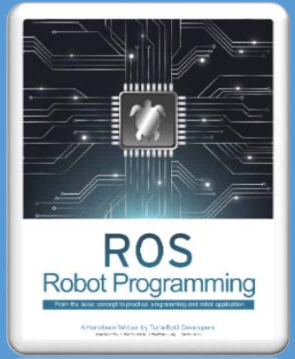
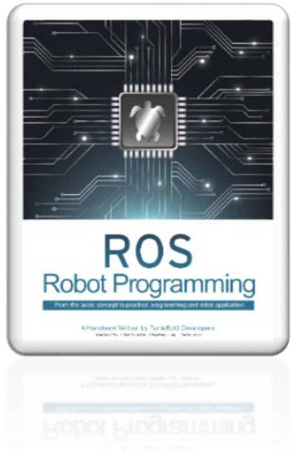
Important concepts of ROS



Subscribe

Textbook
P. 40~89

Contents



Textbook
P. 40~89

Robot operating system ROS term!

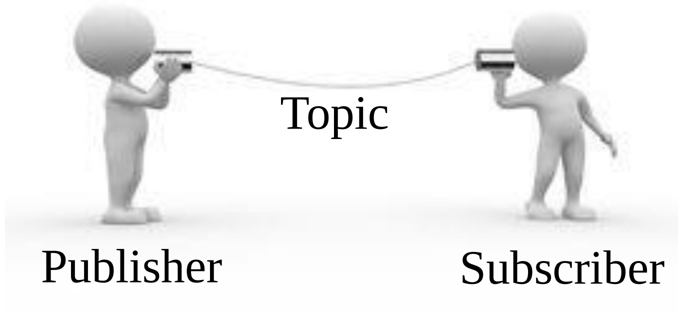
ROS terms

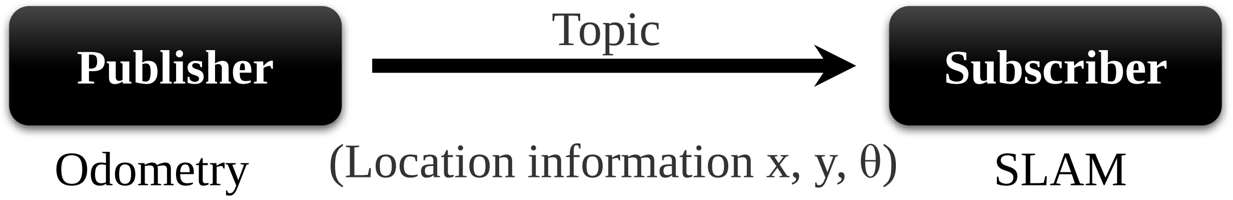
- Node
  - The smallest unit of executable processors. It can be regarded as single executable program. In ROS, a system is consist of many nodes. Each node transmits and receives data by message communication.

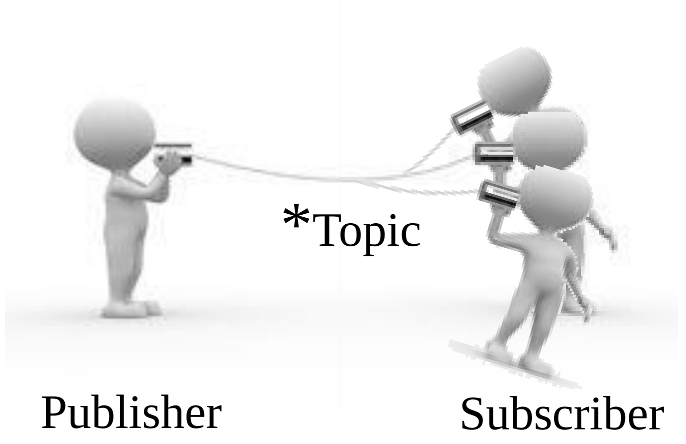
- Package
  - One or more nodes, information for node execution, etc. Also, bundles of packages are called as metapackages.

