ORC-PG-ROS2-WEEK3

Chapter 9 - 12

[CH9] 패키지 설치와 노드 실행

Turtlesim 패키지 설치

```
$ sudo apt update
```

```
$ sudo apt install ros-foxy-turtlesim
```

ROS2 Humble을 설치했으므로 아래의 명령어를 통해 Turtlesim 구현

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run thrtlesim turtle_teleop_key
```

```
$ ros2 topic echo /turtle1/cmd_vel
```

```
$ rqt
```

Turtlesim 이란?

- turtlesim 패키지는 ROS 1 초창기부터 ROS 2까지 ROS 커뮤니티에서 가장 널리 쓰이는 패키지
- ROS를 처음 접하는 유저들에게 튜토리얼로 제공하기 위해 제작
- 본래 터틀(Turtle)은 1967년에 개발된 교육용 컴퓨터 프로그래밍 언어인 로고(Logo)를 사용하여 실제 로봇을 구동하기 위해 만들어진 터틀 로봇에서 유래
- 실제 로봇이 없더라도 컴퓨터 화면상에서 쉽게 ROS의 기본 개념을 설명하고자 turtlesim이라는 프로그램 제작.

Turtlesim 패키지와 노드

- ROS에서는 프로그램 재사용성을 극대화하기 위해 최소 단위의 실행 가능한 프로세스라고 정의하는 노드(Node) 단위로 프로그램 작성
- 하나 이상의 노드 또는 노드 실행을 위한 정보 등을 묶어 놓은 것을 패키지라고 하며, 패키지의 묶음을 메타패키지(Metapackage)라 부름

패키지 확인하기

