

---

# ROS Package, Node, Topic

---

# Overview

---

- 개요
- ROS Package 소개, 실습
- ROS Node 소개, 실습
- ROS Topic 소개, 실습



# 개요

---

- ROS 용어

- **Node**

- 최소 단위의 실행 가능한 프로세스를 가리키는 용어로써 하나의 실행 가능한 프로그램으로 생각하면 된다. ROS 에서는 최소한의 실행단위로 프로그램을 나누어 작업하게 된다. 각 노드는 메시지 통신으로 데이터를 주고 받는다.

- **Package**

- 하나 이상의 노드, 노드 실행을 위한 정보 등을 묶어 놓은 것. 또한, 패키지의 묶음을 메타패키지라 하여 따로 분리한다.

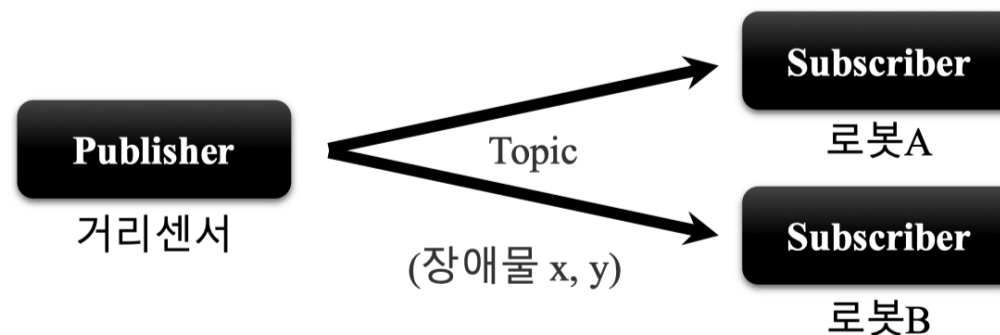
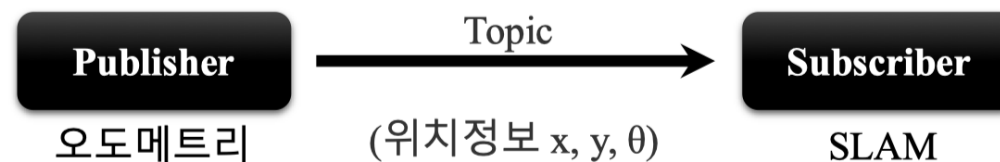
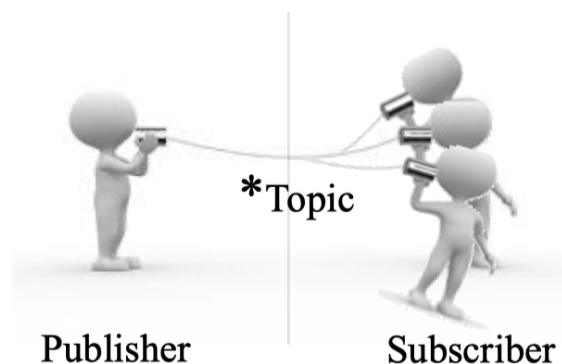
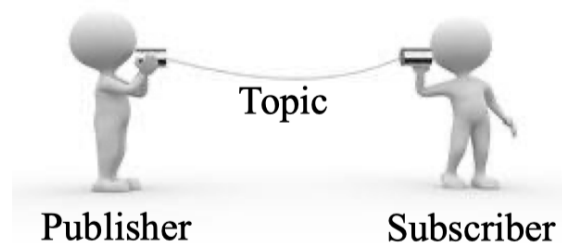
- **Message**

- 메시지를 통해 노드간의 데이터를 주고받게 된다. 메시지는 integer, floating point, boolean 와 같은 변수형태이다. 또한, 메시지 안에 메시지를 품고 있는 간단한 데이터 구조 및 메시지들의 배열과 같은 구조도 사용할 수 있다.



# 개요

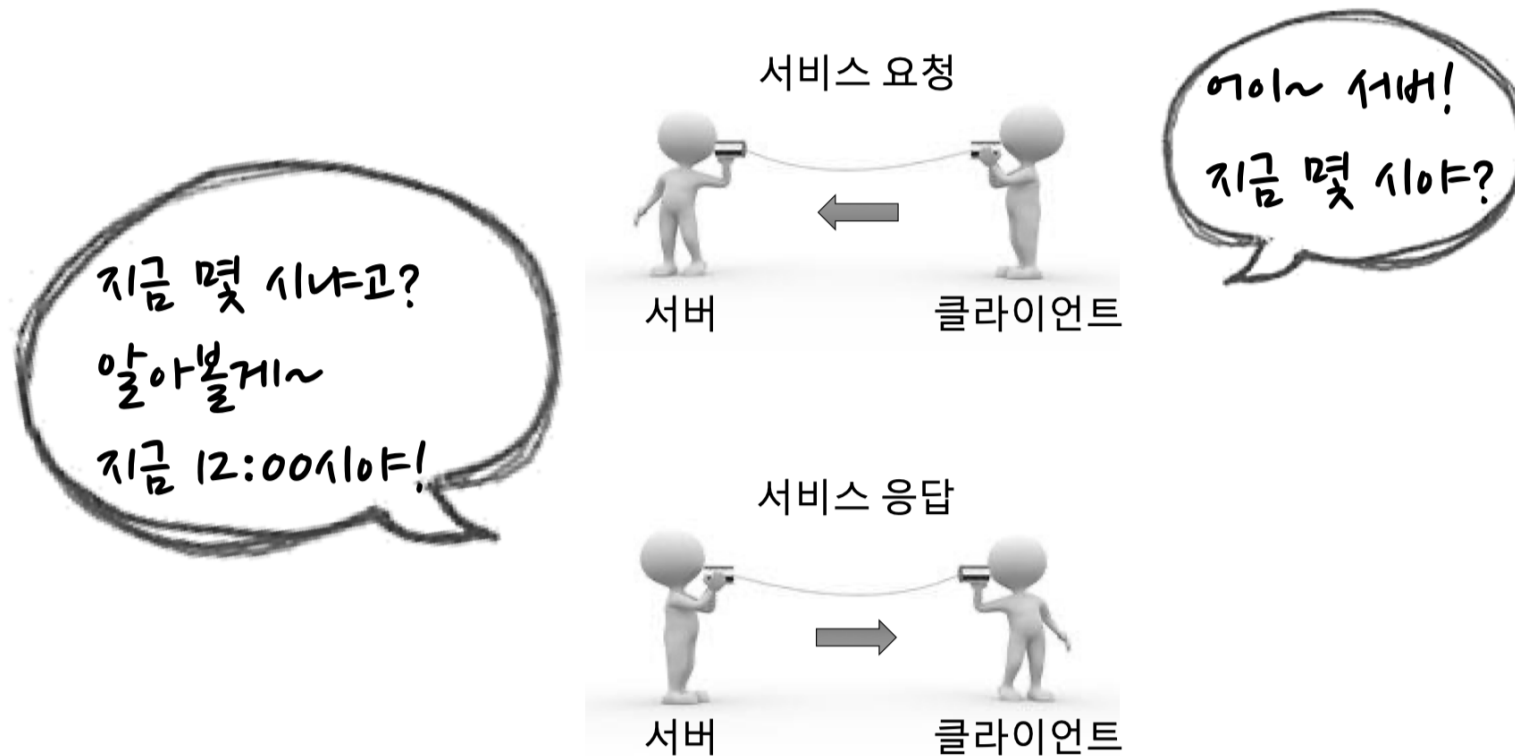
- ROS 용어
  - Topic, Publisher, Subscriber



\*Topic 에 대해 1:1의 Publisher, Subscriber 통신도 가능하며, 목적에 따라서 1:N, N:1, N:N 통신도 가능하다.

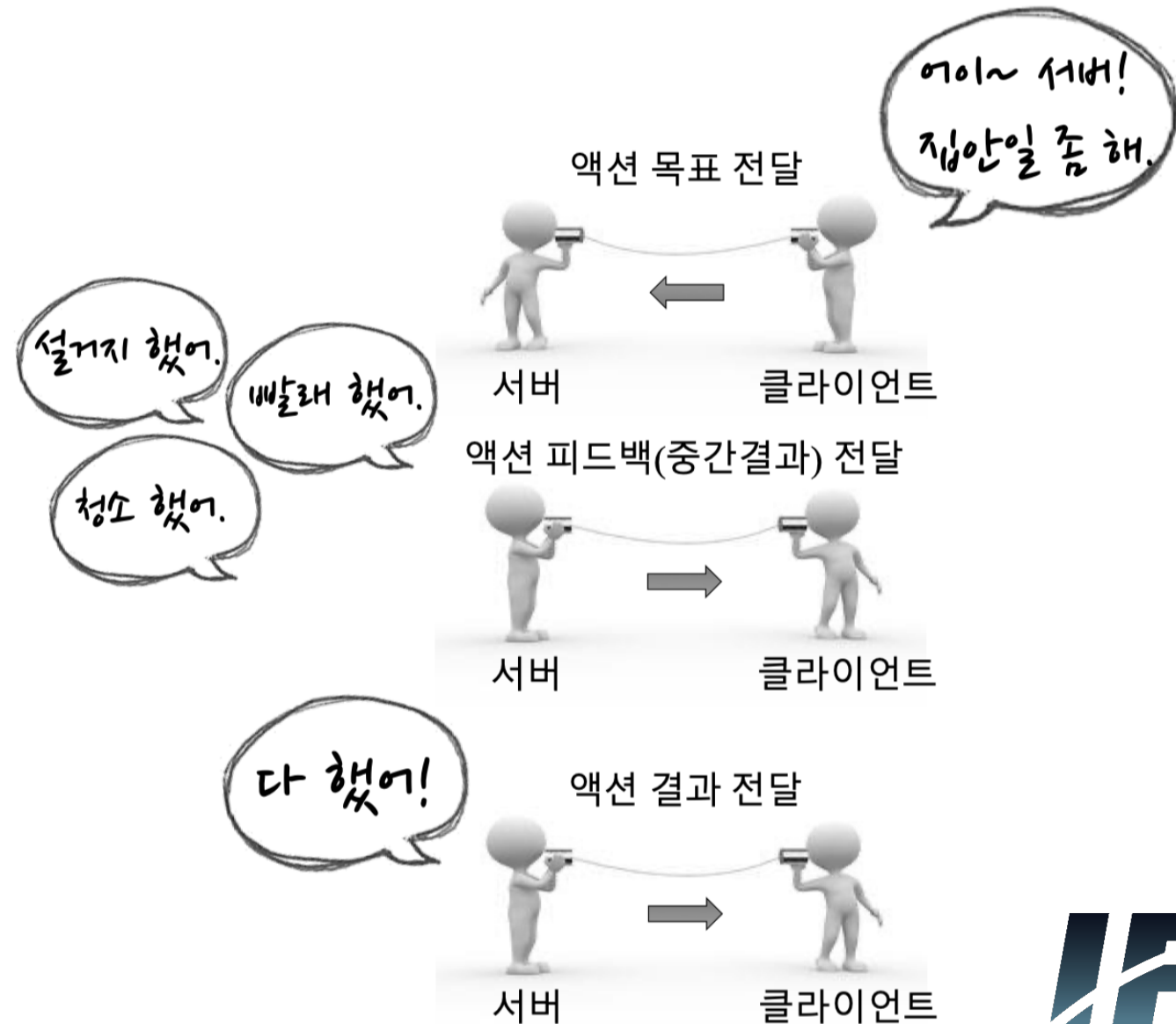
# 개요

- ROS 용어
  - Service, Service server, Service client



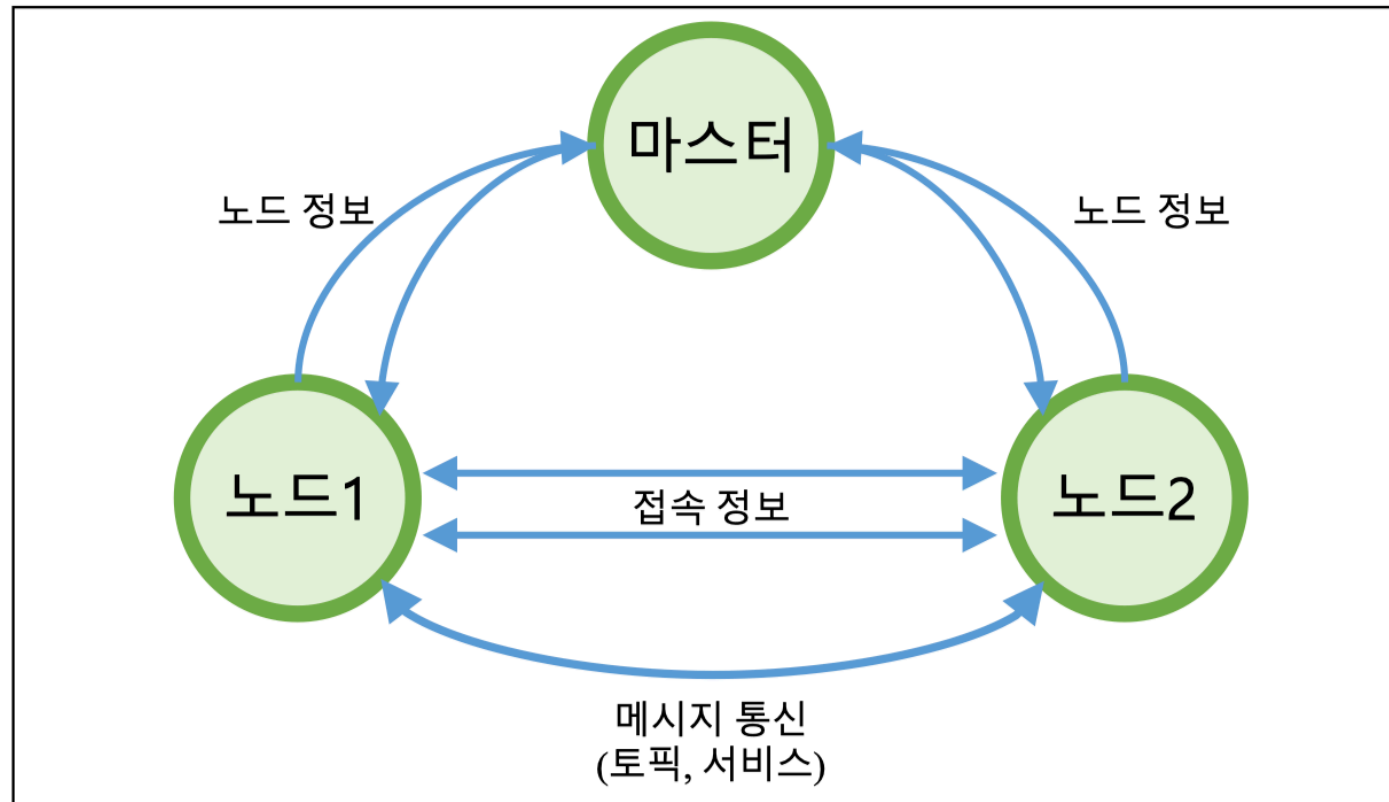
# 개요

- ROS 용어
  - Action, Action server, Action client



# 개요

- 메시지 통신 개념
  - ROS에서 가장 기본이 되는 기술적 포인트: **노드간의 메시지 통신!**



# 개요

---

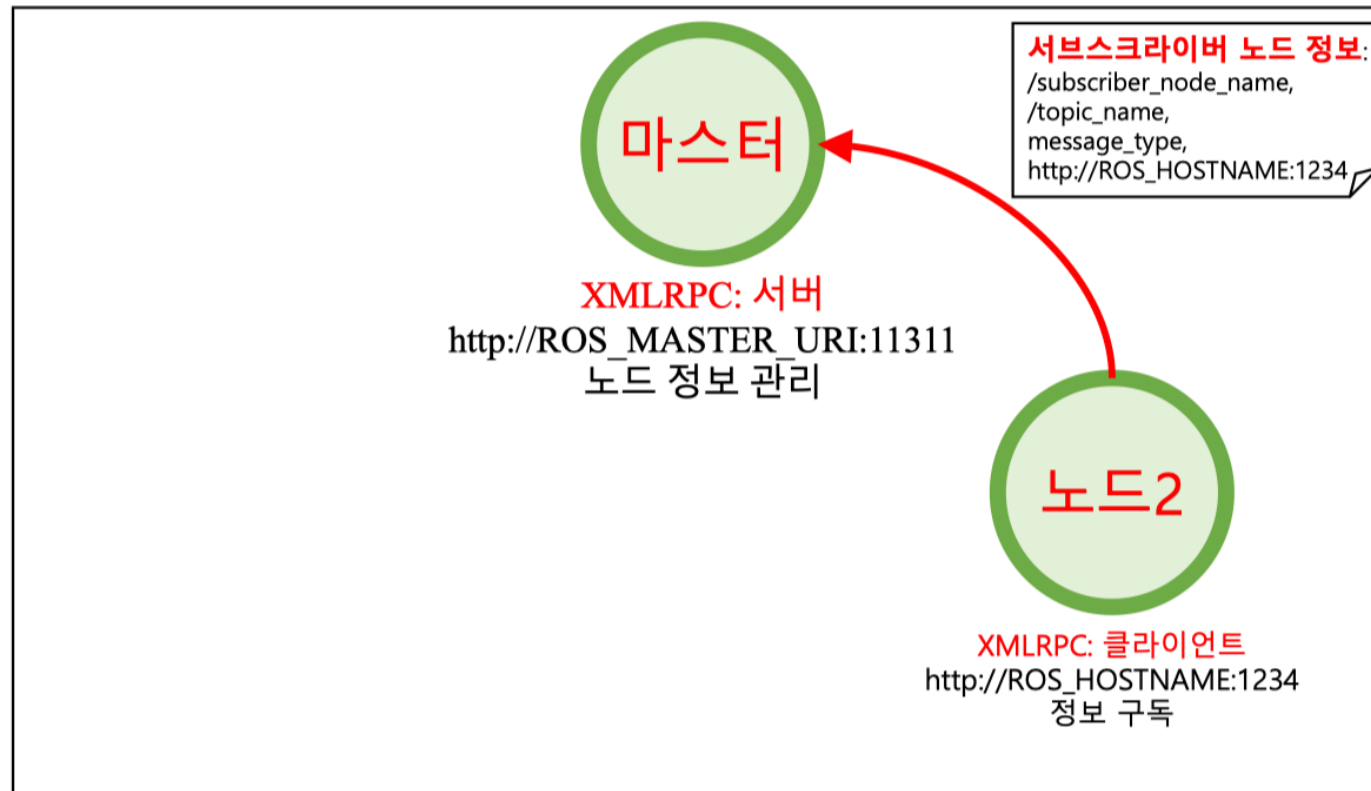
- 메시지 통신 개념
  - 1. 마스터 구동: XMLRPC(XML-Remote Procedure Call)
  - \$ roscore