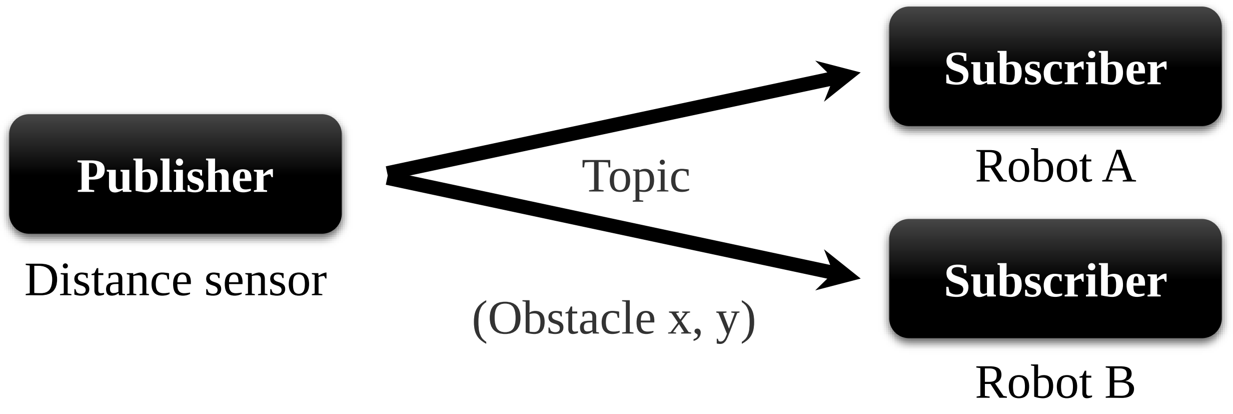
- Message
  - Data is transmitted and received through message between nodes. Messages can have various types such as integer, floating point, and boolean. You can also use structures such as a simple data structure and an array of messages that hold messages in the message.

