코드 레벨과 분리되어 다양한 프로그래밍 언어 사용가능
- 신뢰성 있는 멀티캐스트를 구현함
- 데이터 중심으로 적절한 수신자에게 적절한 정보를 효율적으로 전달함
- 노드 정보를 별도로 등록하지 않아도 되는 동적검색 기능
- 인프라, 항공, 우주 산업같은 초대형 시스템으로 확장 가능한 설계
- 다양한 벤더로 변경 가능
- 데이터 송수신 설정을 유저가 직접 할 수 있는 QoS(Quality of Service)기능
- 보안 이슈 해결



ROS 2 노드와 메시지 통신



실습

```
$ ros2 run <package_name> <executable_name>
```

터미널을 2개 열고 아래 코드를 각각 실행시켜 보자

과제 1

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim turtle_teleop_key
```

Turtlesim 패키지와 노드

- 두 노드 사이 '키보드 값'이 전달됨
 - 구체적으로 병진 속도(Linear velocity)와 회전 속도(Angular velocity)를 포함
- 이처럼 다양한 센서 데이터, 제어 데이터를 주고받아야 할 경우 '메시지(message)' 형태로 제공함.
- 이 예시에서는 어떤 message type을 사용했는지 알아보시다

과제2

```
$ ros2 node info turtlesim_node #특정 노드 정보 보기
```

```
$ rqt_graph #노드와 토픽의 그래프 뷰
```

```
$ ros2 node list #현재 실행 노드 보기
```

```
$ ros2 topic list #현재 실행 topic 보기
```

```
$ ros2 service list #현재 실행 service 보기
```

```
$ ros2 action list #현재 실행 action 보기
```


도메인(Domain)

- 가상의 데이터 공간을 구분하는 논리적 네트워크 공간

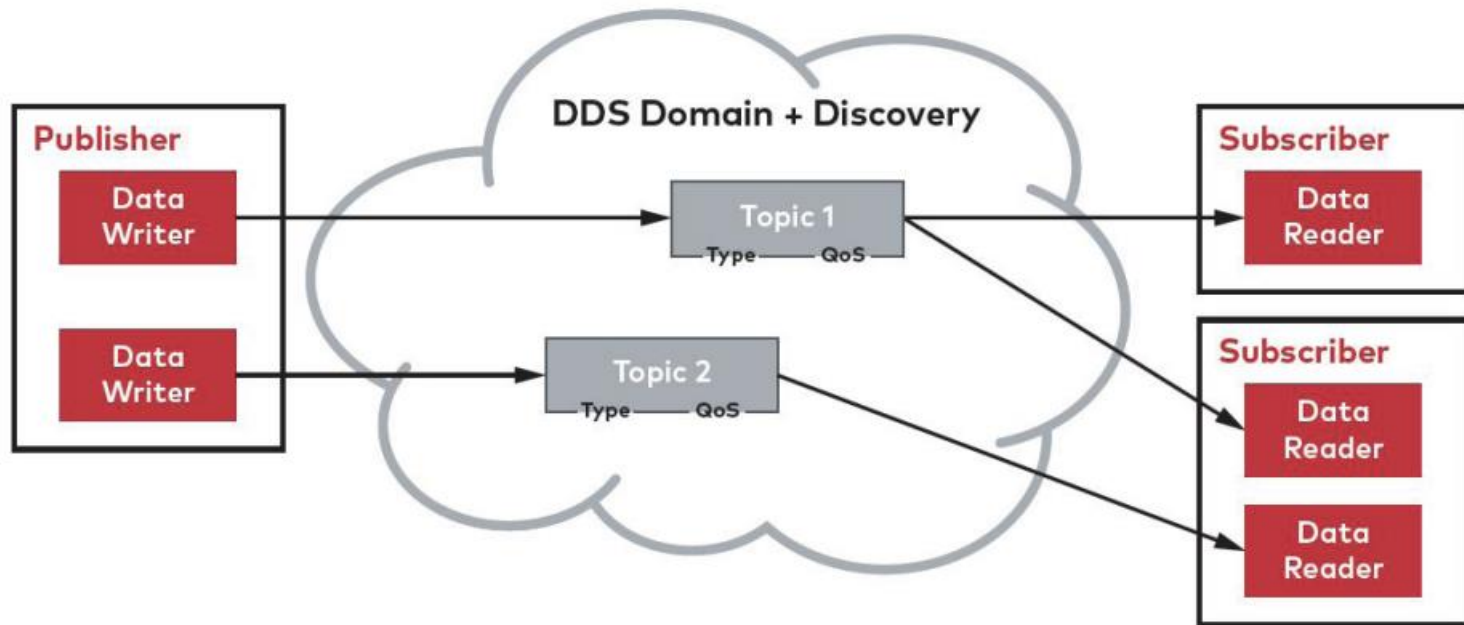


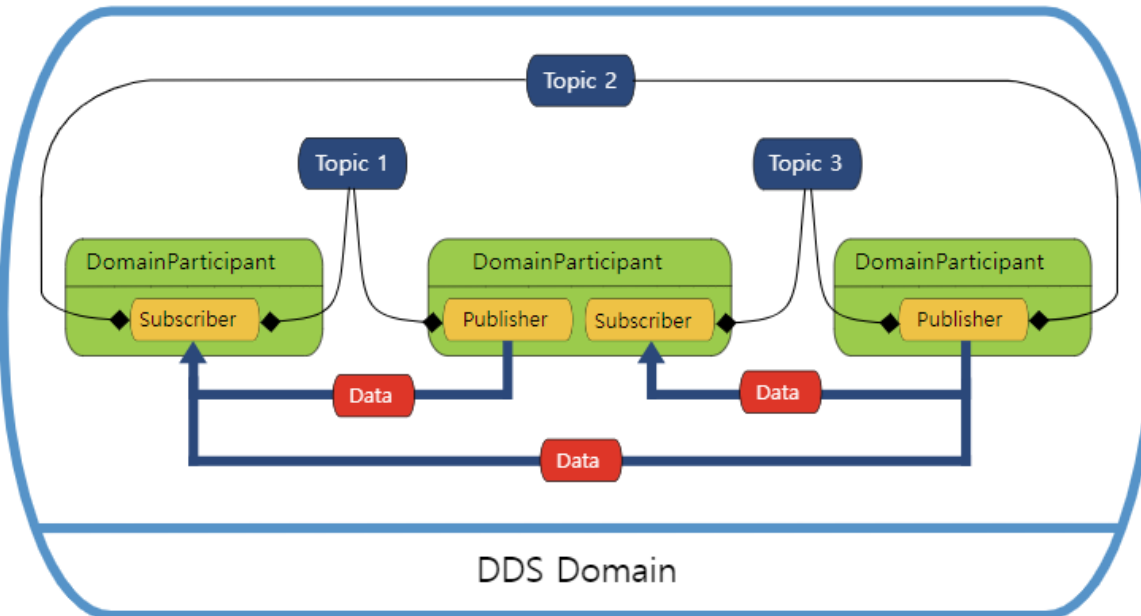
그림 출처: https://cdn.vector.com/cms/content/know-how/_technical-articles/PREEvision/PREEvision_MiddlewareProtocols_ElektronikAutomotive_202003_PressArticle_KO.pdf

도메인(Domain) 변경하기

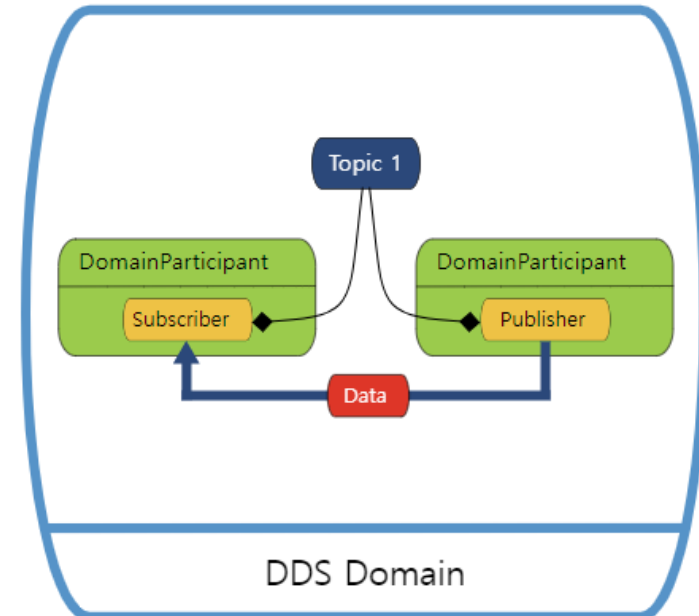
```
$ export ROS_DOMAIN_ID=<number>
```