```
$ ros2 pkg list
```

패키지에 포함된 노드 확인하기

```
$ ros2 pkg executables turtlesim
```

```
turtlesim draw_square
```

```
turtlesim mimic
turtlesim turtle_teleop_key
```

```
turtlesim turtlesim_node
```

- draw_square : 사각형 모양으로 turtle을 움직이게 하는 노드

- mimic : 유저가 지정한 토픽으로 동일 움직임의 turtlesim_node를 복수 개 실행시킬 수 있는 노드
- turtle_teleop_key : turtlesim_node를 움직이게 하는 속도 값을 퍼블리시 하는 노드
- turtlesim_node : turtle_teleop_key로부터 속도 값을 토픽으로 받아 움직이게 하는 간단한 2D 시뮬레이터 노드

Turtlesim 패키지의 노드 실행하기

`$ ros2 run turtlesim turtlesim_node` 거북이 한 마리 등장

`$ ros2 run turtlesim turtle_teleop_key` 거북이 조정 가능 (화살표키)

노드, 토픽, 서비스, 액션의 조회

`$ ros2 run turtlesim turtle_teleop_key` 를 통해 움직이는 것에 중요한 점은 단순히 키보드 값을 전달하여 거북이를 움직이는 게 아니라 눌려진 키보드 키 값에 해당되는 병진 속도(Linear velocity)와 회전 속도(Angular velocity)를 geometry_msgs 패키지의 Twist 메시지 형태로 통신한다는 점

현재 어떠한 노드들이 실행되고 있는지, 어떠한 토픽들이 있는지, 어떠한 서비스와 액션이 있는지 알아보기 위해서는 각각의 명령어를 터미널 창에 실행

`$ ros2 node list`

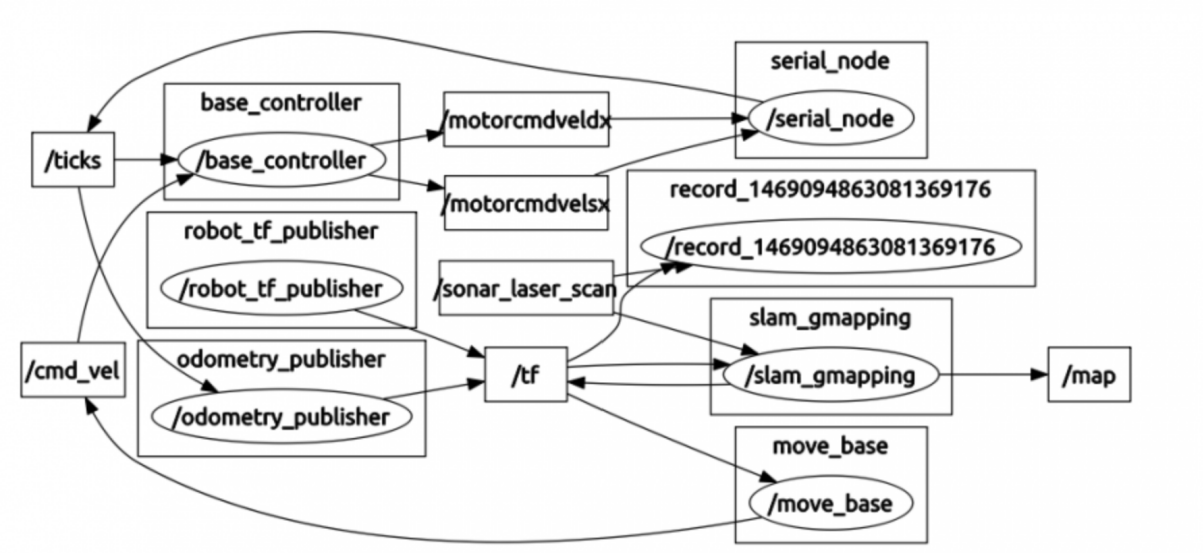