ROS terms　Topic, Publisher, Subscriber





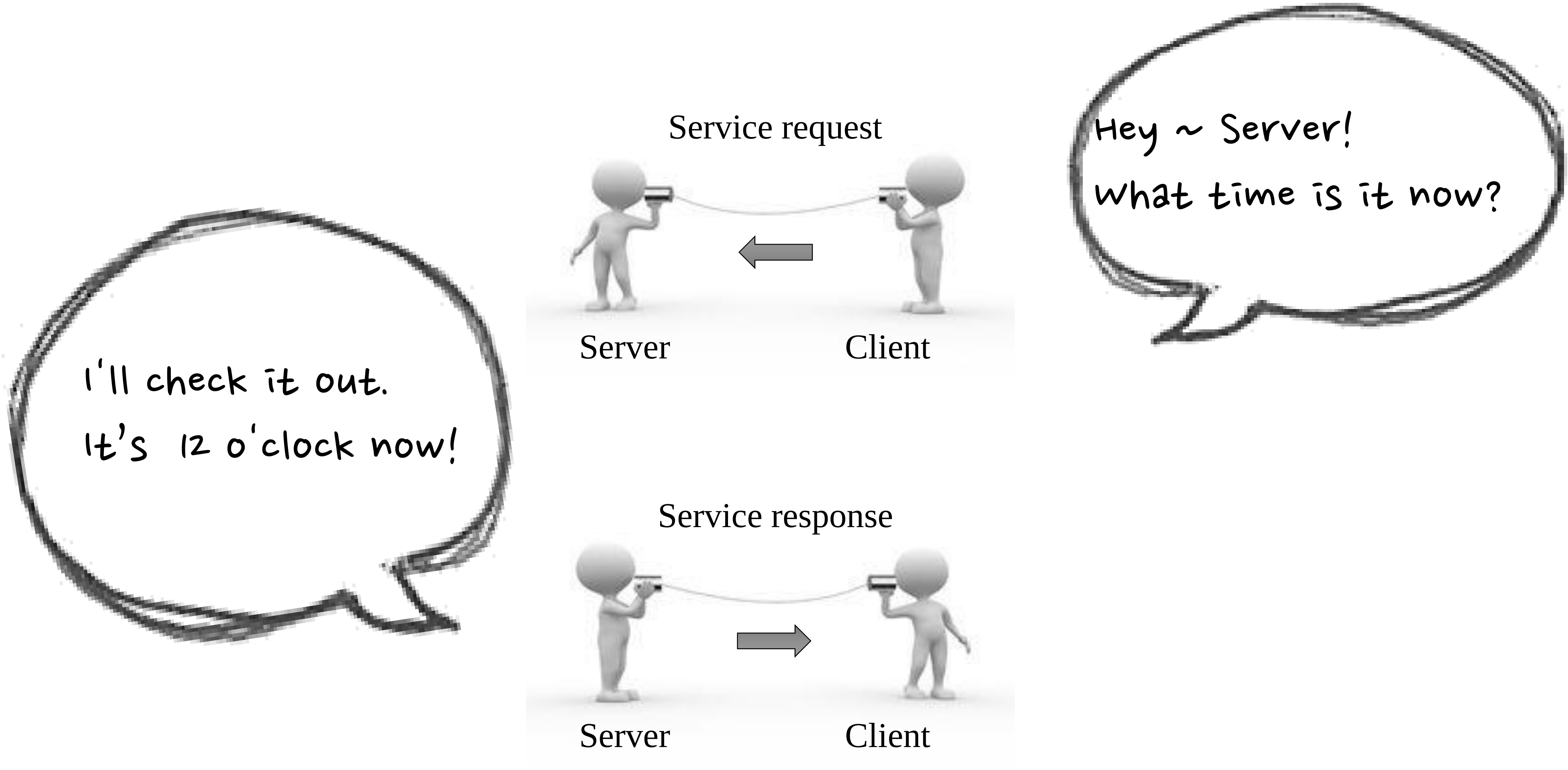




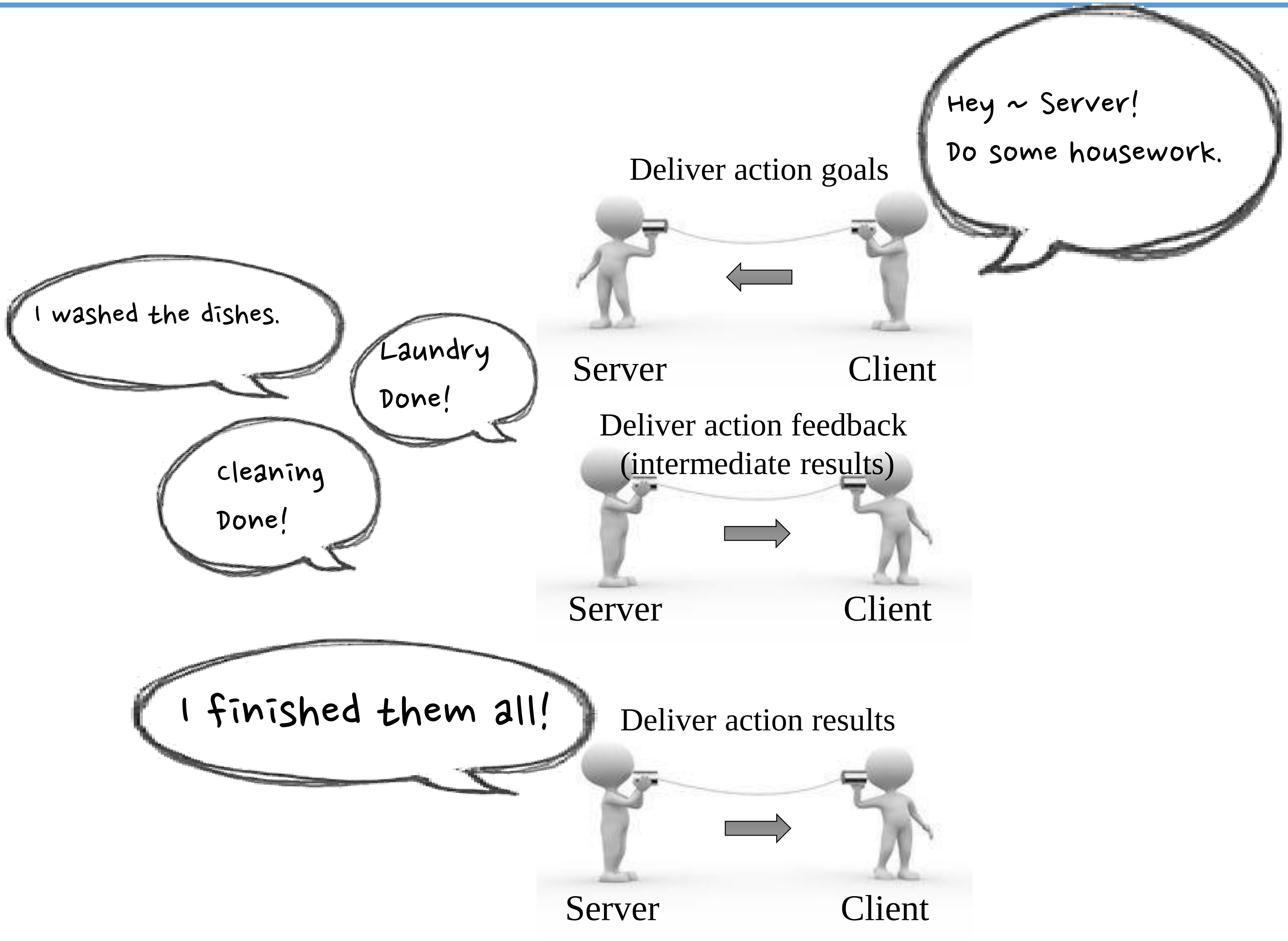
* 1: 1 Publisher and Subscriber communication is also possible for Topic,
and 1: N, N: 1, N: N communication is also possible depending on the purpose.