```
$ export ROS_DOMAIN_ID=30
```

ROS_DOMAIN_ID=1



ROS_DOMAIN_ID=2



특정 도메인 그룹(ROS_DOMAIN_ID가 같은 숫자)에 대해서만 데이터를 전송함

노드와 메시지 통신

- 노드와 메시지
 - 노드(Node): 최소 단위 실행 가능한 프로세스
 - 메시지(Message): Node와 Node 사이에 서로 주고 받는 데이터
- ROS 2 메시지 통신
 - 토픽(Topic): 비동기식 단방향 메시지 송수신 방식
 - 서비스(Service): 동기식 양방향 메시지 송수신 방식
 - 액션(Action): 비동기식+동기식 양방향 메시지 송수신 방식
 - 파라미터(Parameter): 노드의 매개변수를 변경하거나 가져오는 송수신 방식



[teleop_turtle]

Publisher
메시지 발행



[/turtle1/cmd_vel]

geometry_msgs/msg/Twist
메시지



[turtlesim]

Subscriber
메시지 구독

[geometry_msgs/msg/Twist]

Vector3 linear
Vector3 angular

[geometry_msgs/msg/Vector3]

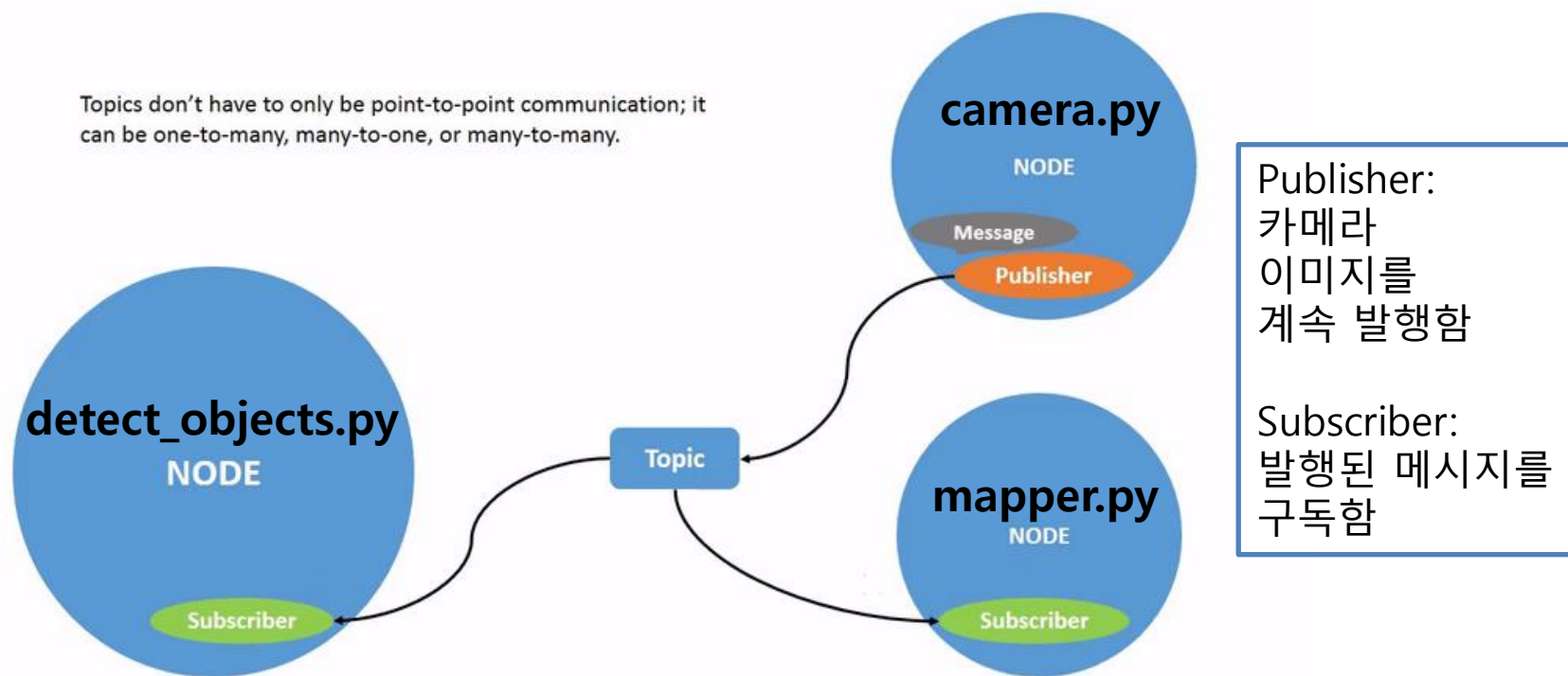
float64 x
float64 y
float64 z

[geometry_msgs/msg/Vector3]

float64 x
float64 y
float64 z

Topic

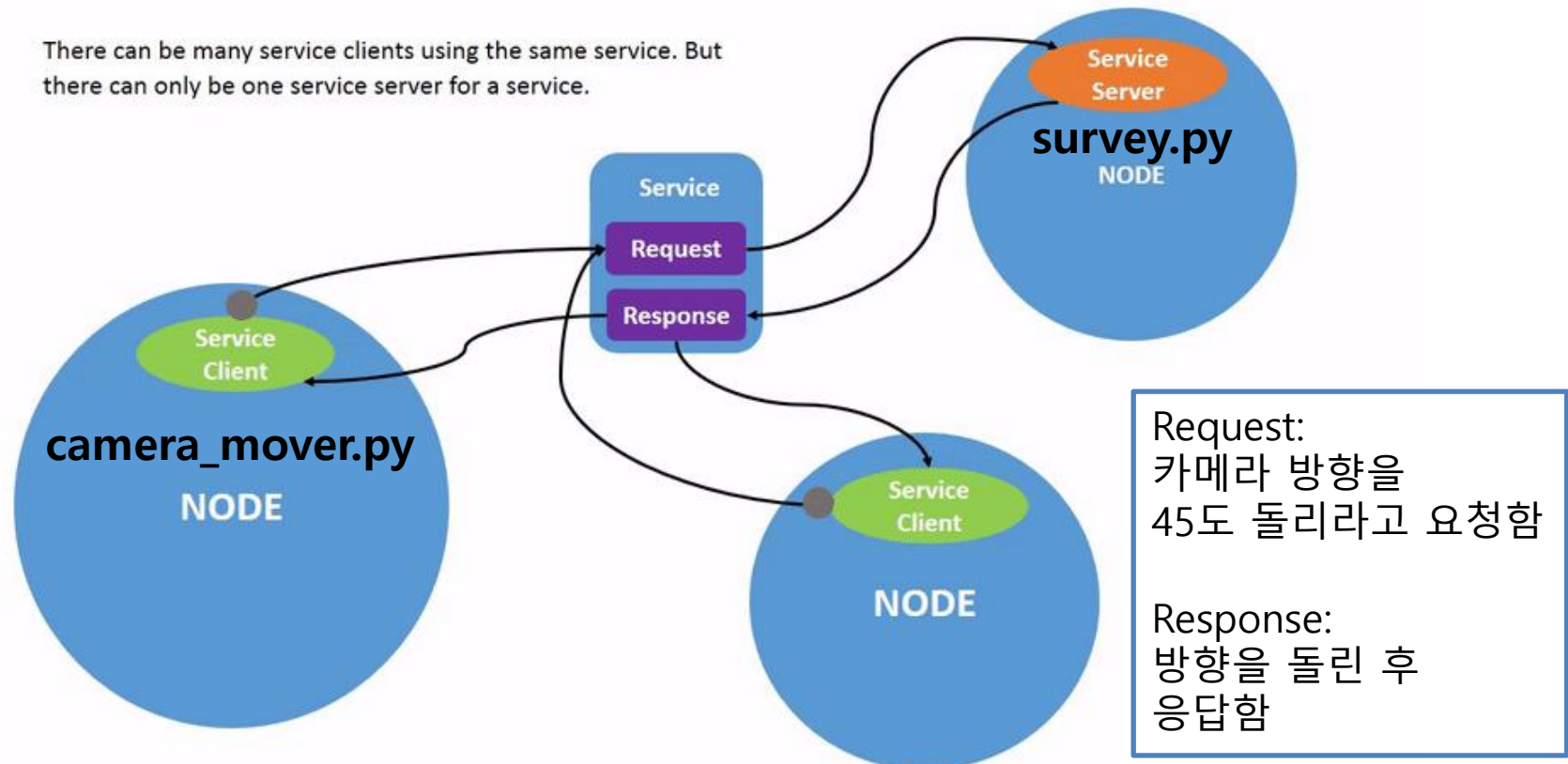
- 비동기식 단방향 메시지 송수신 방식(1:1, 1:N, N:1, N:N)
- 메시지를 발간하는 publish와 메시지를 구독하는 subscriber로 구성
- 토픽 데이터(data) : msg 메시지



<https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html>

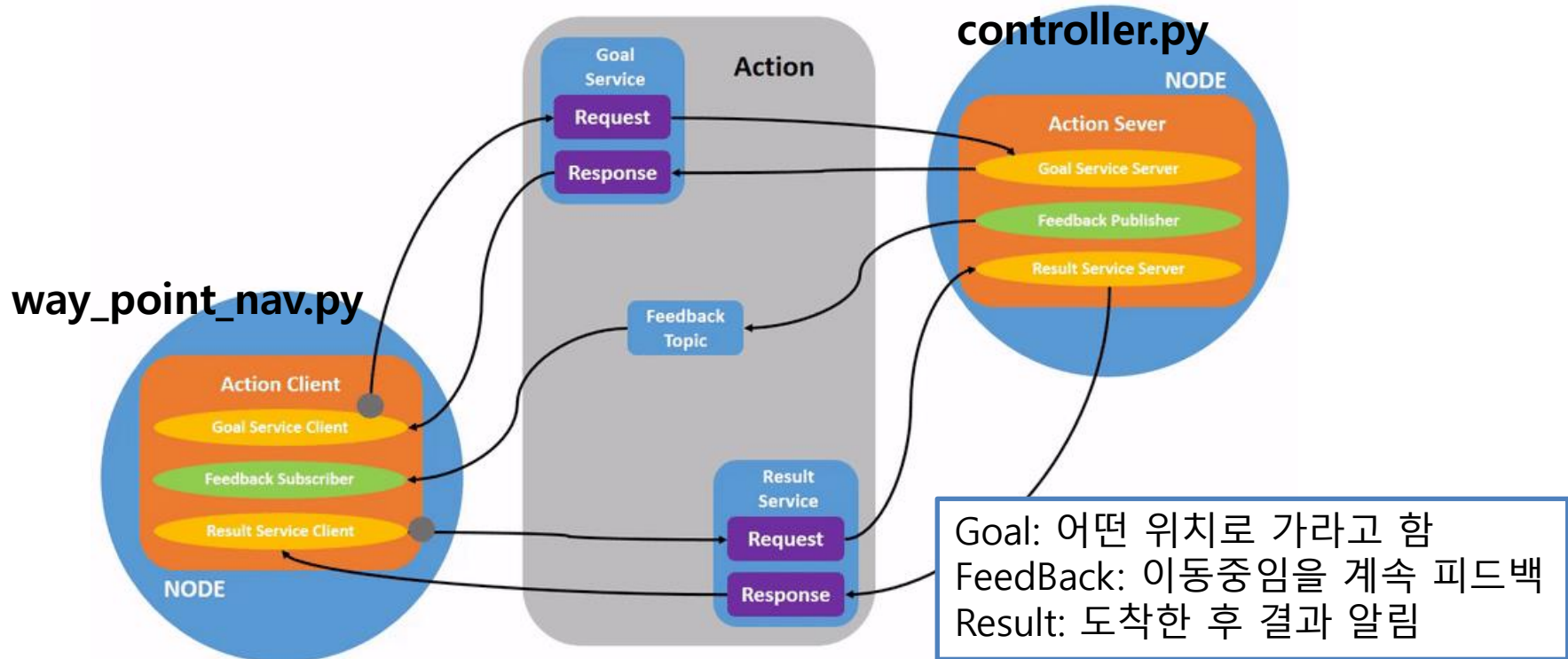
Service

- 동기식 양방향 메시지 송수신 방식 (1:1, 1:N)
- 특정 요청을 하는 client와 요청을 받는 server의 통신
- 서비스 요청(request), 서비스 응답(response) : srv 메시지