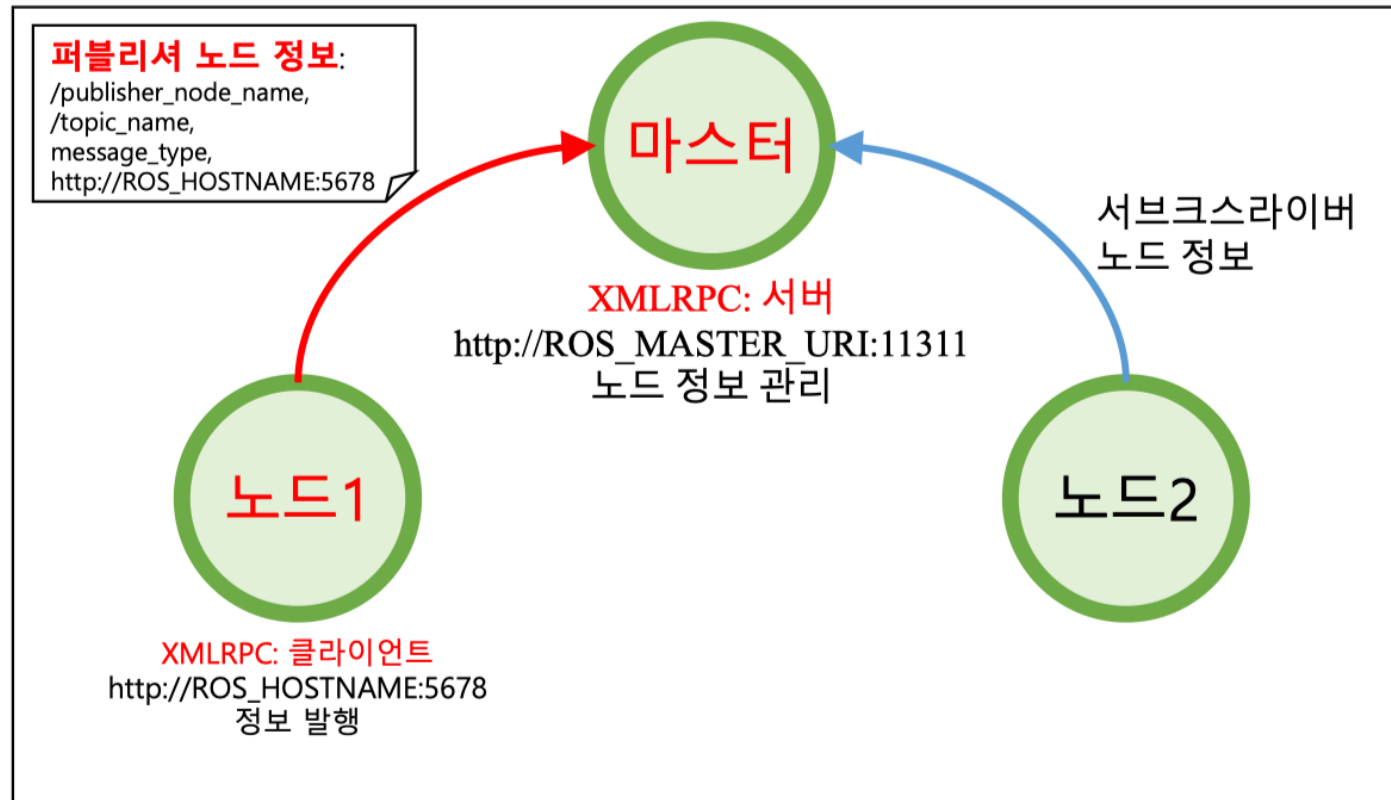
# 개요

- 메시지 통신 개념
  - 2. 서브스크라이버 노드(Node) 구동
  - \$roslaunch 패키지이름 노드이름



# 개요

- 메시지 통신 개념
  - 3. 퍼블리셔 노드(Node) 구동
  - \$roslaunch 패키지이름 노드이름

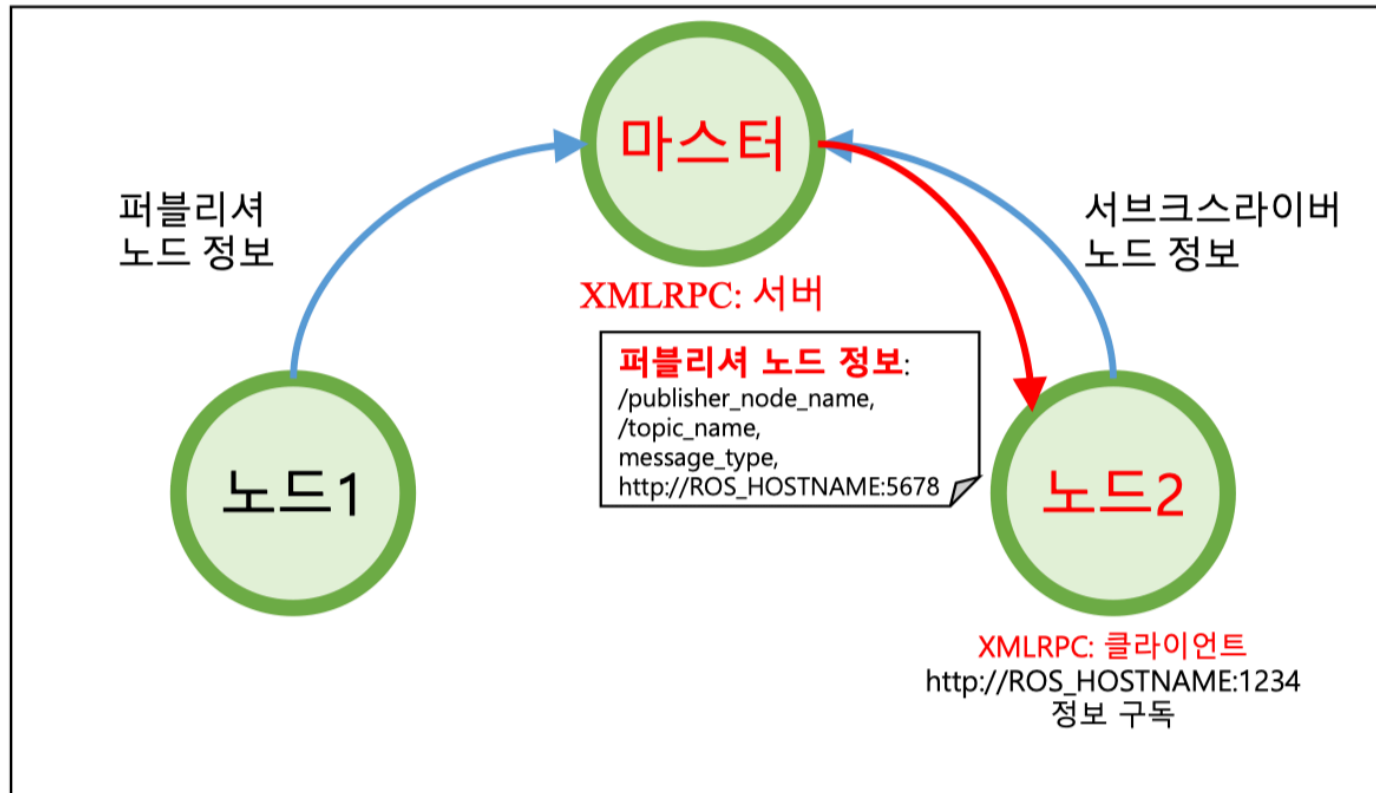


# 개요

- 메시지 통신 개념

- 4. 퍼블리셔 정보 알림

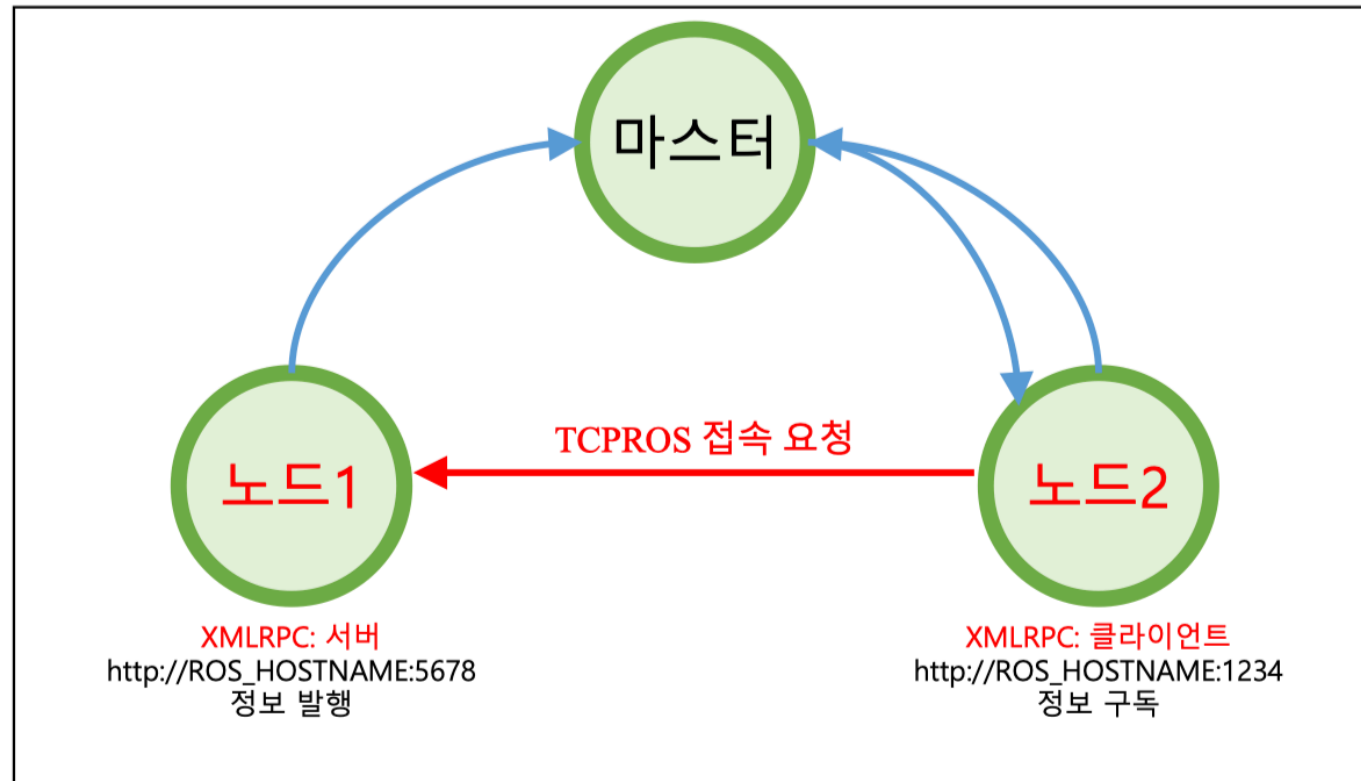
- 마스터는 서브스크라이버 노드에게 새로운 퍼블리셔 정보를 알린다.



# 개요

- 메시지 통신 개념

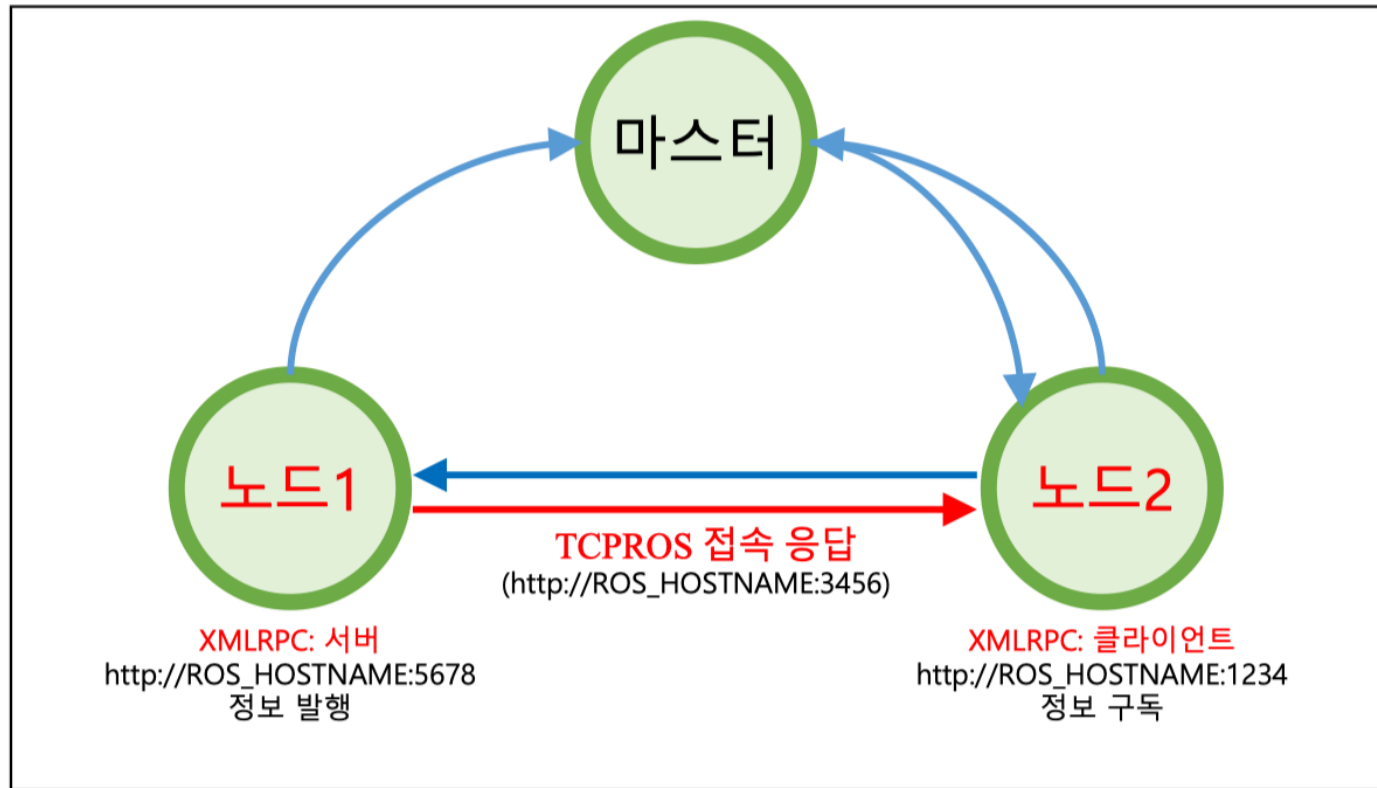
- 5. 퍼블리셔 노드에 접속 요청
- 마스터로부터 받은 퍼블리셔 정보를 이용하여 TCPROS 접속을 요청



# 개요

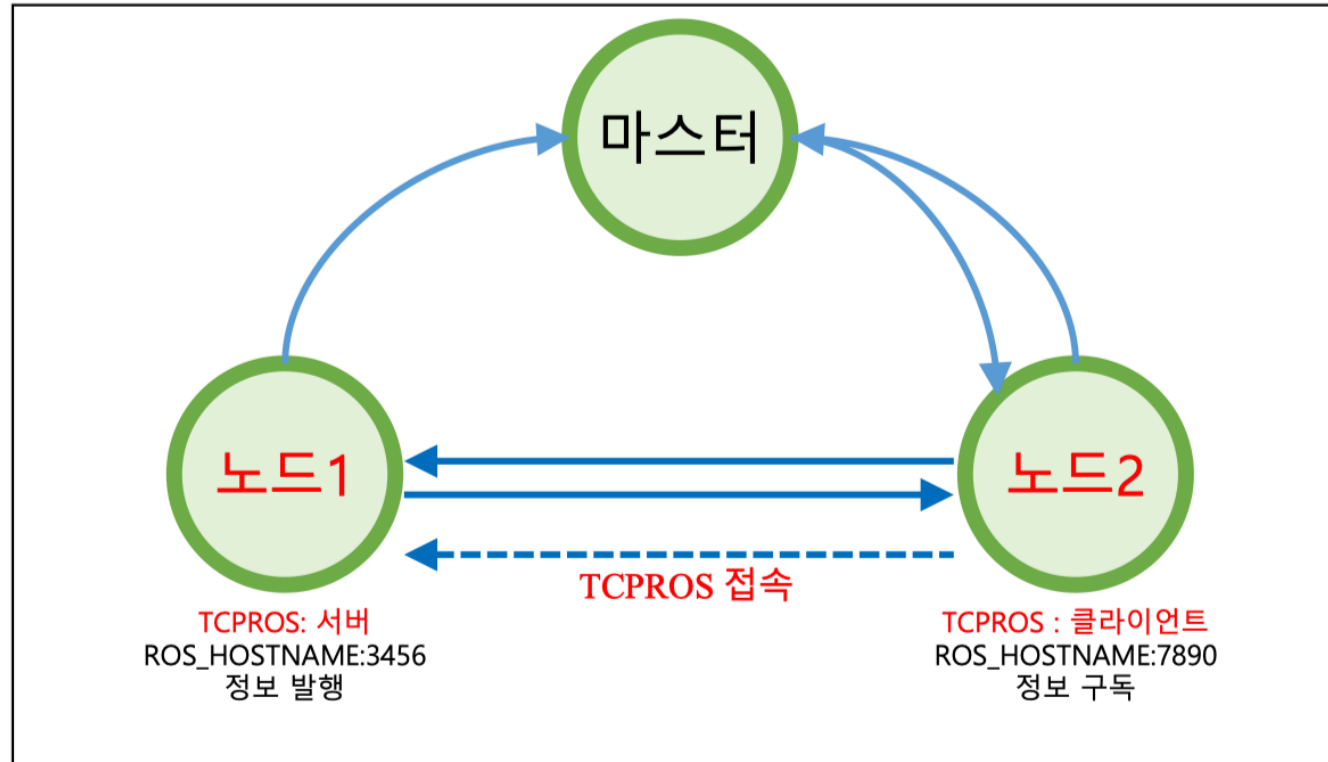
- 메시지 통신 개념

- 6. 서브스크라이버 노드에 접속 응답
- 접속 응답에 해당되는 자신의 TCP URI 주소와 포트번호를 전송



# 개요

- 메시지 통신 개념
  - 7. TCP 접속
  - TCPROS를 이용하여 퍼블리셔 노드와 직접 연결한다.