http://www.dreamstime.com/illustration/people-talk-listen-tin-can-phone-communication.html

ROS terms Service, Service server, Service client



ROS terms   Action, Action server, Action client



It's easy, isn't it?

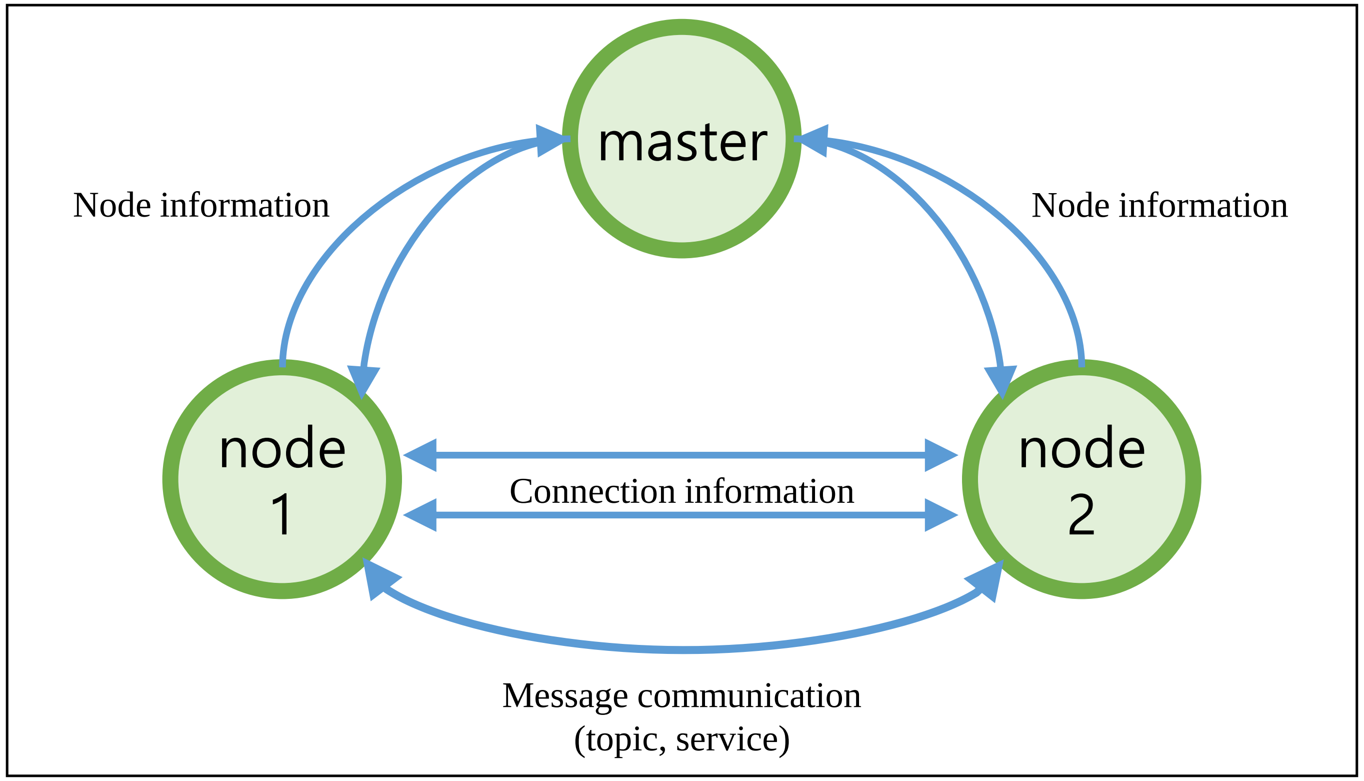
We will see in more detail~ ☺