`$ ros2 topic list`

`$ ros2 service list`

`$ ros2 action list`

rqt_graph로 보는 노드와 토픽의 그래프 뷰

`$ rqt_graph`



해당 툴을 통해 현재 개발환경에서의 모든 노드와 토픽, 액션을 그래프로 확인할 수 있음

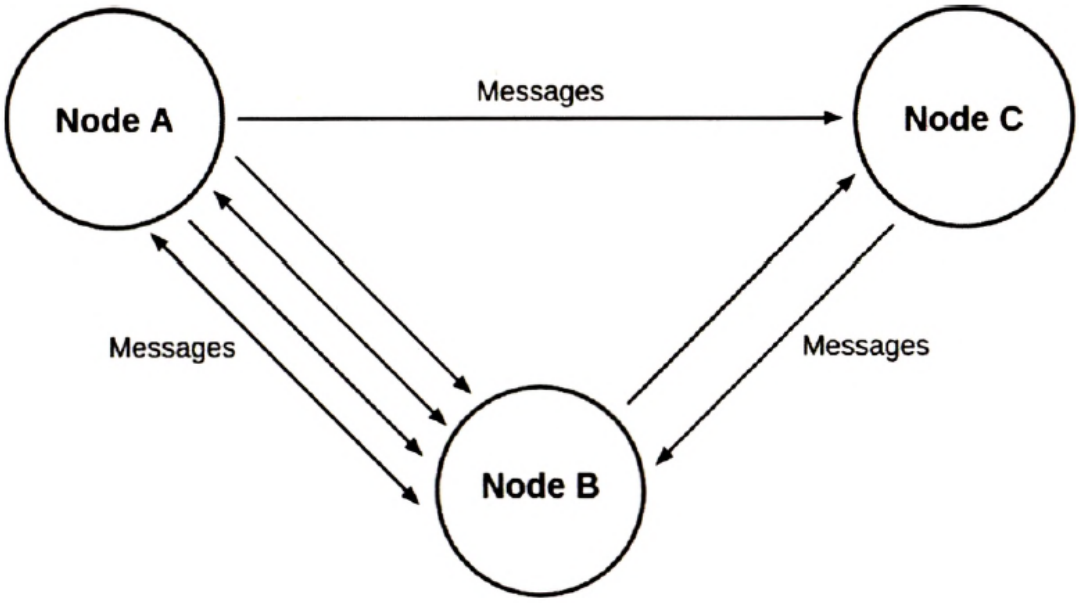
[CH10] ROS 2 노드와 데이터 통신

노드와 메시지 통신

- 노드(Node)는 최소 단위의 실행 가능한 프로세스를 가리키는 용어
- ROS에서는 보통 최소한의 실행 단위로 프로그램을 나누어 작업하게 되고 각 노드의 역할을 목적에 맞추어 세분화시켜 **각 노드들 간의 의존성을 줄이고 독립성을 높여 다른 목적의 작업에서도 일부 노드를 재사용할 수 있도록 한다.**

> 이처럼 수많은 노드들이 연동되는 ROS 시스템을 위해서는 노드와 노드 사이에 입출력 데이터를 서 로 주고받을 수 있도록 설계해야만 한다. 여기서 주고받는 데이터를 ROS에서는 메시지(Message) 라고 하고 주고받는 방식을 메시지 통신(Message Communication)이라고 한다. 여기서 데이터에 해당되는 메시지(Message)는 Integer, Floating point, Boolean, String과 같은 데이터 형태이며 메시지 안에 메시지를 품고 있는 데이터 구조 또는 배열로도 사용할 수 있다. 그리고 메시지를

> 주고 받는 통신 방법에 따라 토픽(Topic), 서비스(Service), 액션(Action), 파라미터(Parameter)로 나눌 수 있다.



노드 실행(ros2 run)

노드를 실행하기 위한 시작은 명령어를 먼저 입력하자!

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 run turtlesim turtle_teleop_key
```