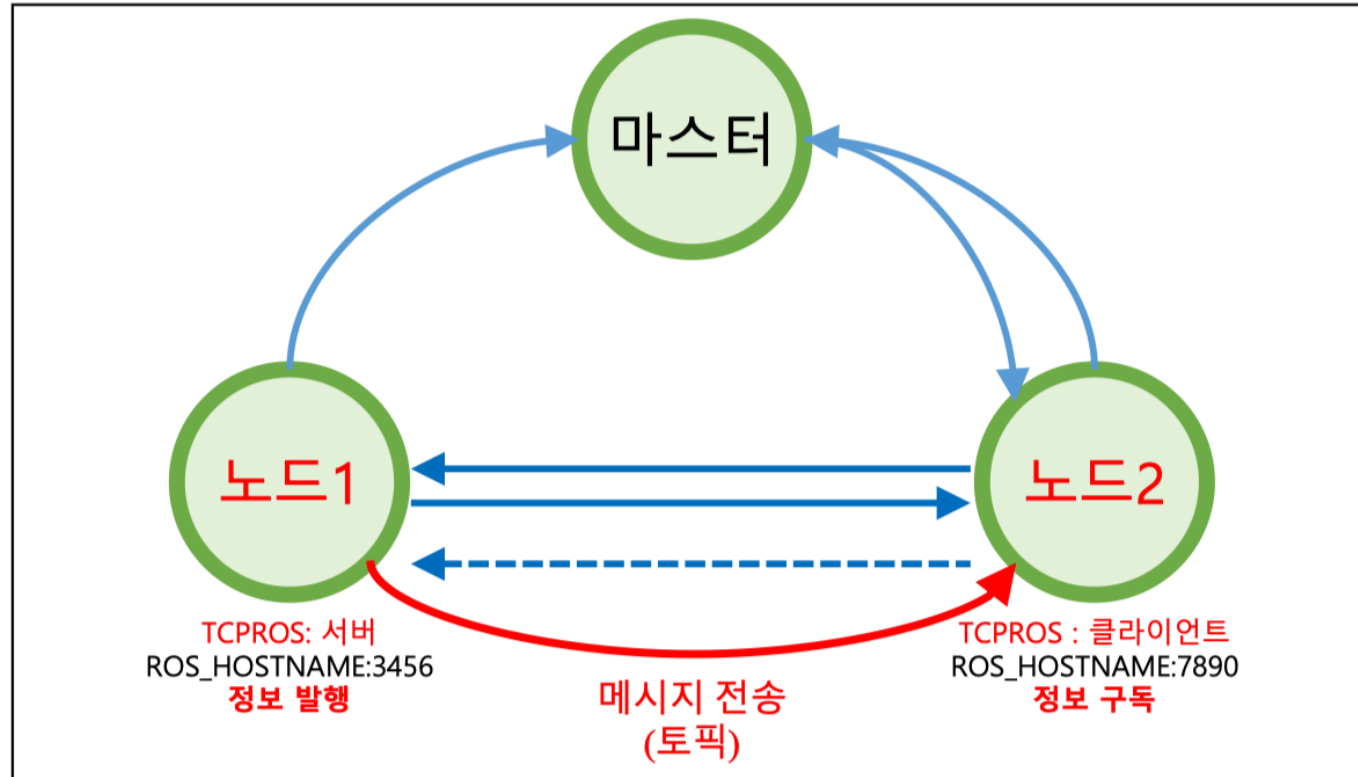


# 개요

- 메시지 통신 개념

- 8. 메시지 전송

- 발행자 노드는 서브스크라이버 노드에게 메시지를 전송 (토픽)

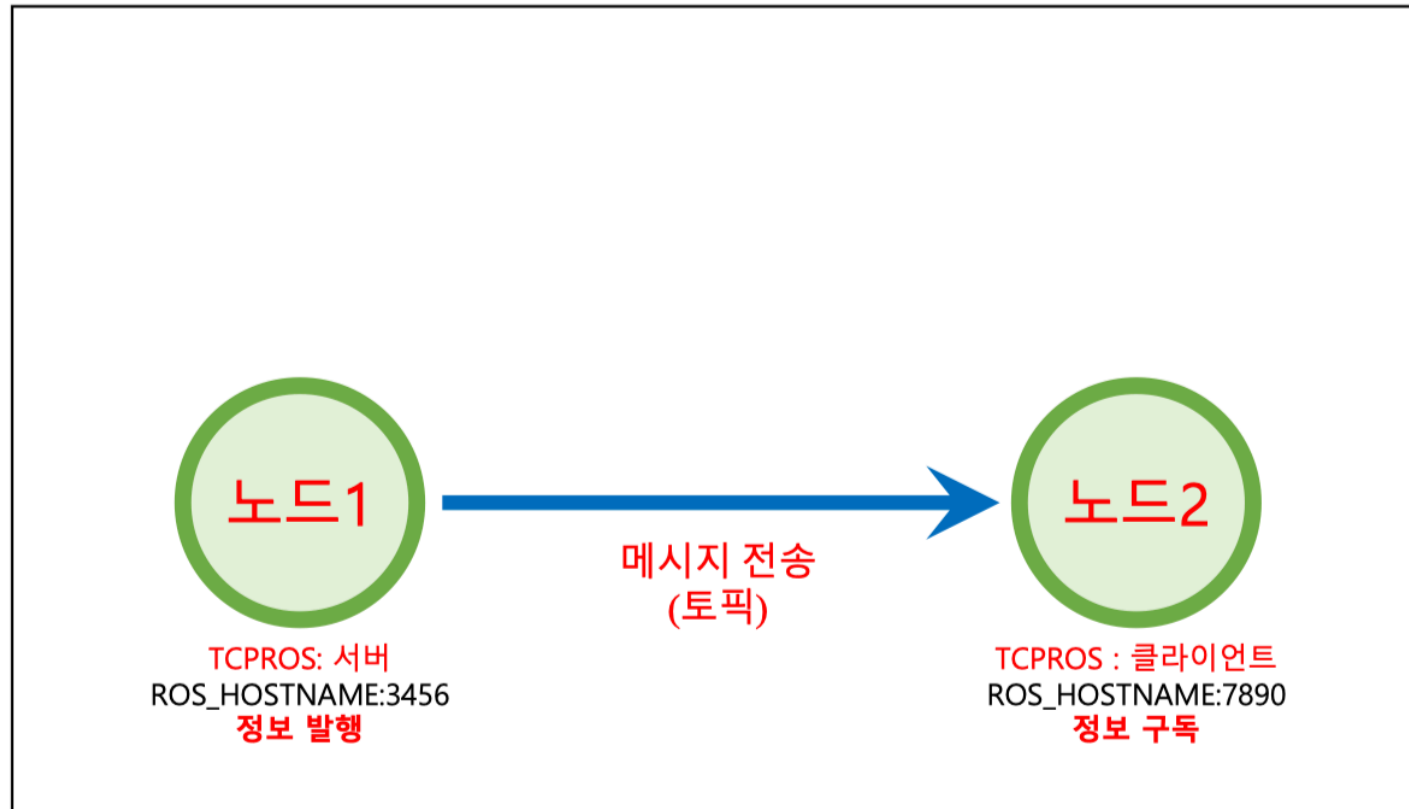


# 개요

- 메시지 통신 개념

- 8. 메시지 전송

- 토픽방식에서는 접속을 끊지 않는 이상 지속적으로 메시지를 전송한다. (연속성)



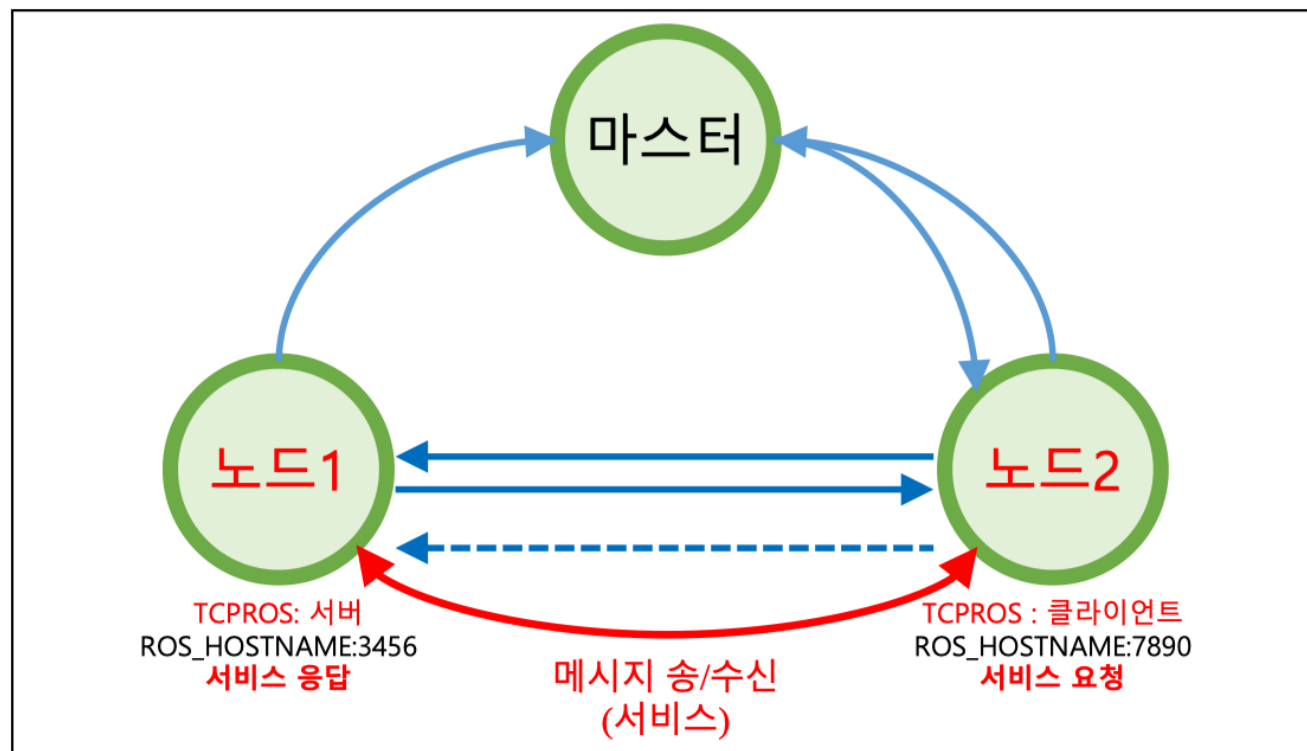


# 개요

- 메시지 통신 개념

- 서비스 요청 및 응답

- 1회에 한해 접속, 서비스 요청 및 서비스 응답이 수행되고 서로간의 접속을 끊는다.

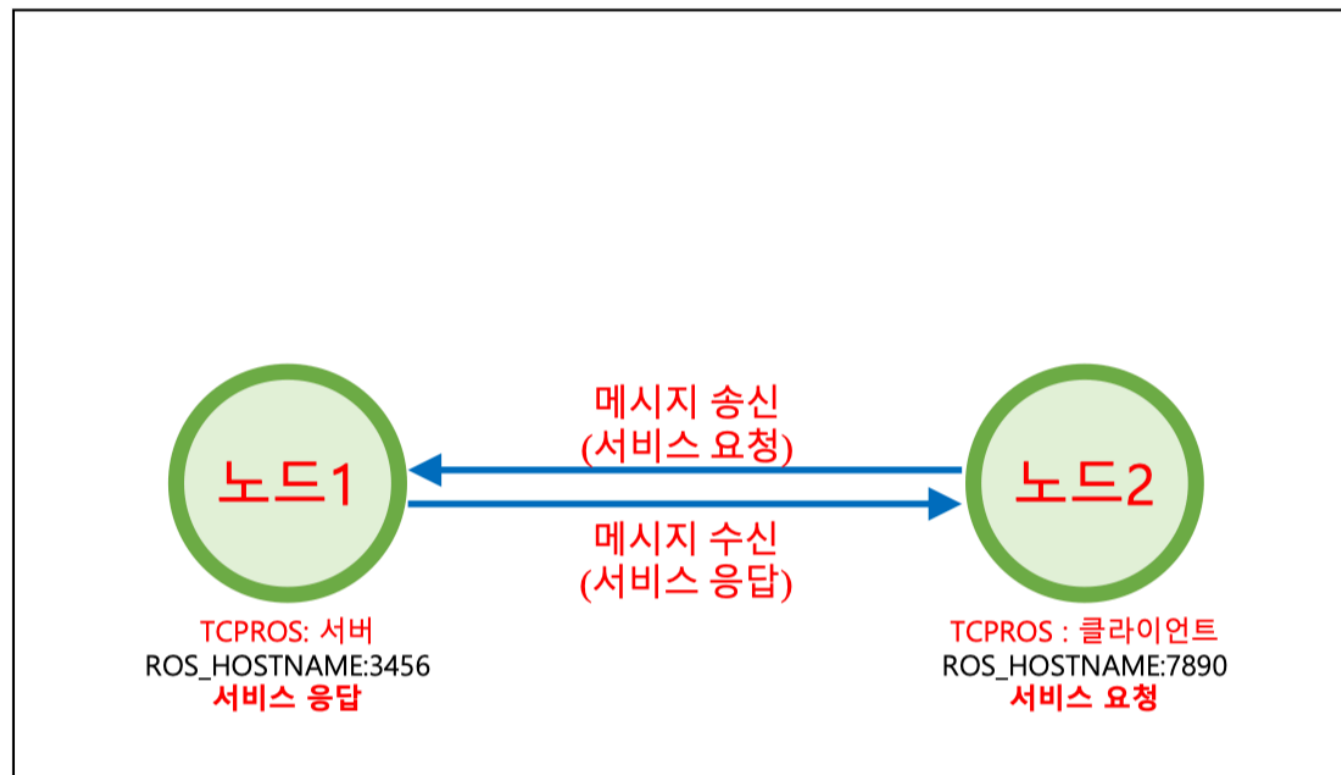


# 개요

- 메시지 통신 개념

- 서비스 요청 및 응답

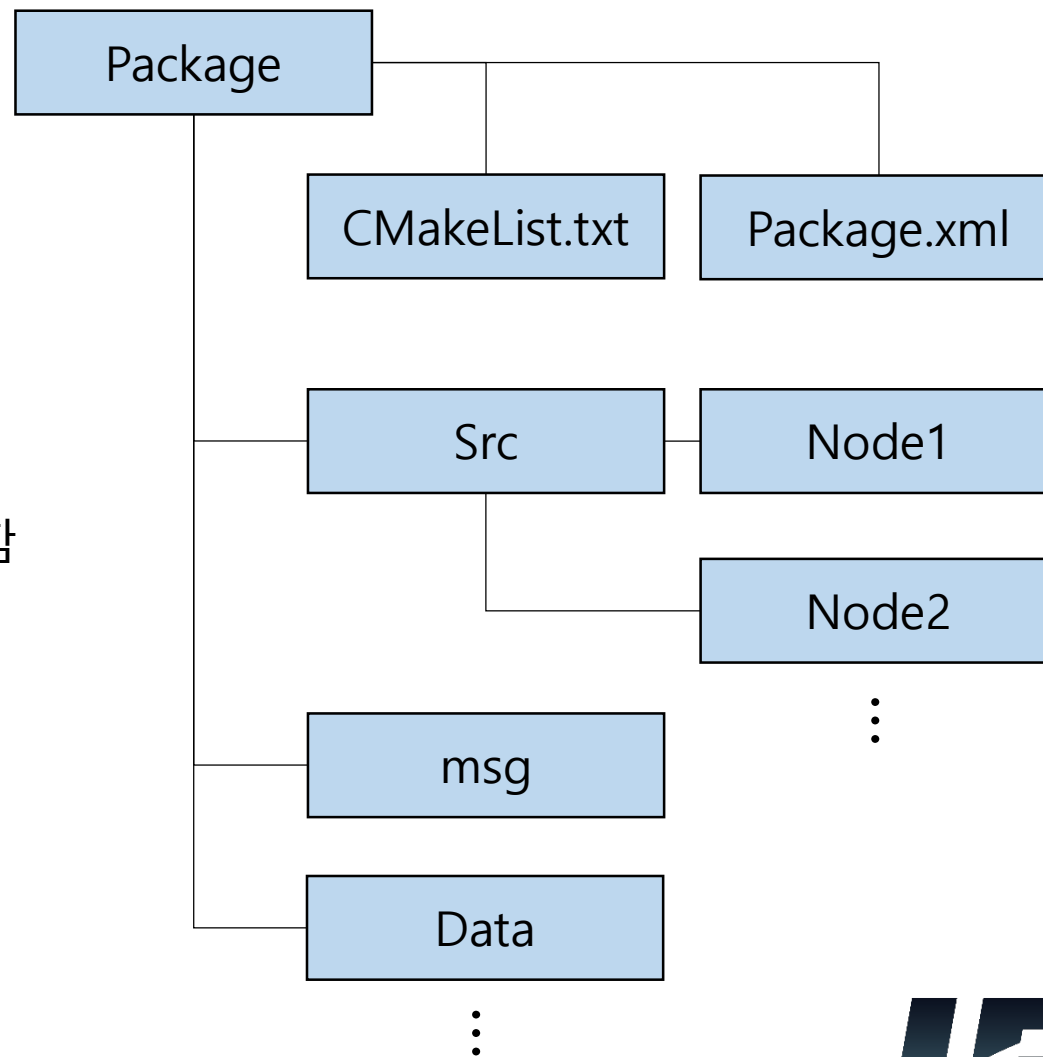
- 서비스는 토픽과 달리 1회에 한해 접속하고 서비스 요청 및 서비스 응답이 수행한 후 서로간의 접속을 끊는다. (1회성)