Other terms will be explained at the time when necessary during the course!

Message communication

Understanding message communication

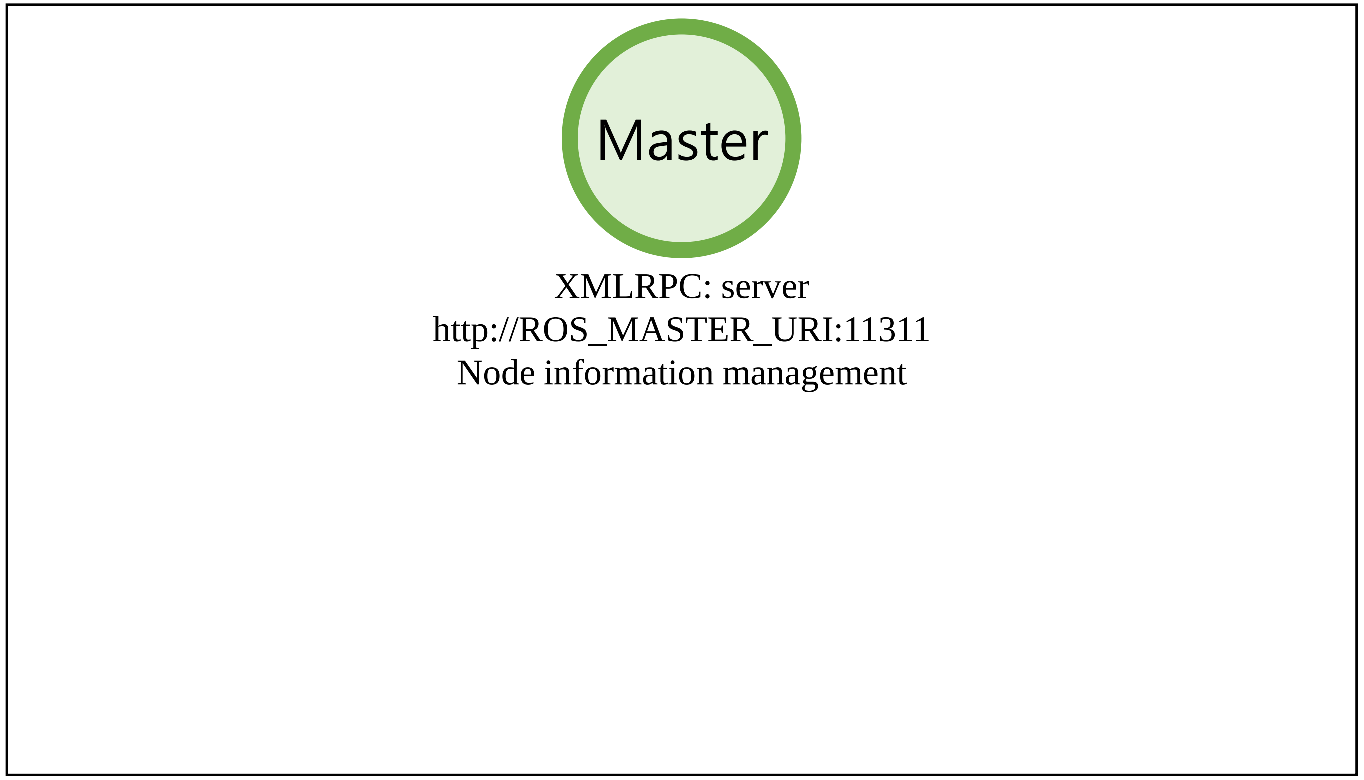
- The most fundamental technical point of ROS: **message communication among nodes!**