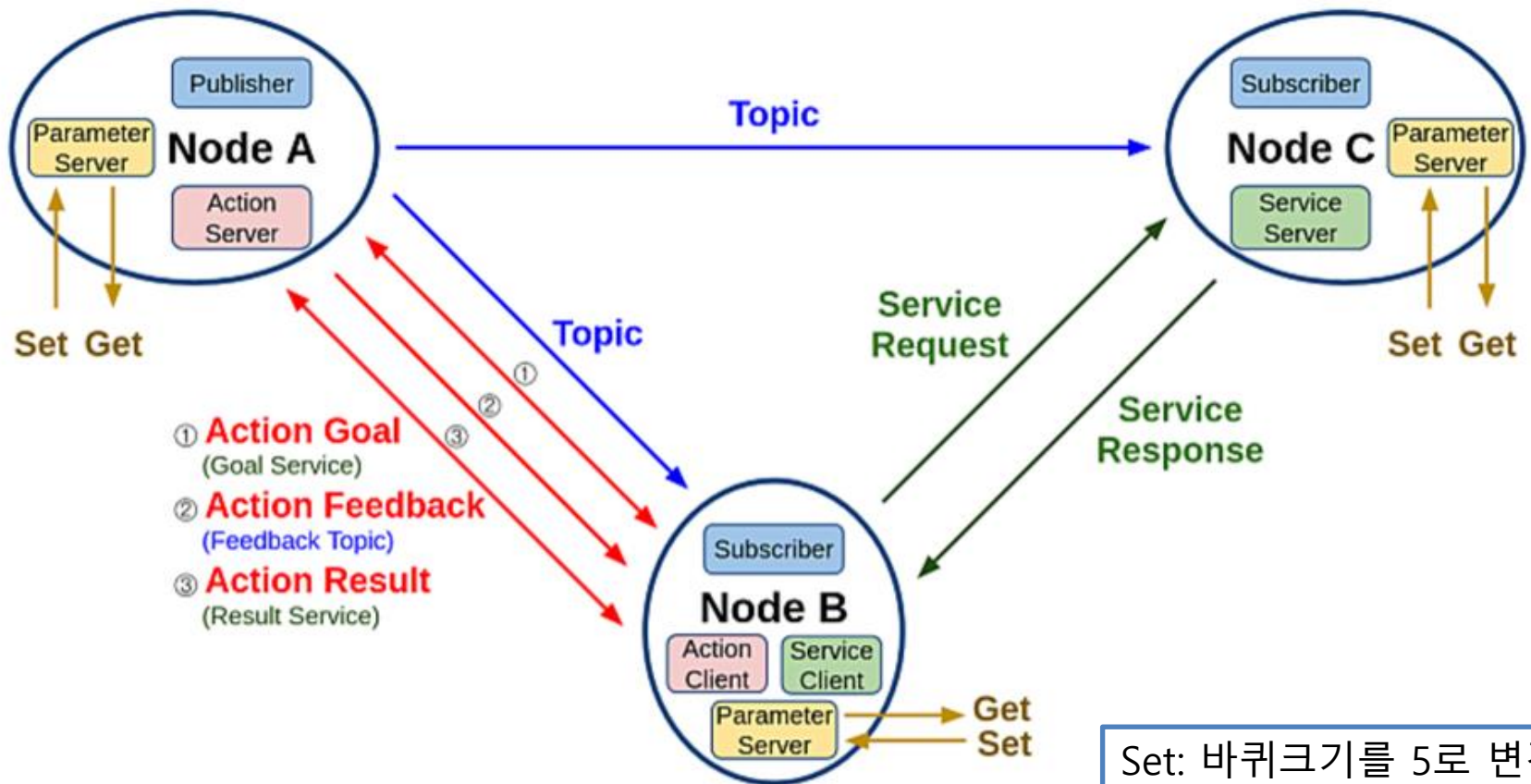
Action

- 비동기식+동기식 양방향 메시지 송수신 방식(1:1)
- 액션 목표 Goal Action client가 지정 후, 목표를 받아 특정 태스크를 수행하면서 중간 값인 Action feedback과 최종 결과값에 해당하는 Action result를 server가 전송
- 액션 목표(goal), 액션 결과(result), 액션 피드백(feedback) : action 메시지



Parameter

- 노드 내 매개변수를 쉽게 설정(Set)하거나 가져옴(Get)



Set: 바퀴크기를 5로 변경
Get: 바퀴크기 가져오기

turtlesim 실습

- ros_tutorial 하위에 있는 ROS 학습용 패키지

turtlesim 시뮬레이션 창

```
$ ros2 run turtlesim turtlesim_node
```

turtlesim 제어 - 터미널 창 선택 후, 키보드 화살표 키 사용

```
$ ros2 run turtlesim turtle_teleop_key
```

과제3

노드, 토픽, 서비스, 액션 조회

```
$ ros2 node list  
$ ros2 topic list  
$ ros2 service list  
$ ros2 action list  
$ ros2 param list
```

참고:

Turtlesim: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html>

Node: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>

Topic: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html>

Service: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html>

Action: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html>

Parameter: <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Parameters/Understanding-ROS2-Parameters.html>