# ROS Package

- ROS Package

- ROS 의 Software 구성단위
- Node, dependency lib. 등 을 포함
- 기본적으로 CMake build system을 사용
- Package의 package.xml에는 메타데이터를 포함



# ROS Package

---

- ROS Package 공통 파일, 폴더
  - include/package\_name: C++ include headers (make sure to export in the CMakeLists.txt)
  - msg/: Folder containing Message (msg) types
  - src/package\_name/: Source files, especially Python source that are exported to other packages.
  - srv/: Folder containing Service (srv) types
  - scripts/: executable scripts
  - CMakeLists.txt: CMake build file (see catkin/CMakeLists.txt)
  - package.xml: Package catkin/package.xml
  - CHANGELOG.rst: Many packages will define a changelog which can be automatically injected into binary packaging and into the wiki page for the package



# ROS Package

- Create package
  - Workspace 생성

```
a@a: ~/test_ws
a@a:~$ mkdir -p test_ws/src
a@a:~$ cd test_ws/
a@a:~/test_ws$ catkin_make
Base path: /home/a/test_ws
Source space: /home/a/test_ws/src
Build space: /home/a/test_ws/build
Devel space: /home/a/test_ws/devel
Install space: /home/a/test_ws/install
Creating symlink "/home/a/test_ws/src/CMakeLists.txt" pointing to "/opt/ros/kinetic/share/catkin/cmake/toplevel.cmake"
####
#### Running command: "cmake /home/a/test_ws/src -DCATKIN_DEVEL_PREFIX=/home/a/test_ws/devel -DCMAKE_INSTALL_PREFIX=/home/a/test_ws/install -G Unix Makefiles" in "/home/a/test_ws/build"
####
-- The C compiler identification is GNU 5.5.0
-- The CXX compiler identification is GNU 5.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
```

```
$ mkdir -p <workspaceName>/src
$ cd <workspaceName>
$ catkin_make
```



# ROS Package

- Create package

- 기본적인 dependency를 갖는 package 생성

```
$ cd <workspaceName>/src
```

```
$ catkin_create_pkg <packageName> <dependency1> ...
```

- Catkin\_make로 빌드

```
$ cd <workspaceName>
```

```
$ catkin_make
```

```
a@a: ~/test_ws/src
a@a: ~/test_ws/src 80x24
a@a:~/test_ws$ cd src/
a@a:~/test_ws/src$ catkin_create_pkg testpkg roscpp rospy std_msgs
Created file testpkg/package.xml
Created file testpkg/CMakeLists.txt
Created folder testpkg/include/testpkg
Created folder testpkg/src
Successfully created files in /home/a/test_ws/src/testpkg. Please adjust the values in package.xml.
a@a:~/test_ws/src$
```

```
a@a: ~/test_ws
a@a: ~/test_ws 80x24
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/a/test_ws/build/test_results
-- Found gtest sources under '/usr/src/gmock': gtests will be built
-- Found gmock sources under '/usr/src/gmock': gmock will be built
-- Found PythonInterp: /usr/bin/python2 (found version "2.7.12")
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.20
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
--
-- traversing 1 packages in topological order:
--   - testpkg
--
-- +++ processing catkin package: 'testpkg'
-- ==> add_subdirectory(testpkg)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/a/test_ws/build
####
#### Running command: "make -j8 -l8" in "/home/a/test_ws/build"
####
a@a:~/test_ws$
```

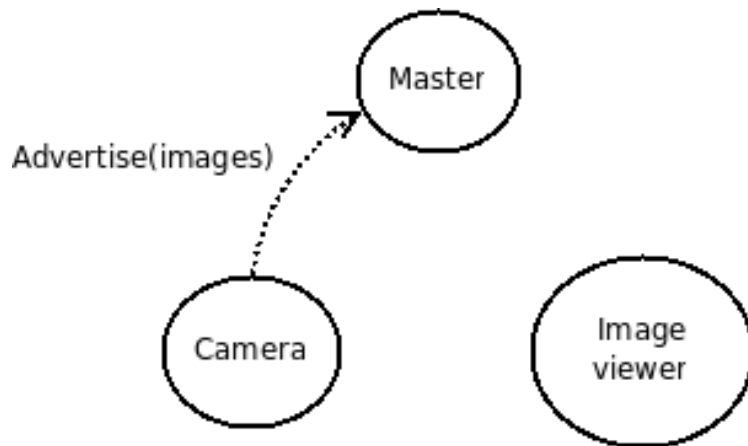
→ Package가 생성된 것을 확인

# ROS Package

---

- ROS Computation Graph Level

- Node : 계산을 수행하는 프로세스 단위, 모듈로 구성이 가능
- Master : 중앙서버역할, master에 등록되어야 통신이 가능
- Message : Node와 Node 간 통신에서 정의된 데이터 구조.
- Topic : 통신 시스템으로 전달되는 데이터의 이름. Node는 다른 Node와 master에 등록된 topic에 message를 publish함.

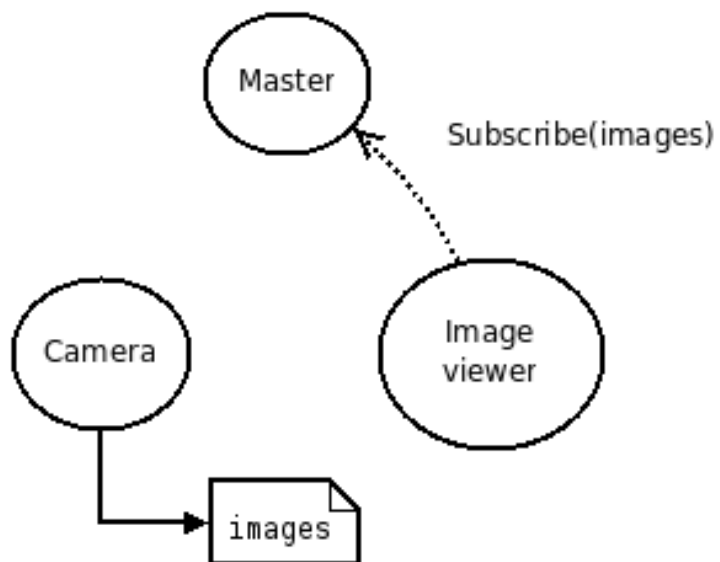


1. Camera node가 image topic을 publish 하기위해 master node에 등록.

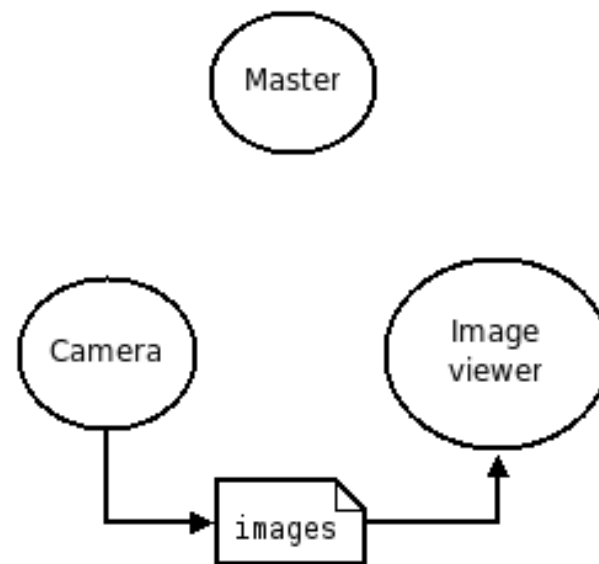
---

# ROS Package

- ROS Computation Graph Level



2. Image viewer node가 subscribe하기 위해 master에 알림



3. Image viewer node에 image topic의 publish를 알리며 topic이 전달됨



# ROS Node

- Create Node – 가장 기본적인 형태의 node 실습

- testpkg/src testnode.cpp 에 다음과 같이 작성.
- testpkg/CMakeList.txt 에 아래 두 줄 추가

```
//testnode.cpp
#include "ros/ros.h"

int main(int argc, char **argv)
{
    //master에 등록
    ros::init(argc, argv, "test");
    //timestamp와 함께 출력.
    ROS_INFO("Hello ROS!");

    return 0;
}
```

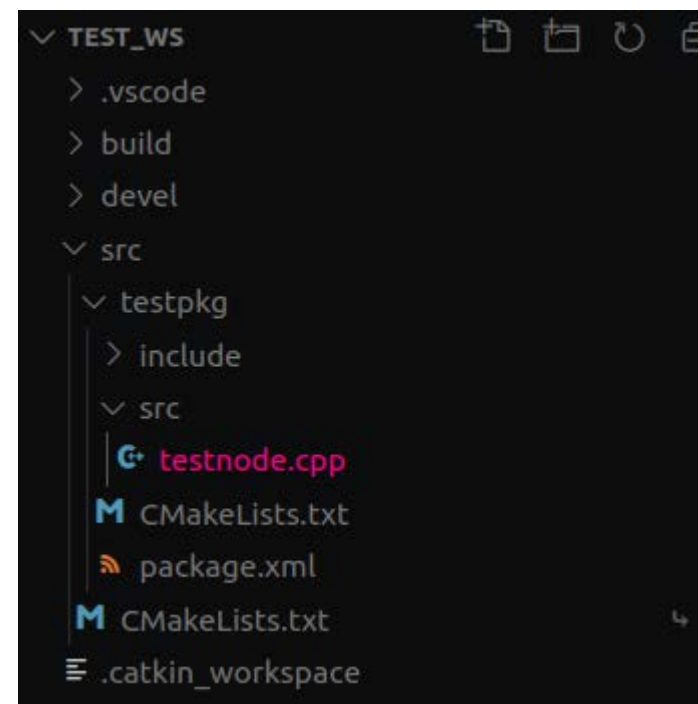
# CMakeList.txt 실행파일 (testnode.o)만들기 위한 설정

# CMakeList.txt의 가장 아래에 아래 두줄을 추가

```
add_executable(testnode src/testnode.cpp)
target_link_libraries(testnode ${catkin_LIBRARIES})
```

\* 간단한 형태의 Node 추가 방법

```
add_executable(<nodeName> <path of source>)
target_link_libraries(<nodeName> ${catkin_LIBRARIES})
```



# ROS Node

- Create Node – 가장 기본적인 형태의 node 실습

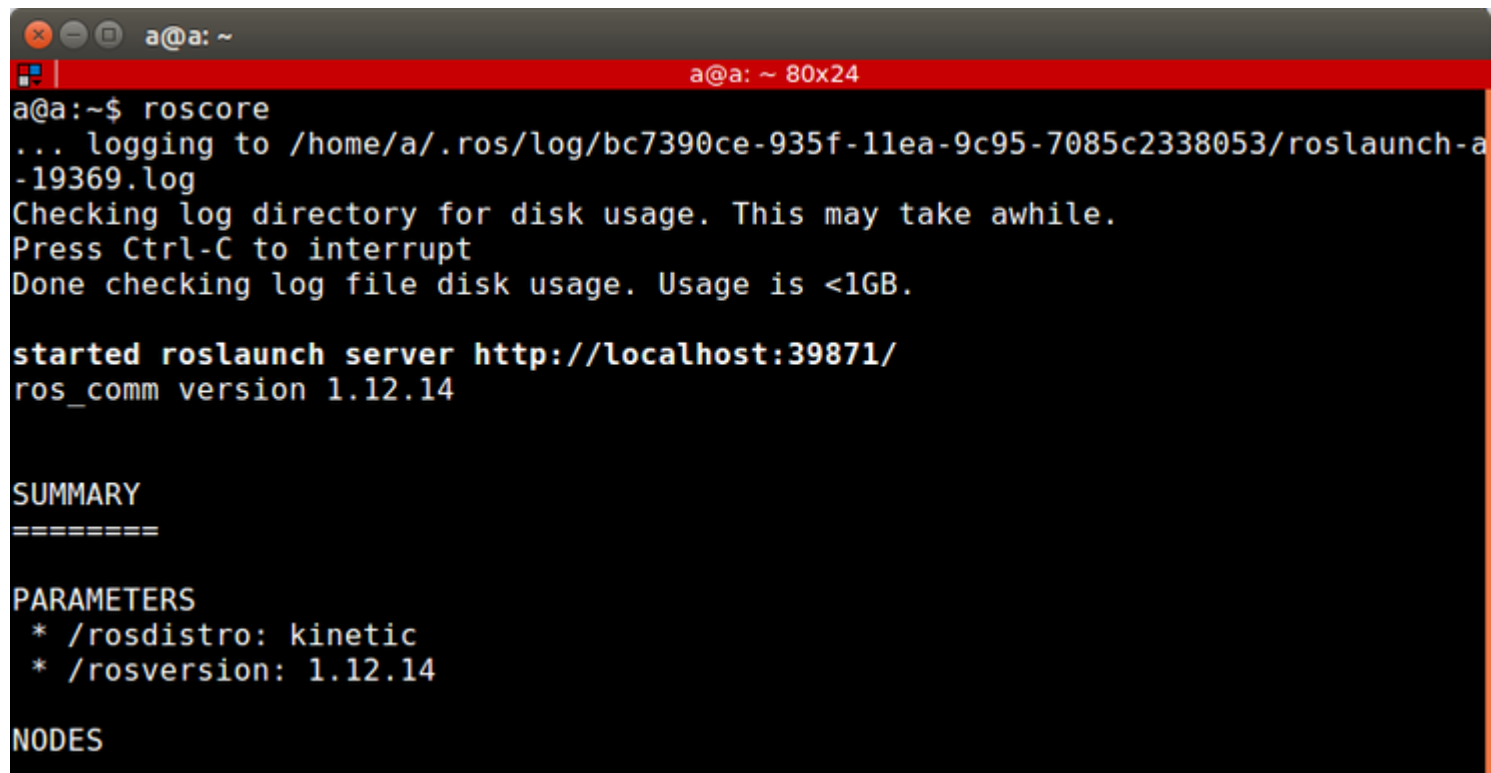
- 패키지의 CMakeList에 실행가능한 node를 만들기위해선 add\_executable, target\_link\_libraries를 작성해야함

```
190 # install(FILES
191 #   # myfile1
192 #   # myfile2
193 #   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
194 # )
195
196 #####
197 ## Testing ##
198 #####
199
200 ## Add gtest based cpp test target and link libraries
201 # catkin_add_gtest(${PROJECT_NAME}-test test/test_testpk
202 # if(TARGET ${PROJECT_NAME}-test)
203 #   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_
204 # endif()
205
206 ## Add folders to be run by python nosetests
207 # catkin_add_nosetests(test)
208
209 add_executable(testnode src/testnode.cpp)
210 target_link_libraries(testnode ${catkin_LIBRARIES})
211
212 add_executable(talkernode src/talker.cpp)
213 target_link_libraries(talkernode ${catkin_LIBRARIES})
```