Understanding message communication

1. Run Master: XMLRPC(XML-Remote Procedure Call)

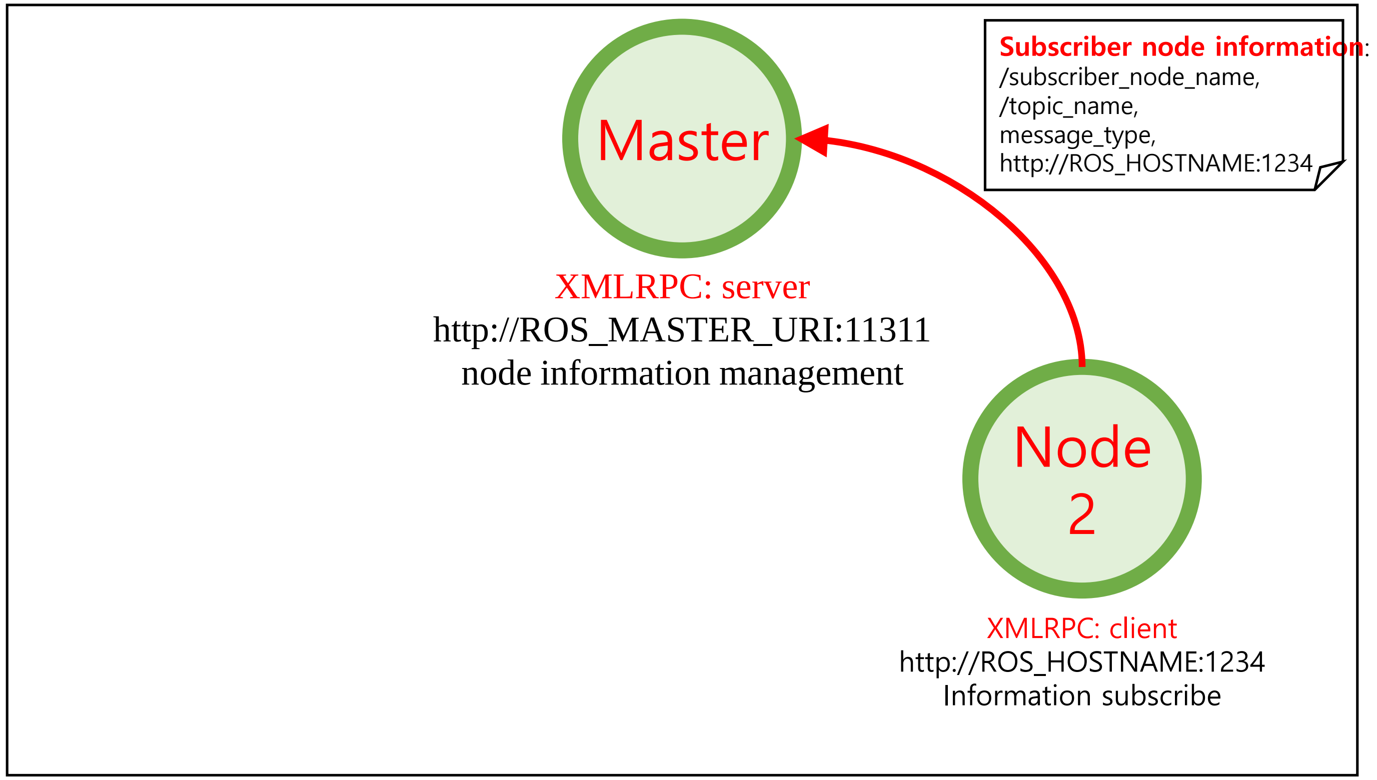
- $ roscore



Understanding message communication

2. Run Subscriber node

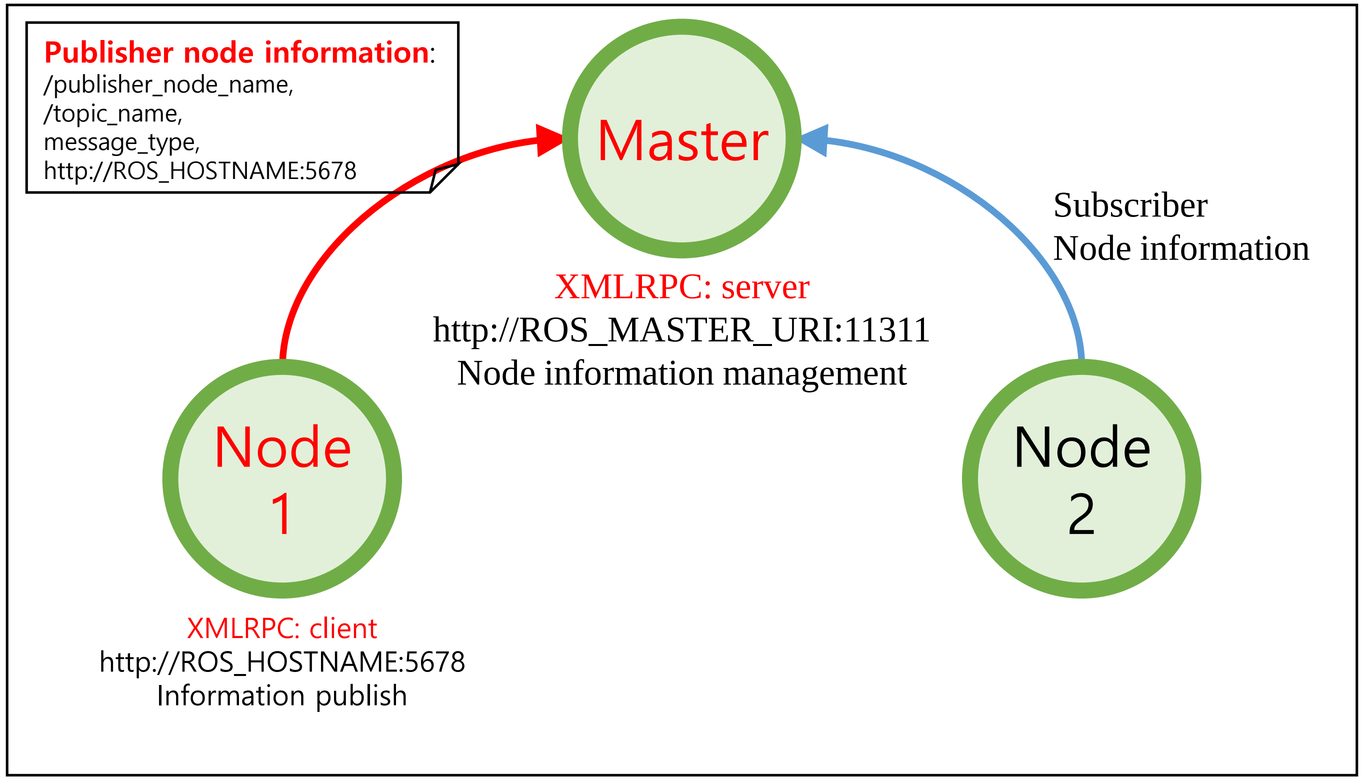
- $rosrun packagename nodename