노드 목록(ros2 node list)

현재 개발환경에서 동작 중이 노드 목록을 보기 위해서는 ros2 node list 명령어를 사용하면 된다.

```
$ ros2 node list
```

동일 노드를 복수 개 실행시키기 위해서는 “동일한 노드 이름으로 실행된다”는 점만 기억하자. 노드명 변경을 위해서는 아래 명령어를 실행

```
$ ros2 run turtlesim turtlesim_node __node:=new_turtle
```

노드 정보(ros2 node info)

노드의 정보를 확인하기 위해서는 다음과 같이 ros2 node info 명령어에 정보를 보기 위한 노드명을 지정하면 된다. 이 정보에는 지정된 노드의 Publishers, Subscriber, Service, Action, Parameter 정보를 확인할 수 있다

```
$ ros2 node info /turtlesim
```

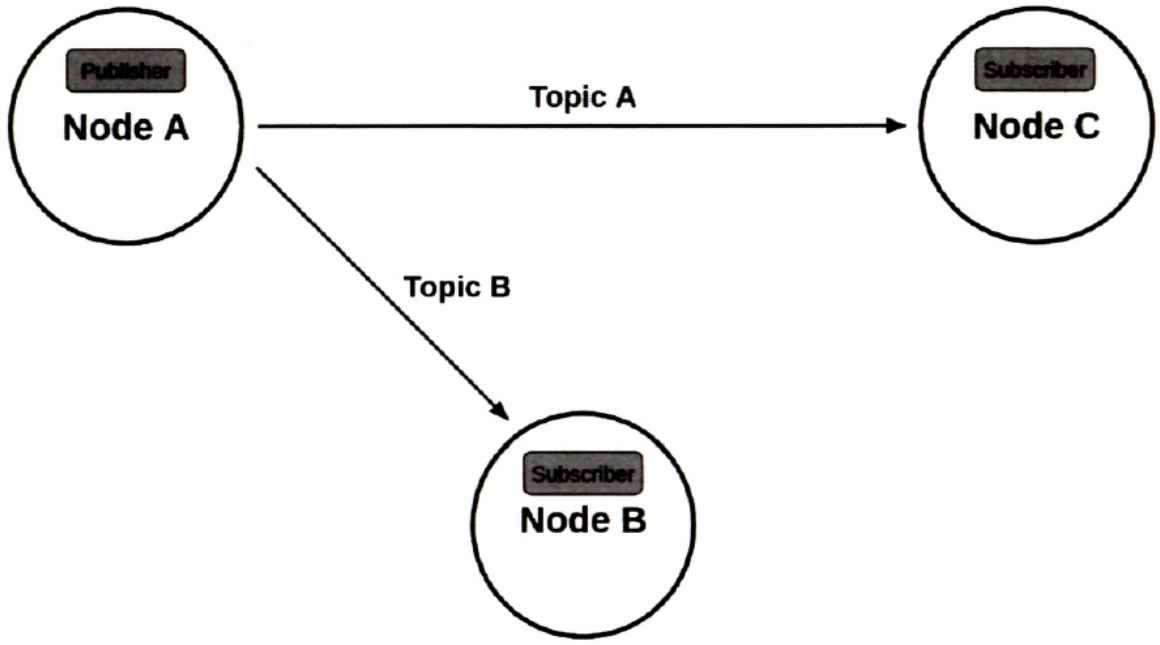
```
$ ros node infor /teleop_turtle
```

[CH11] ROS2 토픽

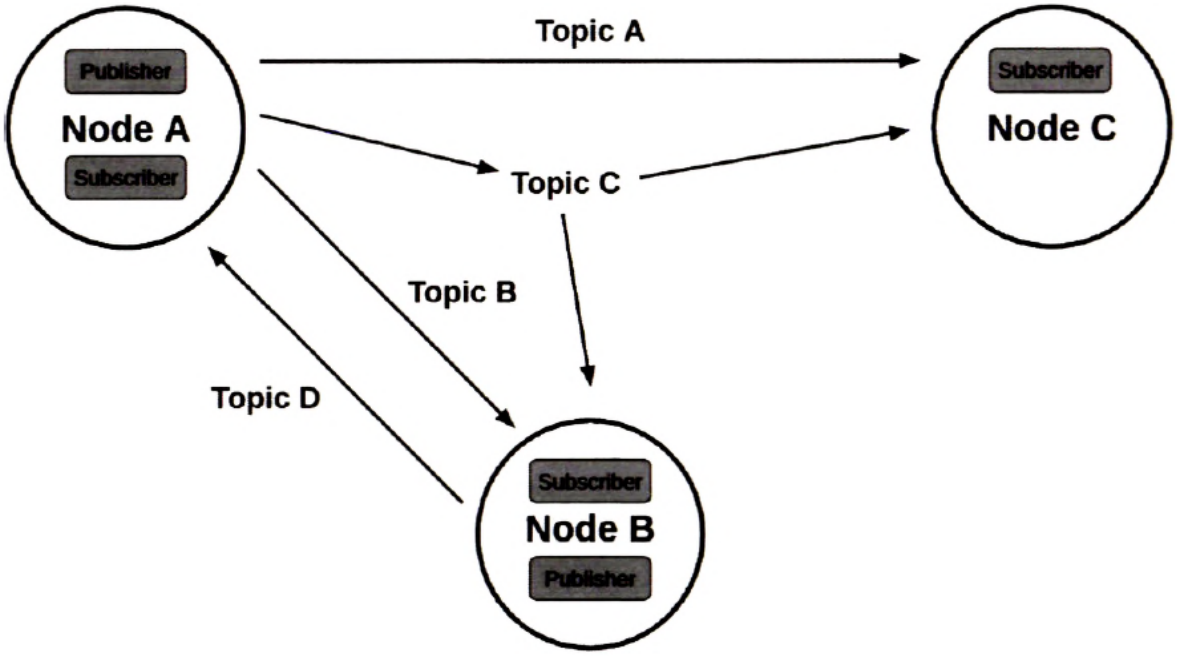
토픽

- 토픽(Topic)은 비동기식 단방향 메시지 송수신 방식. msg 인터페이스 형태의 메시지를 퍼블리시하는 Publisher와 메시지를 서브스크라이브하는 Subscriber간의 통신
- 1:1 통신을 기본으로 하지만 1:N도 가능하고 그 구성 방식에 따라 N:1, N:N 도 가능
- 토픽 기능은 목적에 따라 다양한 방법으로 사용 가능
- 기본 특징으로 비동기성과 연속성을 가지기에 센서 값 전송 및 항시 정보를 주고받아야 하는 부분에서 주로 사용

(퍼블리셔와 서브스크라이버)



(다자 간 통신)



토픽 대역폭 확인(ros2 topic bw)

메시지의 대역폭, 즉 송수신 받는 토픽 메시지의 크기를 확인

```
$ ros2 topic bw /turtle1/cmd_vel
```

토픽 주기 확인(ros2 topic hz)

토픽의 전송 주기를 확인하기 위해 ros2 topic hz 명령어 사용

```
$ ros2 topic hz /turtle1/cmd_vel
```

토픽 지연 시간 확인(ros2 topic delay)