# ROS Node

---

- Node 빌드 & 실행 방법
  - Roscore로 master 실행 한 후,

A terminal window with a dark background and a red title bar. The title bar contains window control icons and the text 'a@a: ~'. Below the title bar, the terminal shows the command 'roscore' being executed. The output includes logging information, a disk usage check, and the successful start of the ROS master. The terminal text is as follows:

```
a@a:~$ roscore
... logging to /home/a/.ros/log/bc7390ce-935f-11ea-9c95-7085c2338053/roslaunch-a
-19369.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:39871/
ros_comm version 1.12.14

SUMMARY
=====

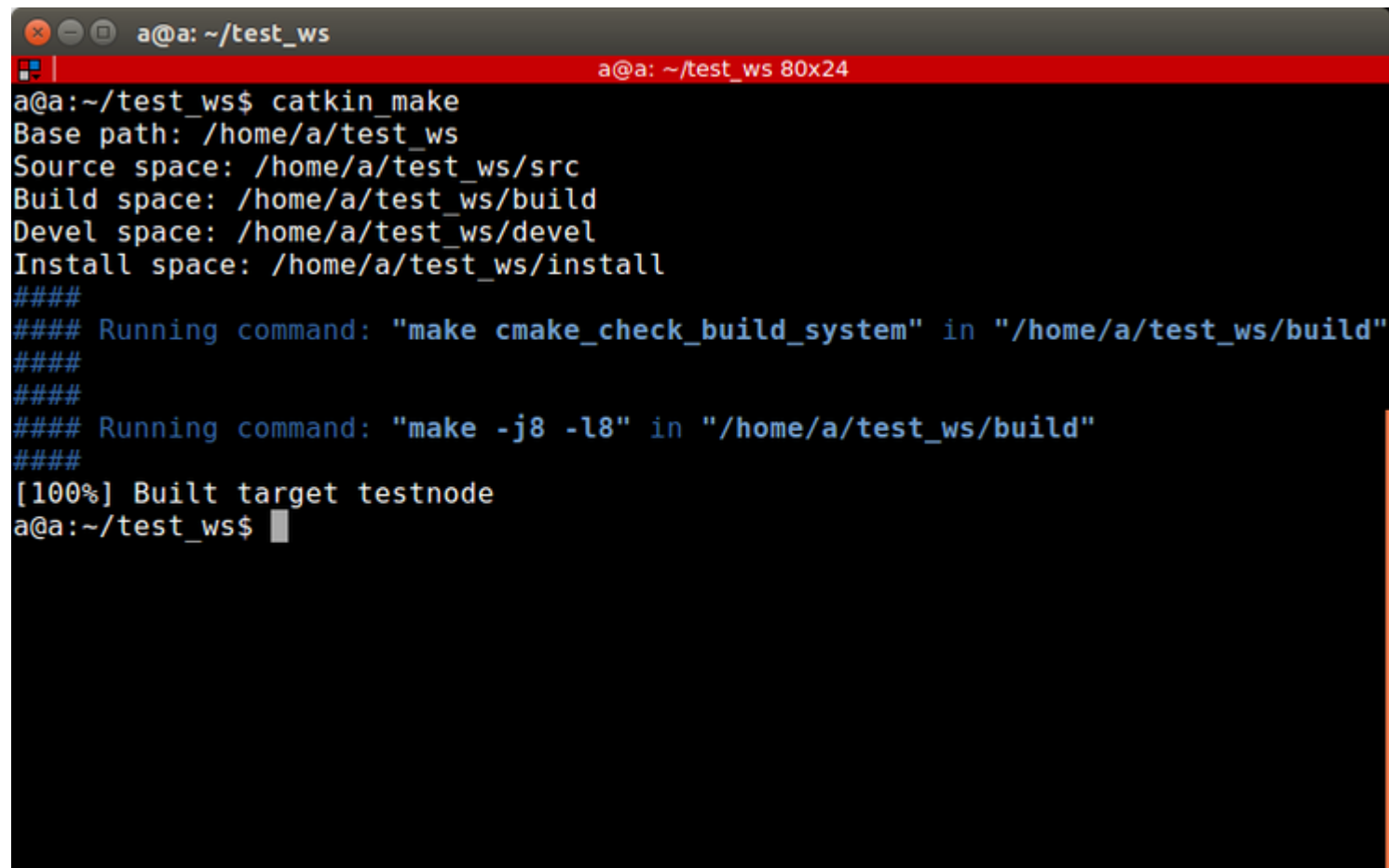
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.14

NODES
```

# ROS Node

- Node 빌드 & 실행 방법
  - Workspace에서 빌드 – 터미널을 추가로 실행

\$ catkin\_make

A terminal window with a dark background and a red title bar. The title bar contains the text 'a@a: ~/test\_ws' and 'a@a: ~/test\_ws 80x24'. The terminal shows the command 'catkin\_make' being executed. The output displays the workspace structure: Base path, Source space, Build space, Devel space, and Install space. It then shows the execution of 'make cmake\_check\_build\_system' and 'make -j8 -l8' in the build space, and finally reports '[100%] Built target testnode'.

```
a@a: ~/test_ws$ catkin_make
Base path: /home/a/test_ws
Source space: /home/a/test_ws/src
Build space: /home/a/test_ws/build
Devel space: /home/a/test_ws/devel
Install space: /home/a/test_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/a/test_ws/build"
####
####
#### Running command: "make -j8 -l8" in "/home/a/test_ws/build"
####
[100%] Built target testnode
a@a:~/test_ws$
```

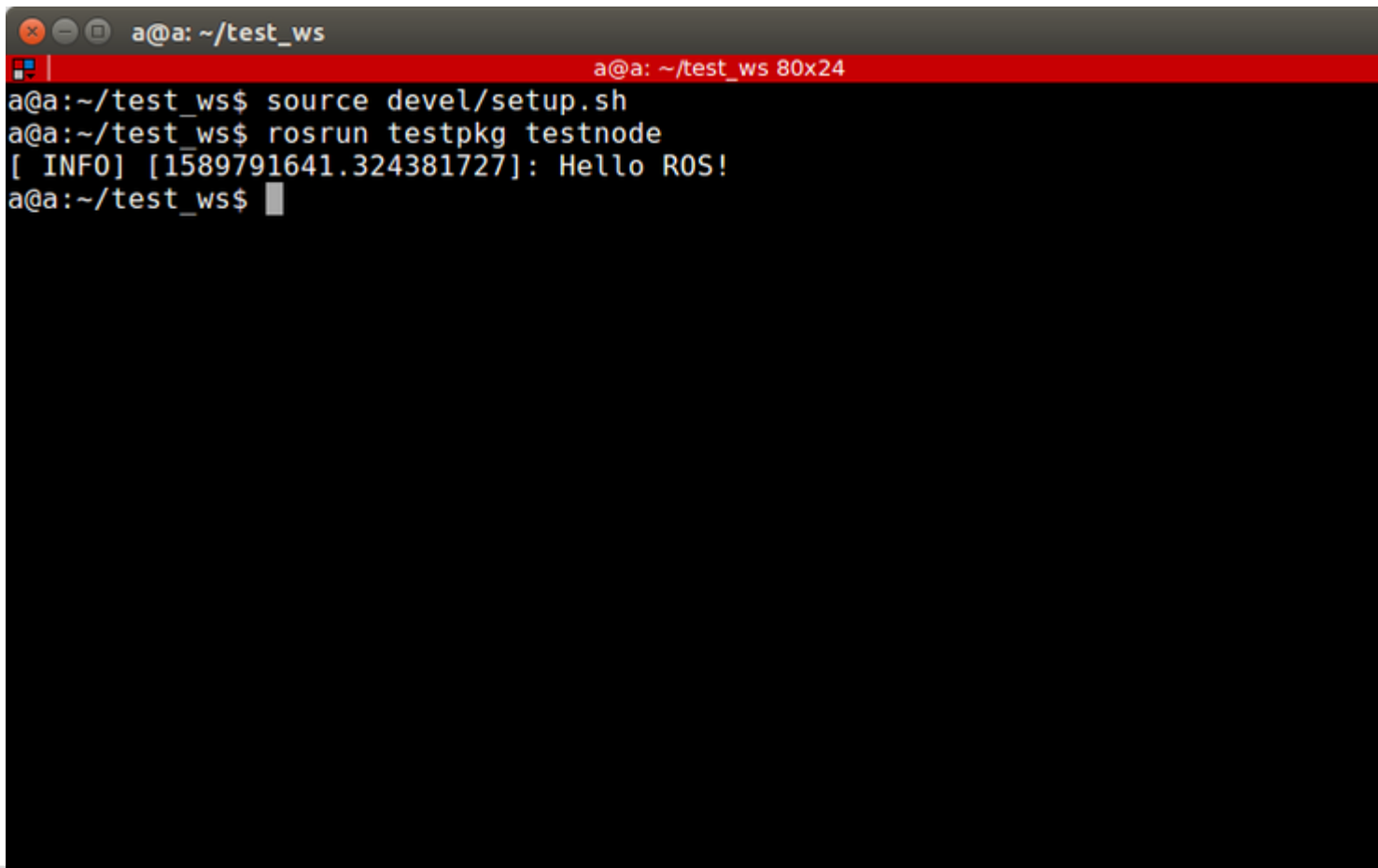
# ROS Node

- Node 빌드 & 실행 방법

- 현재 workspace를 source한 뒤 실행

```
roslaunch packageName nodeName
```

```
$ source devel/setup.sh  
$ roslaunch testpkg testnode
```

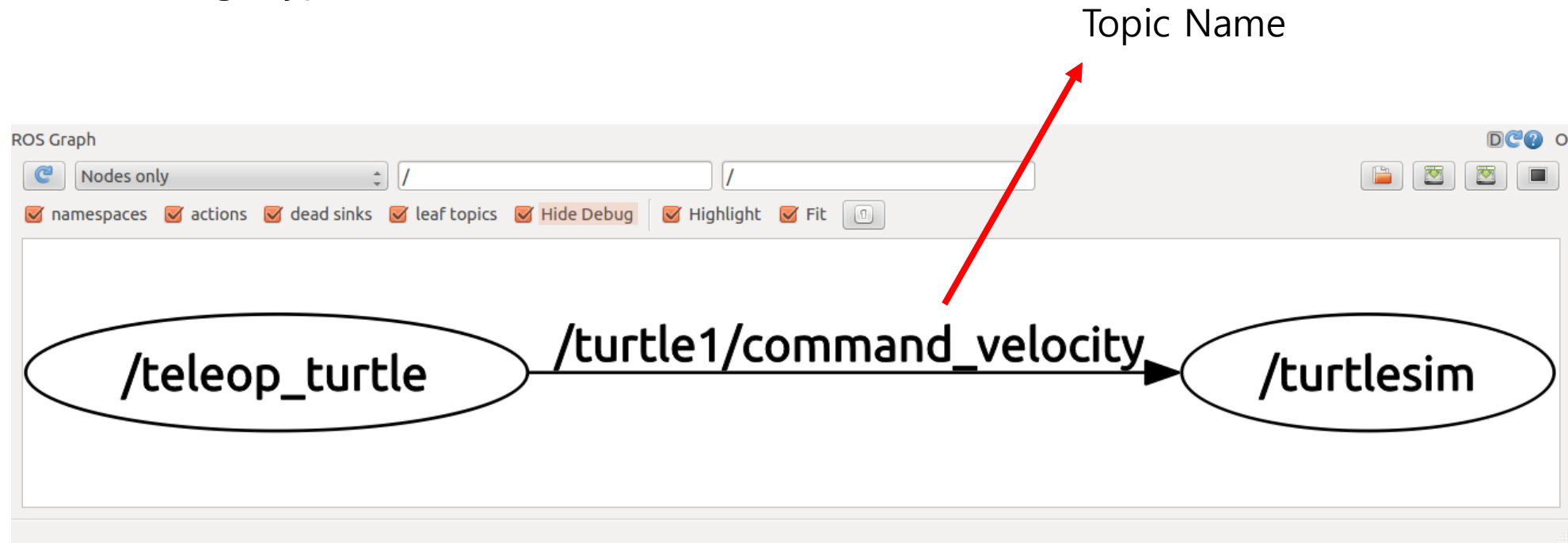
A terminal window with a black background and white text. The window title bar shows 'a@a: ~/test\_ws'. The terminal content shows the following commands and output:

```
a@a:~/test_ws$ source devel/setup.sh  
a@a:~/test_ws$ roslaunch testpkg testnode  
[ INFO] [1589791641.324381727]: Hello ROS!  
a@a:~/test_ws$
```

# ROS Topic

- ROS Topic

- Node가 Message를 교환 하는 버스. 익명으로 발행됨.
- 정의된 Message type이라면 임의의 구조로 발행/구독 가능



# ROS Topic

---

- Topic Understanding with turtlesim

- Install ros tutorials

```
$ sudo apt-get install ros-kinetic-ros-tutorials
```

- Turtlesim node 실행

```
$ rosrun turtlesim turtlesim_node
```

- Turtlesim teleop node 실행 (방향키로 조작)

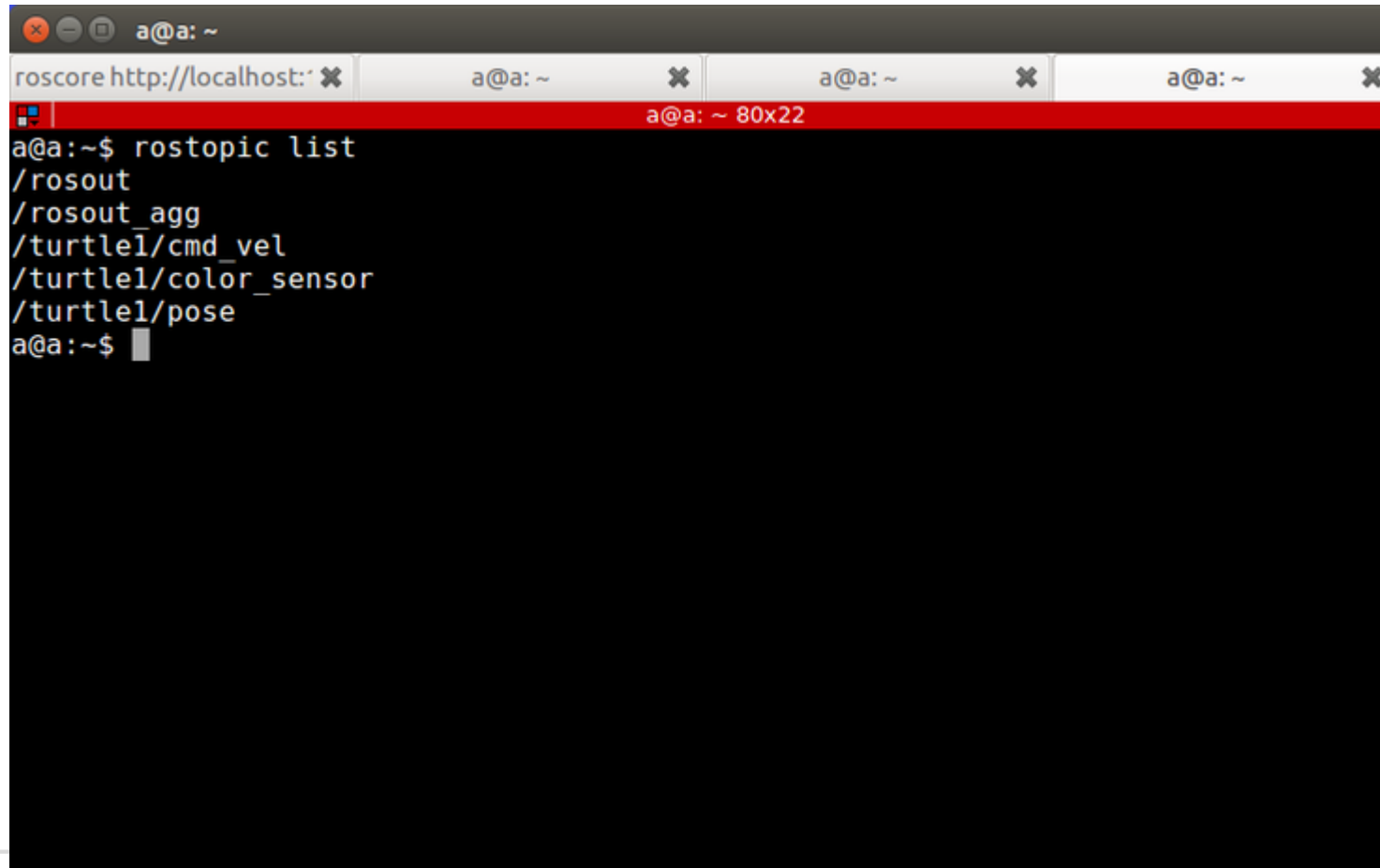
```
$ rosrun turtlesim turtle_teleop_key
```



# ROS Topic

---

- Rostopic command Line
  - rostopic list : 현재 활성화된 topic을 보여줌



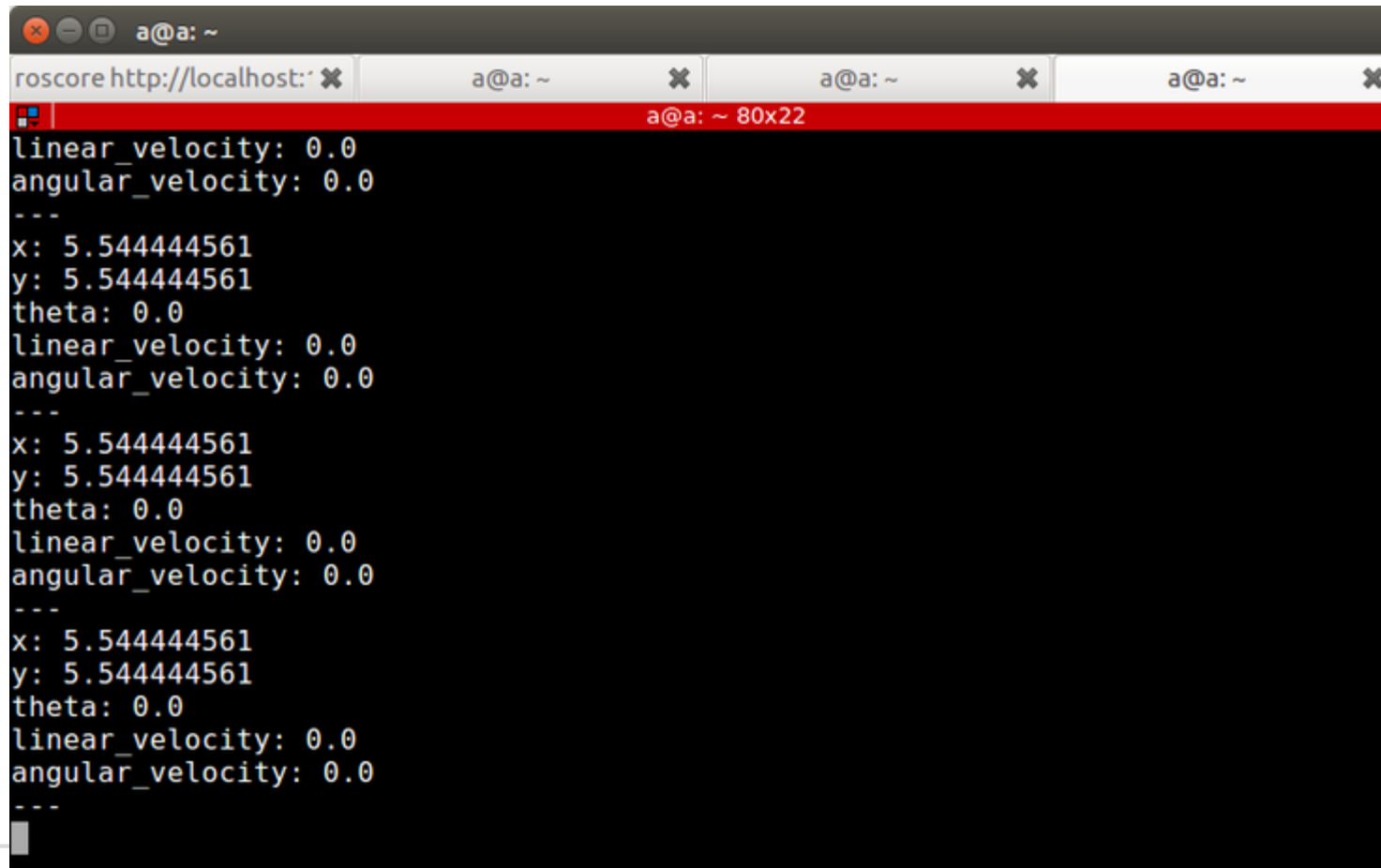
```
a@a: ~  
roscore http://localhost:~  
a@a: ~  
a@a: ~  
a@a: ~  
a@a: ~ 80x22  
a@a:~$ rostopic list  
/rosout  
/rosout_agg  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose  
a@a:~$
```



# ROS Topic

- Rostopic command Line

- rostopic echo [topic name] : publish된 topic의 내용을 보여줌.     \$ rostopic echo /turtle1/pose

A terminal window titled 'a@a: ~' with multiple tabs. The active tab shows the output of the command 'rostopic echo /turtle1/pose'. The output is a repeating sequence of turtle pose data. Each sequence starts with 'linear\_velocity: 0.0' and 'angular\_velocity: 0.0' on separate lines, followed by three dashes '---'. Then, the position and orientation are listed: 'x: 5.544444561', 'y: 5.544444561', and 'theta: 0.0'. This entire block of data is repeated four times in the visible output. The terminal has a red title bar and a black background with white text.

```
a@a: ~
roscore http://localhost:~
a@a: ~
a@a: ~
a@a: ~
a@a: ~ 80x22
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
```

# ROS Topic

- Rostopic command Line

- `rostopic pub [topic name] [message type] [message]:` topic을 publish.
- `publish` cf.) `--rate [hz]` : topic publish frequency option.

```
$ rostopic pub /testtopic std_msgs/String "hello" --rate 10    $ rostopic echo /testtopic
```

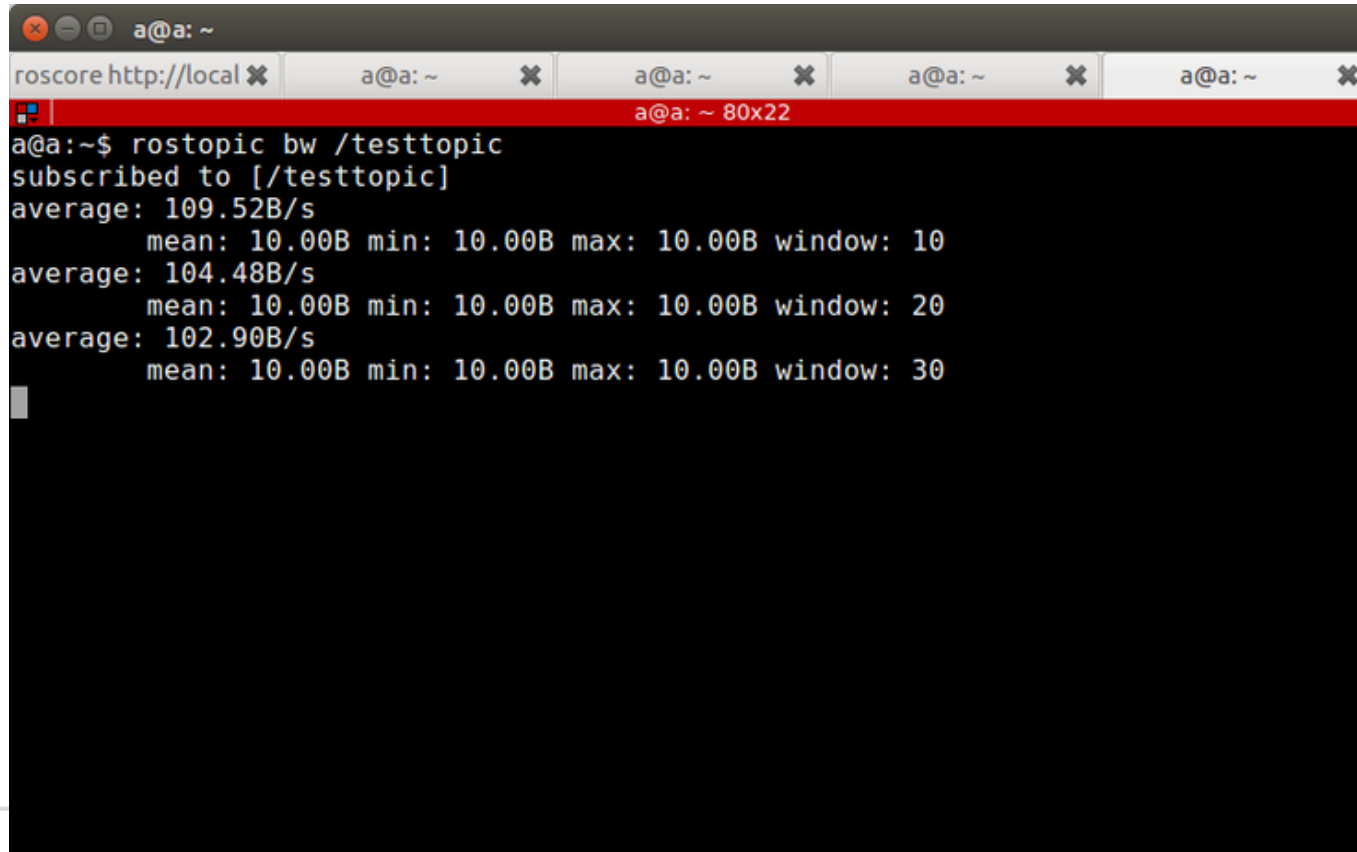
[illegible]

# ROS Topic

- Rostopic command Line

- rostopic bw [topic name] : topic의 bandwidth를 모니터링.

\$ rostopic bw /testtopic

A terminal window with a dark background and a light gray title bar. The title bar contains the text 'a@a: ~' and several window control icons. Below the title bar, there are five tabs, each labeled 'a@a: ~'. The terminal content shows the command 'rostopic bw /testtopic' being executed, followed by the output 'subscribed to [/testtopic]'. Then, three lines of bandwidth data are displayed: 'average: 109.52B/s' with 'mean: 10.00B min: 10.00B max: 10.00B window: 10', 'average: 104.48B/s' with 'mean: 10.00B min: 10.00B max: 10.00B window: 20', and 'average: 102.90B/s' with 'mean: 10.00B min: 10.00B max: 10.00B window: 30'. A red status bar at the top of the terminal area shows 'a@a: ~ 80x22'.

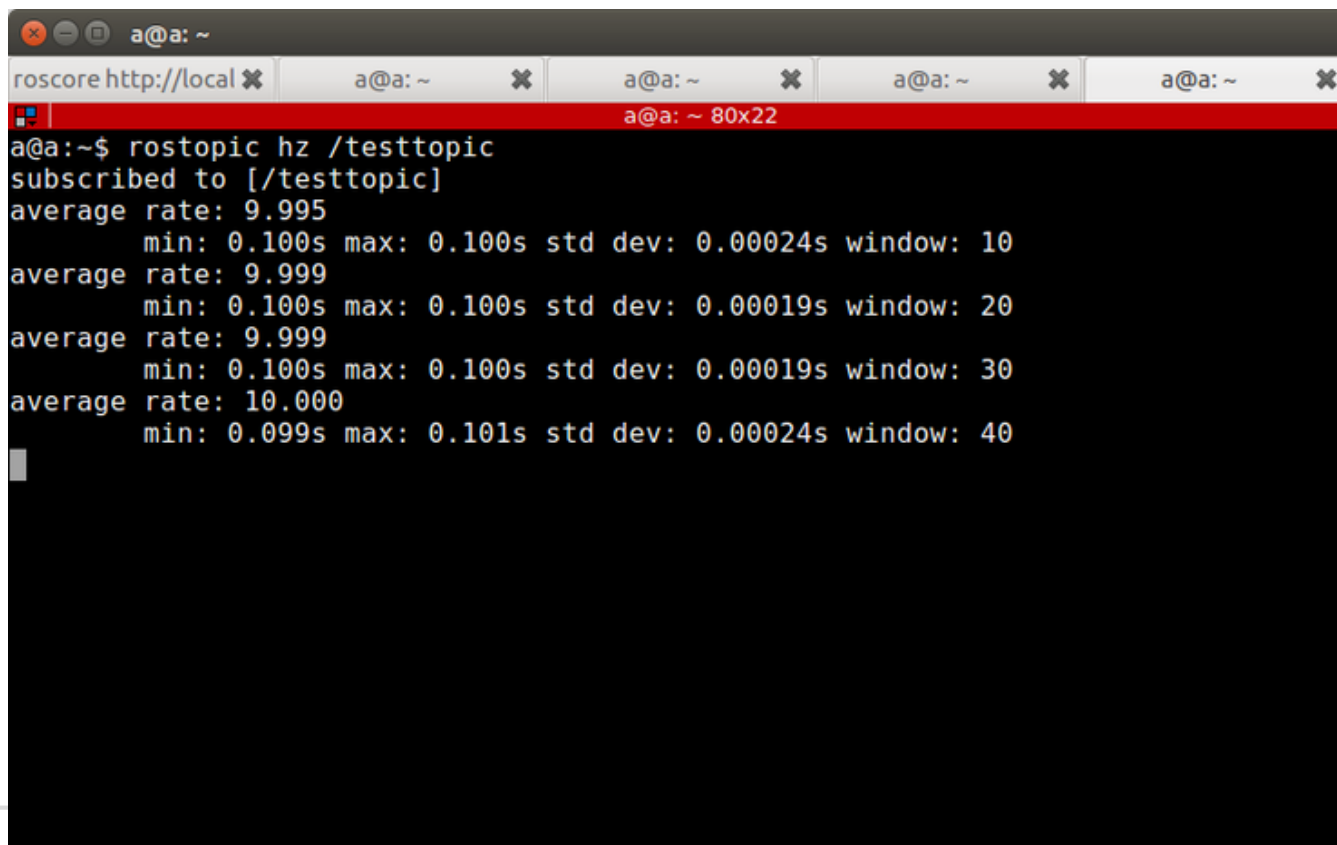
```
a@a: ~  
roscore http://local ✕ a@a: ~ ✕ a@a: ~ ✕ a@a: ~ ✕ a@a: ~ ✕  
a@a: ~ 80x22  
a@a:~$ rostopic bw /testtopic  
subscribed to [/testtopic]  
average: 109.52B/s  
    mean: 10.00B min: 10.00B max: 10.00B window: 10  
average: 104.48B/s  
    mean: 10.00B min: 10.00B max: 10.00B window: 20  
average: 102.90B/s  
    mean: 10.00B min: 10.00B max: 10.00B window: 30
```

# ROS Topic

- Rostopic command Line

- rostopic hz [topic name] : topic의 frequency를 모니터링.

rostopic hz /testtopic

A terminal window with a dark background and light text. The window title bar shows 'a@a: ~' and several open tabs. The terminal content shows the command 'rostopic hz /testtopic' being executed, followed by a series of output lines showing the average rate and min/max/std dev/window for the topic. The output is as follows:

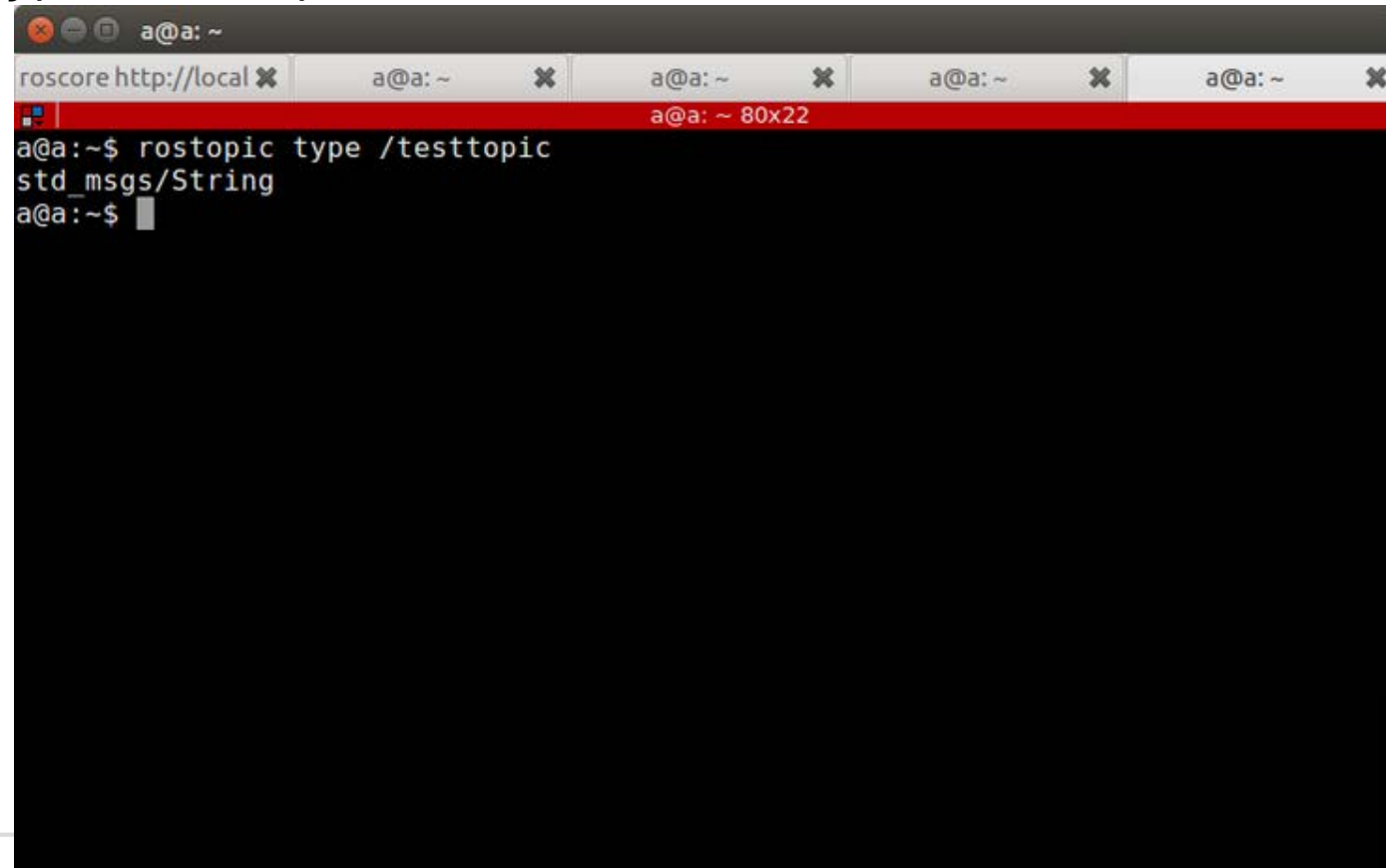
```
a@a:~$ rostopic hz /testtopic
subscribed to [/testtopic]
average rate: 9.995
  min: 0.100s max: 0.100s std dev: 0.00024s window: 10
average rate: 9.999
  min: 0.100s max: 0.100s std dev: 0.00019s window: 20
average rate: 9.999
  min: 0.100s max: 0.100s std dev: 0.00019s window: 30
average rate: 10.000
  min: 0.099s max: 0.101s std dev: 0.00024s window: 40
```