토픽은 RMW 및 네트워크 장비를 거치기 때문에 지연 시간이 반드시 존재

```
$ ros2 topic delay /TOPIC_NAME
```

토픽 퍼블리시(ros2 topic pub)

ros2 topic pub 명령어에 토픽 이름, 토픽 메시지 타임, 메시지 내용 기술

```
$ ros2 topic pub <topic_name> <msg_type> “<args>”
```

bag 기록(ros2 bag record)

- rosbag : 퍼블리시되는 토픽을 파일 형태로 저장하고 필요할 때 저장된 토픽을 다시 불러와 동일한 주기로 재생할 수 있는 기능
- 디버깅에 큰 도움을 준다??
- ros2 bag rocord에 기록하고자 하는 토픽 이름을 기재

```
$ ros2 bag record <topic_name1> <topic_name2> <topic_name3>
```

ex) `$ ros bag record /turtle1/cmd_vel`

bag 정보(ros2 bag info)

- 저장된 rosbag 파일의 정보를 확인하기 위핸 명령어

```
$ ros2 bag info rosbag2
```

- 기록이 언제 시작되고 언제 끝났는지에 대한 타임스탬프와 취득한 토픽의 이름, 메시지 형태, 메시지 별 개수와 총 개수 등이 기록되어 있음

bag 재생(ros2 bag play)

- 순서 : turtlesim 노드를 종료한 후 다시 시작. 이후 명령어 입력

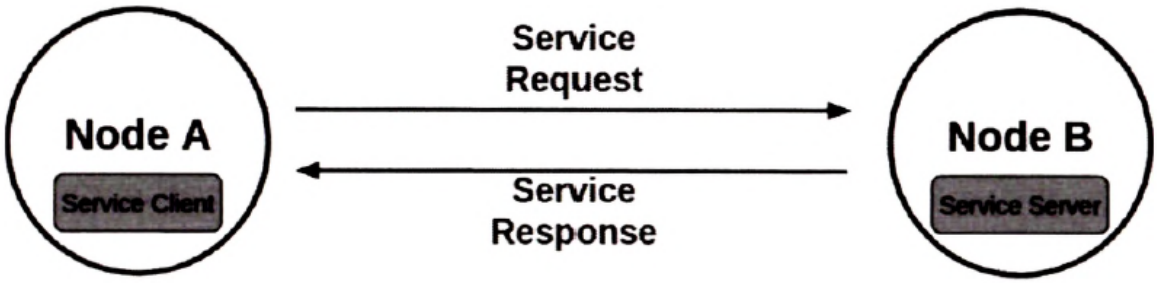
```
$ ros2 bag play rosbag2_****_**_**-**
```

[CH12] ROS 2 서비스

서비스

- 서비스(Service)는 동기식 양방향 메시지 송수신 방식으로 서비스 요청(Request)하는 쪽을 Service Client, 요청받은 서비스를 수행한 후 서비스 응답(Response)하는 쪽을 Service Server라고 한다.
- 서비스는 특정 요청을 하는 클라이언트단과 요청받은 일을 수행한 후에 결과값을 전달하는 서버 단과의 통신을 의미함

(서비스 통신에서의 서비스 서버와 서비스 클라이언트)



서비스 목록 확인(ros2 service list)

```
$ ros2 run turtlesim turtlesim_node
```

```
$ ros2 service list
```

서비스 형태 확인(ros2 service type)

```
$ ros2 service type /clear
std_srvs/srv/Empty
$ ros2 service type /kill
turtlesim/srv/kill
$ ros2 service type /spawn
turtlesim/srv/Spawn
```

서비스 찾기(ros2 service find)

서비스 형태 확인에서 언급한 서비스 형태 확인 명령어와 반대로 특정 형태를 입력하면 해당 형태의 서비스를 사용하는 서비스명을 확인할 수 있다.

```
$ ros2 service find std_srvs/srv/Empty
$ ros2 service find turtlesim/srv/Kill
```

서비스 요청(ros2 service call)

실제 서비스 서버에게 서비스 요청(Request)

아래 명령어를 이용하여 매개 변수로 서비스명, 서비스 형태, 서비스 요청 내용을 기술

```
$ ros2 service call <service_name> <service_type> “<arguments>”
```