# ROS Topic

---

- Rostopic command Line
  - rostopic type [topic name] : topic의 type을 보여줌.

rostopic type /testtopic



```
a@a: ~  
roscore http://local ✕ a@a: ~ ✕ a@a: ~ ✕ a@a: ~ ✕ a@a: ~ ✕  
a@a: ~ 80x22  
a@a:~$ rostopic type /testtopic  
std_msgs/String  
a@a:~$
```

# ROS Topic

- Rqt graph

- Rqt : node, topic, tf tree 등을 visualize하기 위한 ROS GUI Tool.
- Node와 topic을 그래프로 시각화

```
$ rosrun rqt_graph rqt_graph
```

새로고침

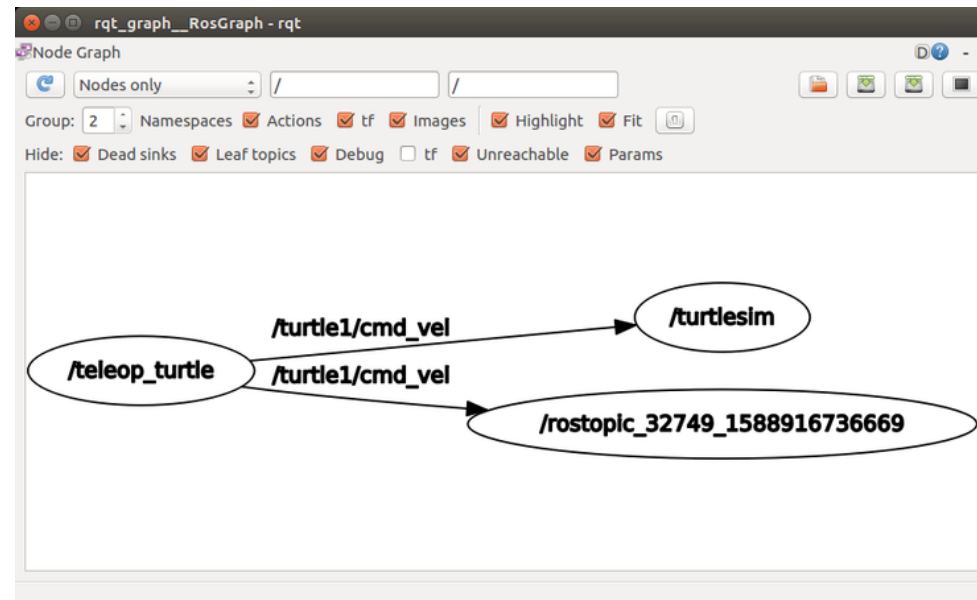


# ROS Topic

- Rqt graph

- Rqt : node, topic, tf tree 등을 visualize하기 위한 ROS GUI Tool.
  - Topic echo 를 실행한 뒤 그래프 확인

```
$ rostopic echo /turtle1/cmd_vel
```



→ 여러 노드에서 같은 Topic subscribe 가능!

# ROS Topic

- Talker / listener cpp

- Node에서 publish / subscribe하기.
- 앞서 만든 testpkg/src에 talker.cpp, listener.cpp 작성
- CMakeList.txt에 add\_executable, target\_link\_libraries 작성

\*\* Publisher 선언 구조

```
ros::Publisher pub이름 = n.advertise<토픽타입>("토픽명", 버퍼사이즈);
```

```
//talker.cpp
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>
int main(int argc, char **argv)
{
    ros::init(argc, argv, "talker");
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

    ros::Rate loop_rate(10); //hz
    int count = 0;
    while (ros::ok())
    {
        std_msgs::String msg;

        std::stringstream ss;
        ss << "hello world " << count;
        msg.data = ss.str();

        ROS_INFO("%s", msg.data.c_str());

        chatter_pub.publish(msg);

        ros::spinOnce();

        loop_rate.sleep();
        ++count;
    }
    return 0;
}
```



# ROS Topic

- Talker / listener cpp

- Node에서 publish / subscribe하기.
- 앞서 만든 testpkg/src에 talker.cpp, listener.cpp 작성
- CMakeList.txt에 add\_executable, target\_link\_libraries 작성

\*\* Publisher 선언 구조

```
ros::Publisher pub이름 = n.advertise<토픽타입>("토픽명", 버퍼사이즈);
```

```
## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test
test/test_testpkg.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test
${PROJECT_NAME})
# endif()
```

```
## Add folders to be run by python nosetests
# catkin_add_nosetests(test)
```

```
add_executable(talkernode src/talker.cpp)
target_link_libraries(talkernode ${catkin_LIBRARIES})
```

```
add_executable(listenernode src/listener.cpp)
target_link_libraries(listenernode ${catkin_LIBRARIES})
```

# ROS Topic

- Talker / listener cpp
  - Catkin\_make 빌드 후 실행

```
//listener.cpp
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
    ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

    ros::spin();

    return 0;
}
```

**\*\* Subscriber 선언 구조**

```
ros::Subscriber sub이름 = n.subscribe("토픽명", 버퍼  
퍼사이즈, 콜백함수명);
```

```
void 콜백함수명 (토픽 or 토픽포인터)
```

```
std_msgs::String::ConstPtr& msg
```

```
std_msgs::String msg
```

# ROS Topic

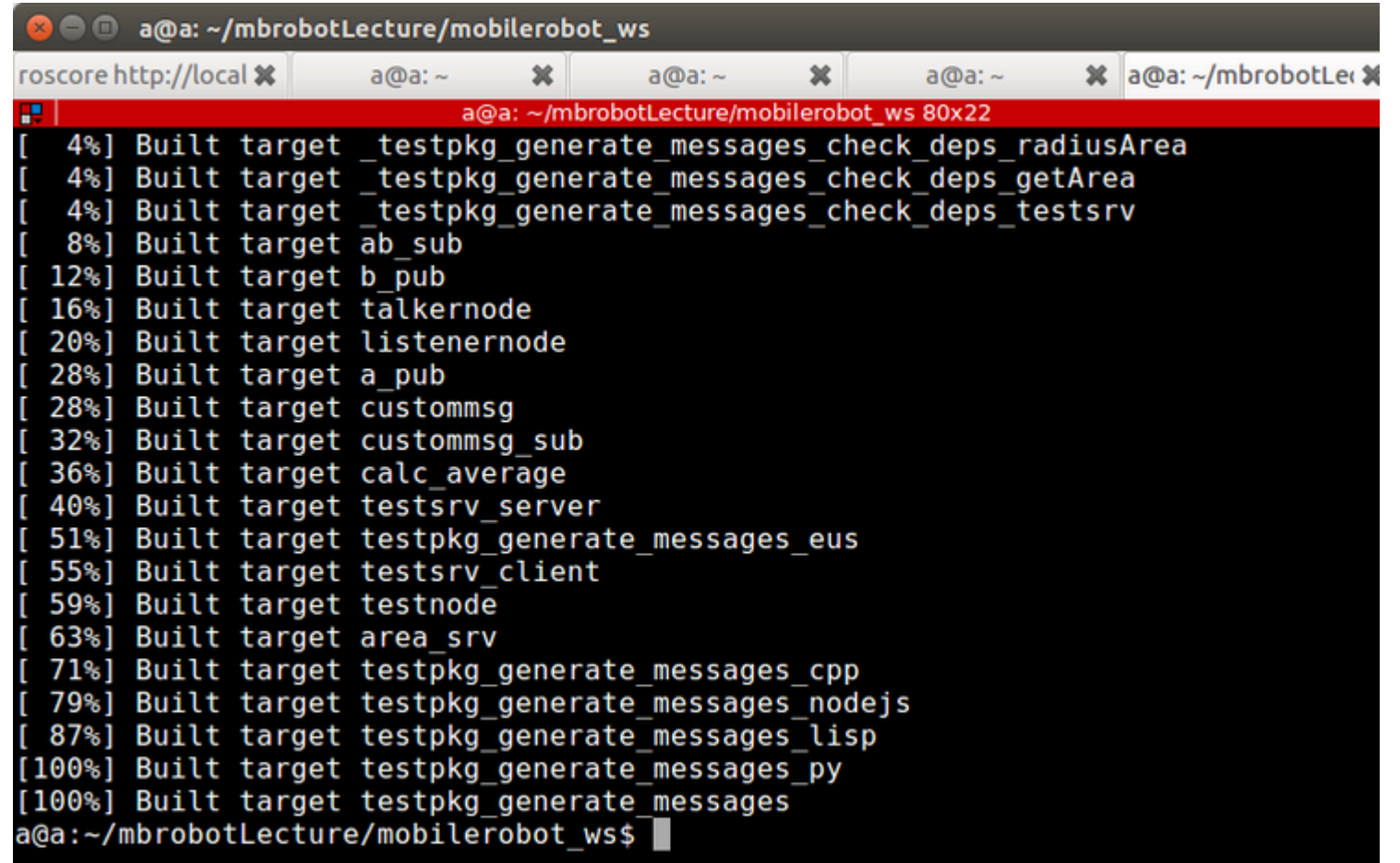
- Talker / listener cpp
  - Catkin\_make 빌드 후 실행

# Terminal 1

**catkin\_make**

source devel/setup.sh

roslaunch testpkg talkernode



The image shows a terminal window titled 'a@a: ~/mbrobotLecture/mobilerobot\_ws'. The window contains the output of a 'catkin\_make' command, showing the progress of building various targets. The progress is displayed as a percentage in brackets followed by the target name. The targets are: \_testpkg\_generate\_messages\_check\_deps\_radiusArea (4%), \_testpkg\_generate\_messages\_check\_deps\_getArea (4%), \_testpkg\_generate\_messages\_check\_deps\_testsrv (4%), ab\_sub (8%), b\_pub (12%), talkernode (16%), listenernode (20%), a\_pub (28%), custommsg (28%), custommsg\_sub (32%), calc\_average (36%), testsrv\_server (40%), testpkg\_generate\_messages\_eus (51%), testsrv\_client (55%), testnode (59%), area\_srv (63%), testpkg\_generate\_messages\_cpp (71%), testpkg\_generate\_messages\_nodejs (79%), testpkg\_generate\_messages\_lisp (87%), testpkg\_generate\_messages\_py (100%), and testpkg\_generate\_messages (100%). The terminal ends with the prompt 'a@a:~/mbrobotLecture/mobilerobot\_ws\$'.

```
a@a: ~/mbrobotLecture/mobilerobot_ws
[ 4%] Built target _testpkg_generate_messages_check_deps_radiusArea
[ 4%] Built target _testpkg_generate_messages_check_deps_getArea
[ 4%] Built target _testpkg_generate_messages_check_deps_testsrv
[ 8%] Built target ab_sub
[ 12%] Built target b_pub
[ 16%] Built target talkernode
[ 20%] Built target listenernode
[ 28%] Built target a_pub
[ 28%] Built target custommsg
[ 32%] Built target custommsg_sub
[ 36%] Built target calc_average
[ 40%] Built target testsrv_server
[ 51%] Built target testpkg_generate_messages_eus
[ 55%] Built target testsrv_client
[ 59%] Built target testnode
[ 63%] Built target area_srv
[ 71%] Built target testpkg_generate_messages_cpp
[ 79%] Built target testpkg_generate_messages_nodejs
[ 87%] Built target testpkg_generate_messages_lisp
[100%] Built target testpkg_generate_messages_py
[100%] Built target testpkg_generate_messages
a@a:~/mbrobotLecture/mobilerobot_ws$
```

# ROS Topic

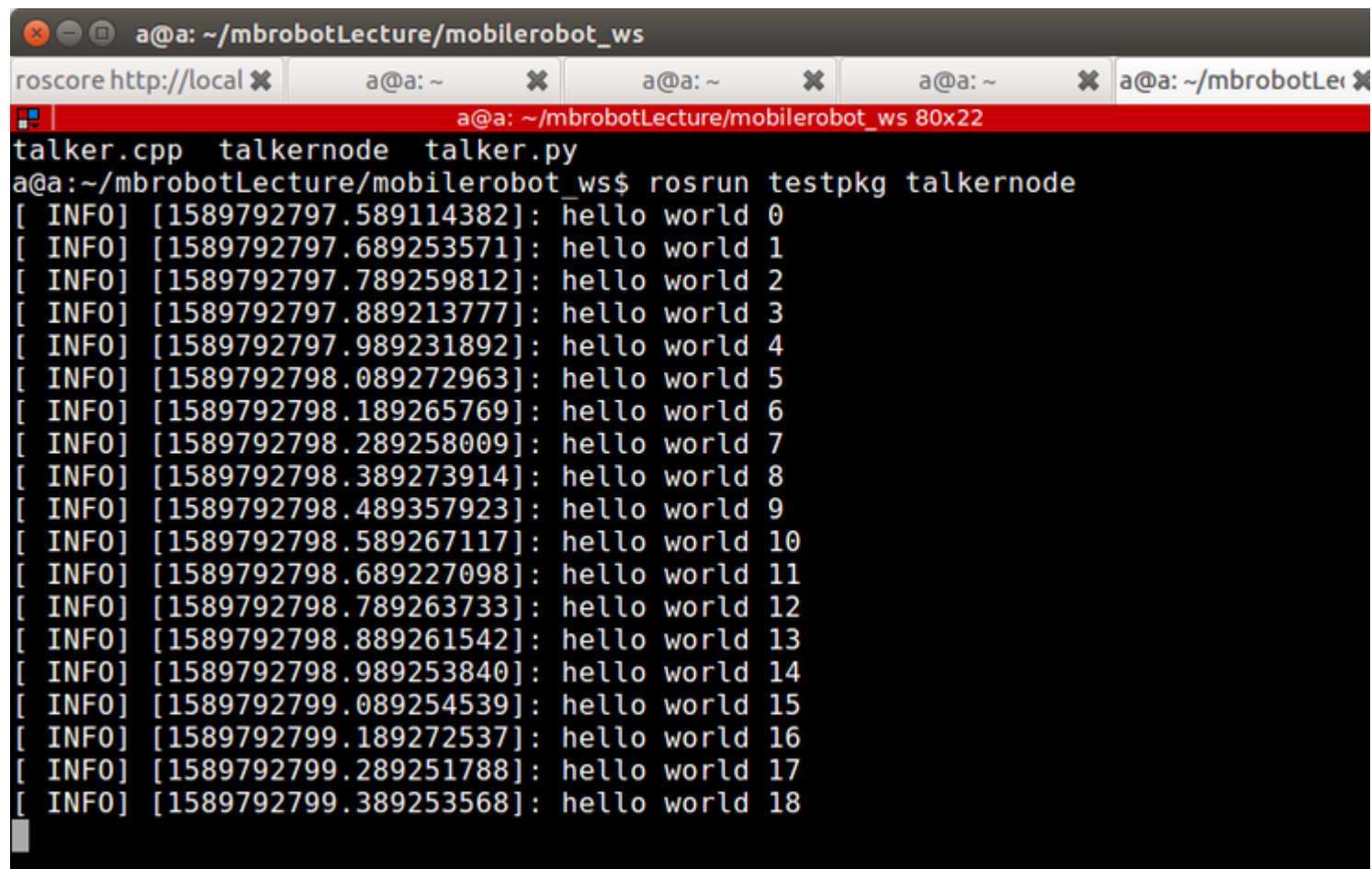
- Talker / listener cpp
  - Catkin\_make 빌드 후 실행

# Terminal 1

catkin\_make

source devel/setup.sh

roslaunch testpkg talker

A terminal window titled 'a@a: ~/mbrobotLecture/mobilerobot\_ws' with multiple tabs. The active tab shows the command 'roslaunch testpkg talker' being executed. The output displays a series of 'hello world' messages from the 'talker' node, each preceded by an INFO log level and a timestamp. The messages are numbered from 0 to 18. The terminal window has a red title bar and standard Linux window controls.

```
a@a: ~/mbrobotLecture/mobilerobot_ws
roslaunch testpkg talker
[ INFO] [1589792797.589114382]: hello world 0
[ INFO] [1589792797.689253571]: hello world 1
[ INFO] [1589792797.789259812]: hello world 2
[ INFO] [1589792797.889213777]: hello world 3
[ INFO] [1589792797.989231892]: hello world 4
[ INFO] [1589792798.089272963]: hello world 5
[ INFO] [1589792798.189265769]: hello world 6
[ INFO] [1589792798.289258009]: hello world 7
[ INFO] [1589792798.389273914]: hello world 8
[ INFO] [1589792798.489357923]: hello world 9
[ INFO] [1589792798.589267117]: hello world 10
[ INFO] [1589792798.689227098]: hello world 11
[ INFO] [1589792798.789263733]: hello world 12
[ INFO] [1589792798.889261542]: hello world 13
[ INFO] [1589792798.989253840]: hello world 14
[ INFO] [1589792799.089254539]: hello world 15
[ INFO] [1589792799.189272537]: hello world 16
[ INFO] [1589792799.289251788]: hello world 17
[ INFO] [1589792799.389253568]: hello world 18
```

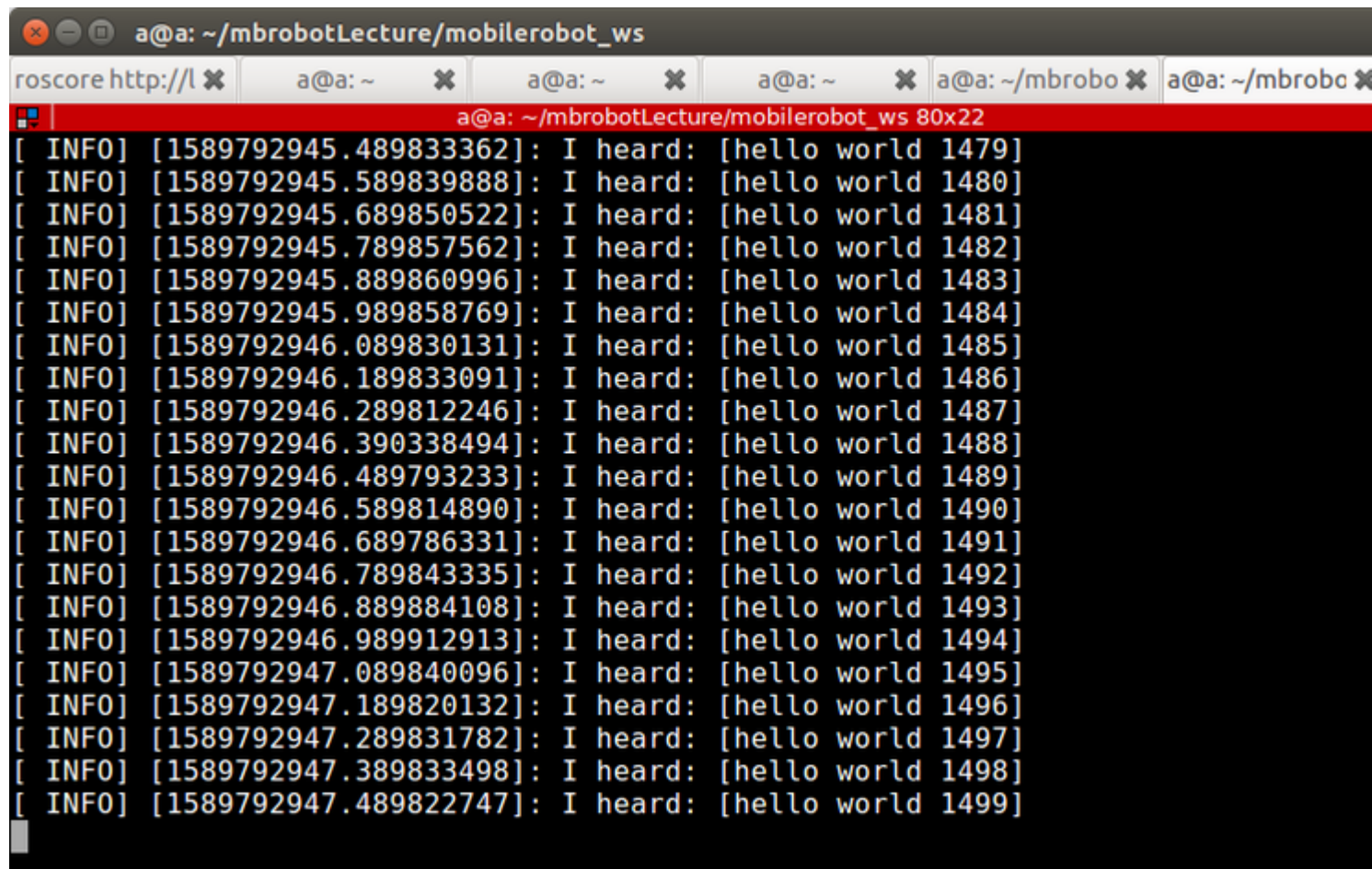
# ROS Topic

- Talker / listener cpp
  - Catkin\_make 빌드 후 실행

# Terminal 2

`source devel/setup.sh`

`roslaunch testpkg listener_node`



```
a@a: ~/mbrobotLecture/mobilerobot_ws
roslaunch testpkg listener_node
[ INFO] [1589792945.489833362]: I heard: [hello world 1479]
[ INFO] [1589792945.589839888]: I heard: [hello world 1480]
[ INFO] [1589792945.689850522]: I heard: [hello world 1481]
[ INFO] [1589792945.789857562]: I heard: [hello world 1482]
[ INFO] [1589792945.889860996]: I heard: [hello world 1483]
[ INFO] [1589792945.989858769]: I heard: [hello world 1484]
[ INFO] [1589792946.089830131]: I heard: [hello world 1485]
[ INFO] [1589792946.189833091]: I heard: [hello world 1486]
[ INFO] [1589792946.289812246]: I heard: [hello world 1487]
[ INFO] [1589792946.390338494]: I heard: [hello world 1488]
[ INFO] [1589792946.489793233]: I heard: [hello world 1489]
[ INFO] [1589792946.589814890]: I heard: [hello world 1490]
[ INFO] [1589792946.689786331]: I heard: [hello world 1491]
[ INFO] [1589792946.789843335]: I heard: [hello world 1492]
[ INFO] [1589792946.889884108]: I heard: [hello world 1493]
[ INFO] [1589792946.989912913]: I heard: [hello world 1494]
[ INFO] [1589792947.089840096]: I heard: [hello world 1495]
[ INFO] [1589792947.189820132]: I heard: [hello world 1496]
[ INFO] [1589792947.289831782]: I heard: [hello world 1497]
[ INFO] [1589792947.389833498]: I heard: [hello world 1498]
[ INFO] [1589792947.489822747]: I heard: [hello world 1499]
```

# ROS Topic

---

- Talker / listener python

- Python은 build 및 CMakeList.txt의 수정이 필요 없음.
- 실행권한만 주면 바로 실행 가능
- Testpkg/src 에 talker.py listener.py를 추가한후 실행

```
#talker.py
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

# ROS Topic

---

- Talker / listener python

- 실행권한 변경

\$ chmod +x <path of pythonfile>

```
#listener.py
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

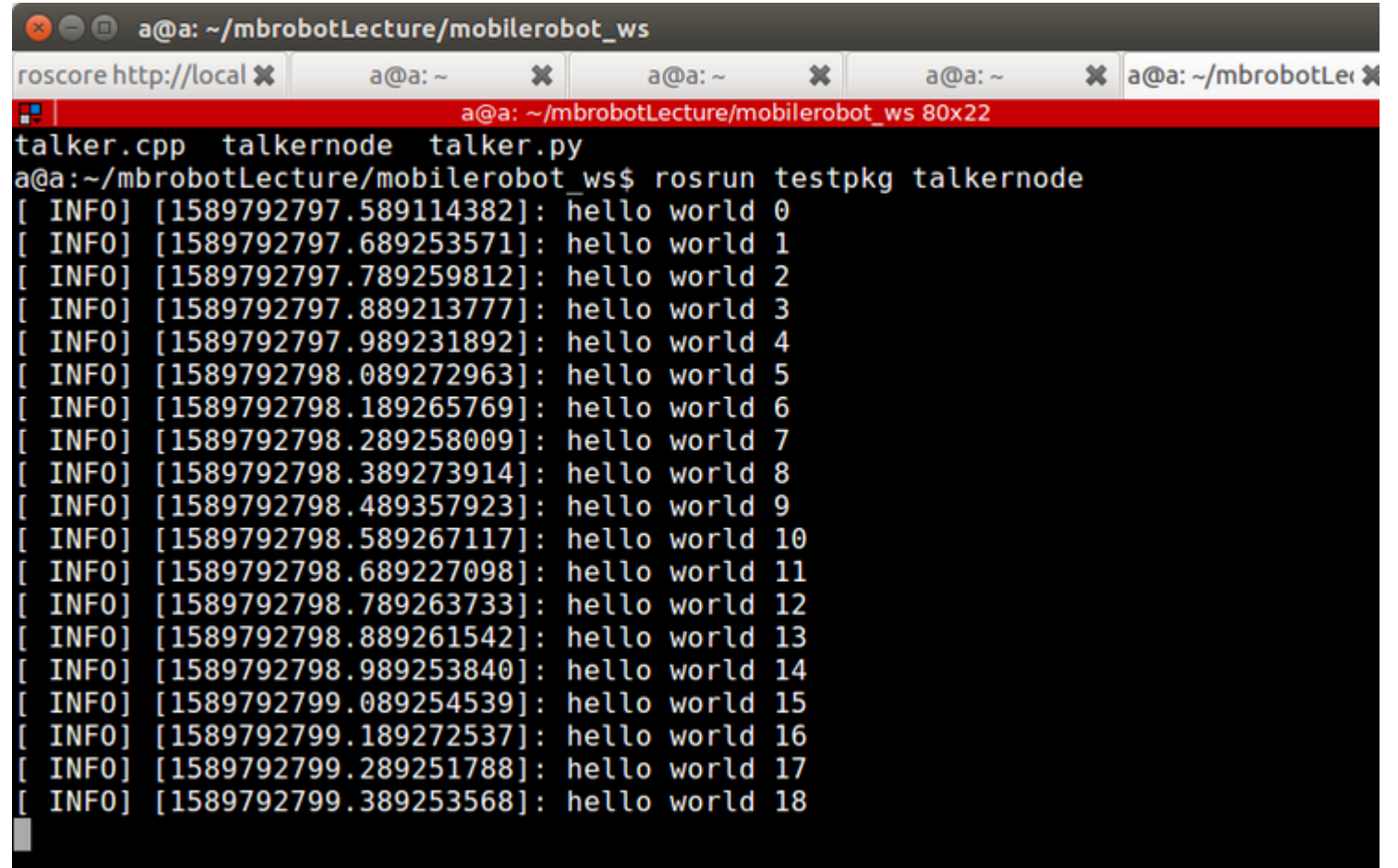


# ROS Topic

- Talker / listener python
  - Catkin\_make 빌드 없이 실행

# Terminal 1

```
source devel/setup.sh
chmod +x src/talker.py
roslaunch testpkg talker.py
```

A terminal window titled 'a@a: ~/mbrobotLecture/mobilerobot\_ws' with multiple tabs. The active tab shows the command 'roslaunch testpkg talker.py' and its output. The output consists of 19 lines of log messages, each starting with '[ INFO ]' followed by a timestamp and the text 'hello world' and a counter from 0 to 18. The terminal has a red title bar and a black background with white text.

```
a@a: ~/mbrobotLecture/mobilerobot_ws
roslaunch testpkg talker.py
[ INFO ] [1589792797.589114382]: hello world 0
[ INFO ] [1589792797.689253571]: hello world 1
[ INFO ] [1589792797.789259812]: hello world 2
[ INFO ] [1589792797.889213777]: hello world 3
[ INFO ] [1589792797.989231892]: hello world 4
[ INFO ] [1589792798.089272963]: hello world 5
[ INFO ] [1589792798.189265769]: hello world 6
[ INFO ] [1589792798.289258009]: hello world 7
[ INFO ] [1589792798.389273914]: hello world 8
[ INFO ] [1589792798.489357923]: hello world 9
[ INFO ] [1589792798.589267117]: hello world 10
[ INFO ] [1589792798.689227098]: hello world 11
[ INFO ] [1589792798.789263733]: hello world 12
[ INFO ] [1589792798.889261542]: hello world 13
[ INFO ] [1589792798.989253840]: hello world 14
[ INFO ] [1589792799.089254539]: hello world 15
[ INFO ] [1589792799.189272537]: hello world 16
[ INFO ] [1589792799.289251788]: hello world 17
[ INFO ] [1589792799.389253568]: hello world 18
```



# ROS Topic

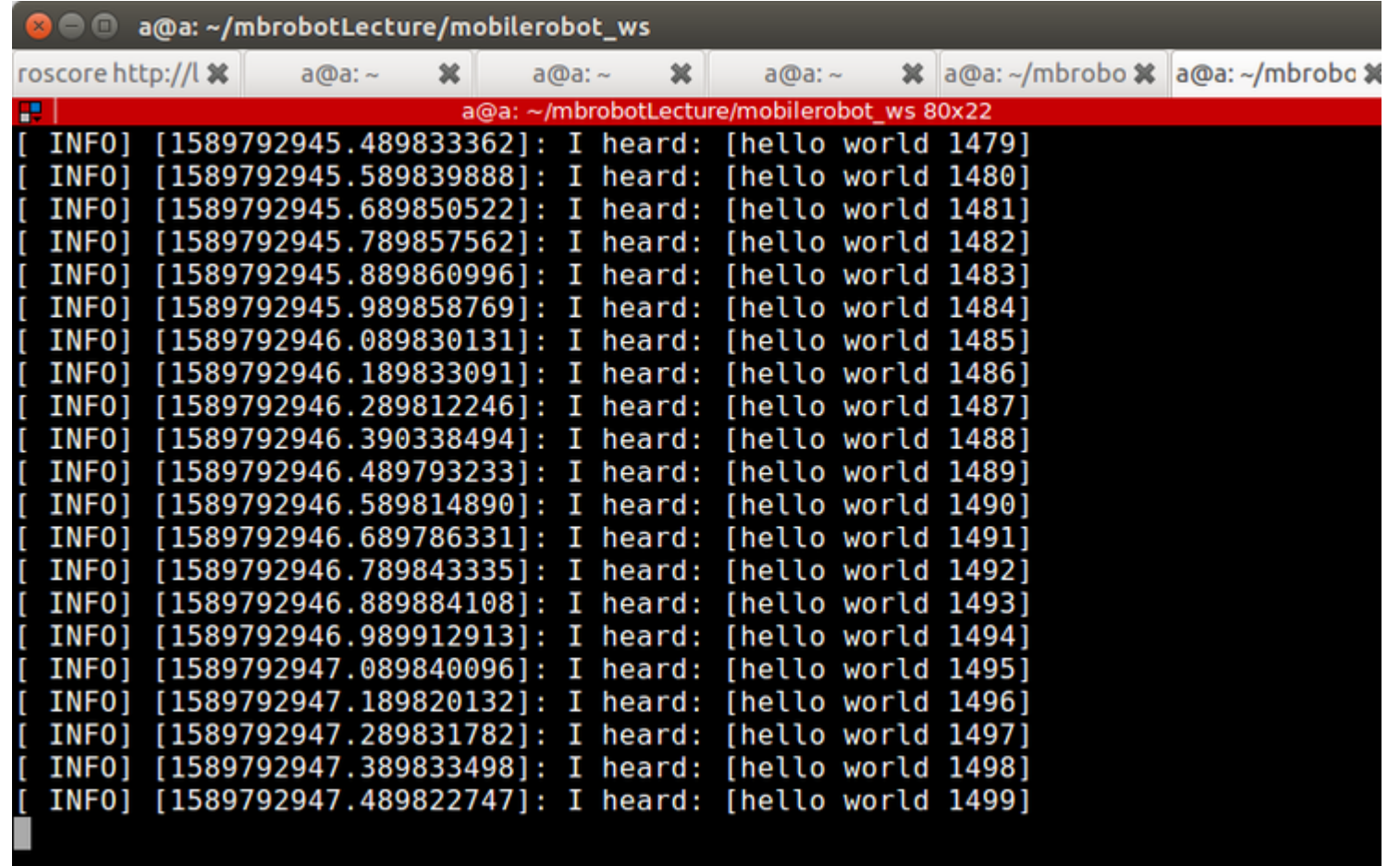
- Talker / listener python
  - Catkin\_make 빌드 없이 실행

# Terminal 2

```
source devel/setup.sh
```

```
chmod +x src/listener.py
```

```
roslaunch testpkg listener.py
```

A terminal window titled 'a@a: ~/mbrobotLecture/mobilerobot\_ws' with multiple tabs. The active tab shows a list of log messages from a ROS listener. Each line starts with '[ INFO ]', followed by a timestamp in brackets, a colon, and the message 'I heard: [hello world' followed by a sequence number in brackets. The sequence numbers range from 1479 to 1499.

```
a@a: ~/mbrobotLecture/mobilerobot_ws
roslaunch testpkg listener.py
[ INFO] [1589792945.489833362]: I heard: [hello world 1479]
[ INFO] [1589792945.589839888]: I heard: [hello world 1480]
[ INFO] [1589792945.689850522]: I heard: [hello world 1481]
[ INFO] [1589792945.789857562]: I heard: [hello world 1482]
[ INFO] [1589792945.889860996]: I heard: [hello world 1483]
[ INFO] [1589792945.989858769]: I heard: [hello world 1484]
[ INFO] [1589792946.089830131]: I heard: [hello world 1485]
[ INFO] [1589792946.189833091]: I heard: [hello world 1486]
[ INFO] [1589792946.289812246]: I heard: [hello world 1487]
[ INFO] [1589792946.390338494]: I heard: [hello world 1488]
[ INFO] [1589792946.489793233]: I heard: [hello world 1489]
[ INFO] [1589792946.589814890]: I heard: [hello world 1490]
[ INFO] [1589792946.689786331]: I heard: [hello world 1491]
[ INFO] [1589792946.789843335]: I heard: [hello world 1492]
[ INFO] [1589792946.889884108]: I heard: [hello world 1493]
[ INFO] [1589792946.989912913]: I heard: [hello world 1494]
[ INFO] [1589792947.089840096]: I heard: [hello world 1495]
[ INFO] [1589792947.189820132]: I heard: [hello world 1496]
[ INFO] [1589792947.289831782]: I heard: [hello world 1497]
[ INFO] [1589792947.389833498]: I heard: [hello world 1498]
[ INFO] [1589792947.489822747]: I heard: [hello world 1499]
```

# ROS Topic

- 실행 결과 확인 – rqt graph

\$ rqt\_graph