Understanding message communication

3. Run Publisher node

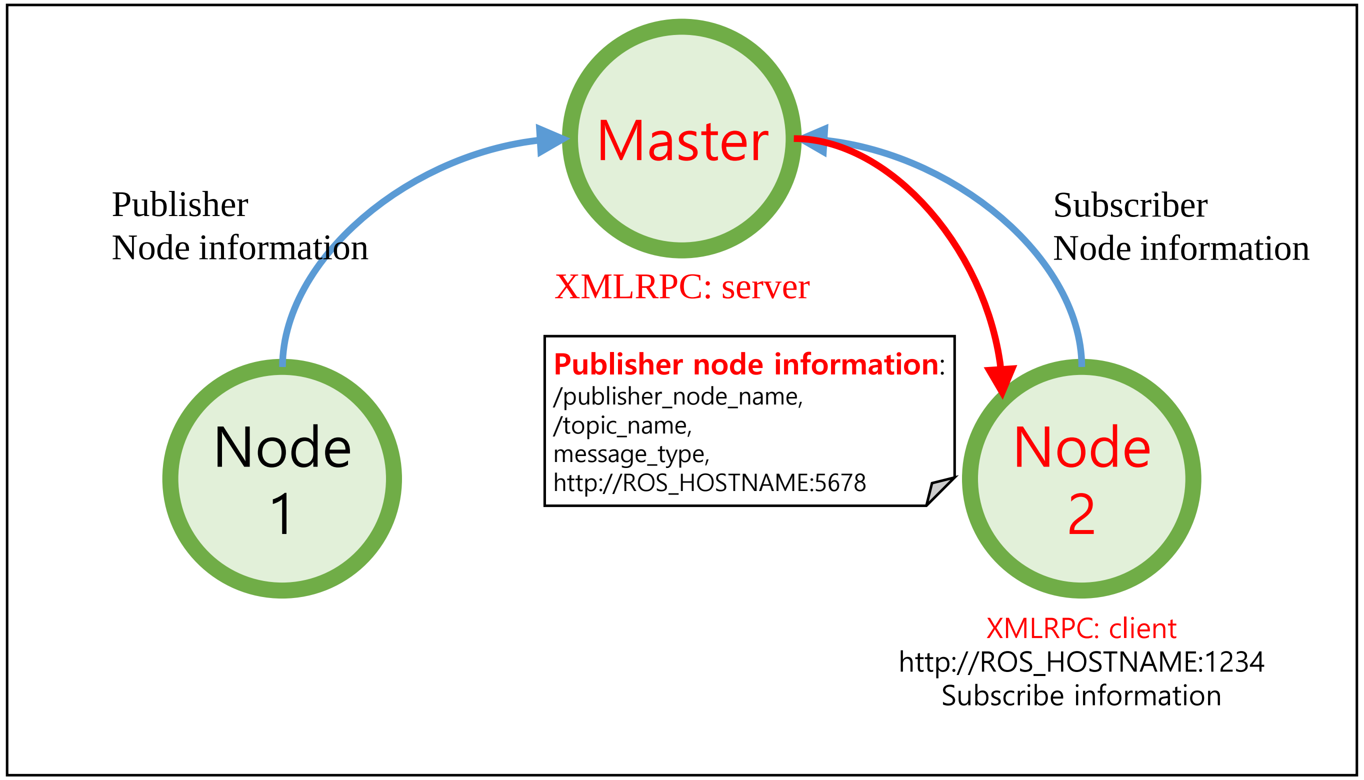
- $rosrun packagename nodename



Understanding message communication

4. Publisher Information

- The master informs the subscriber node of the new publisher information.

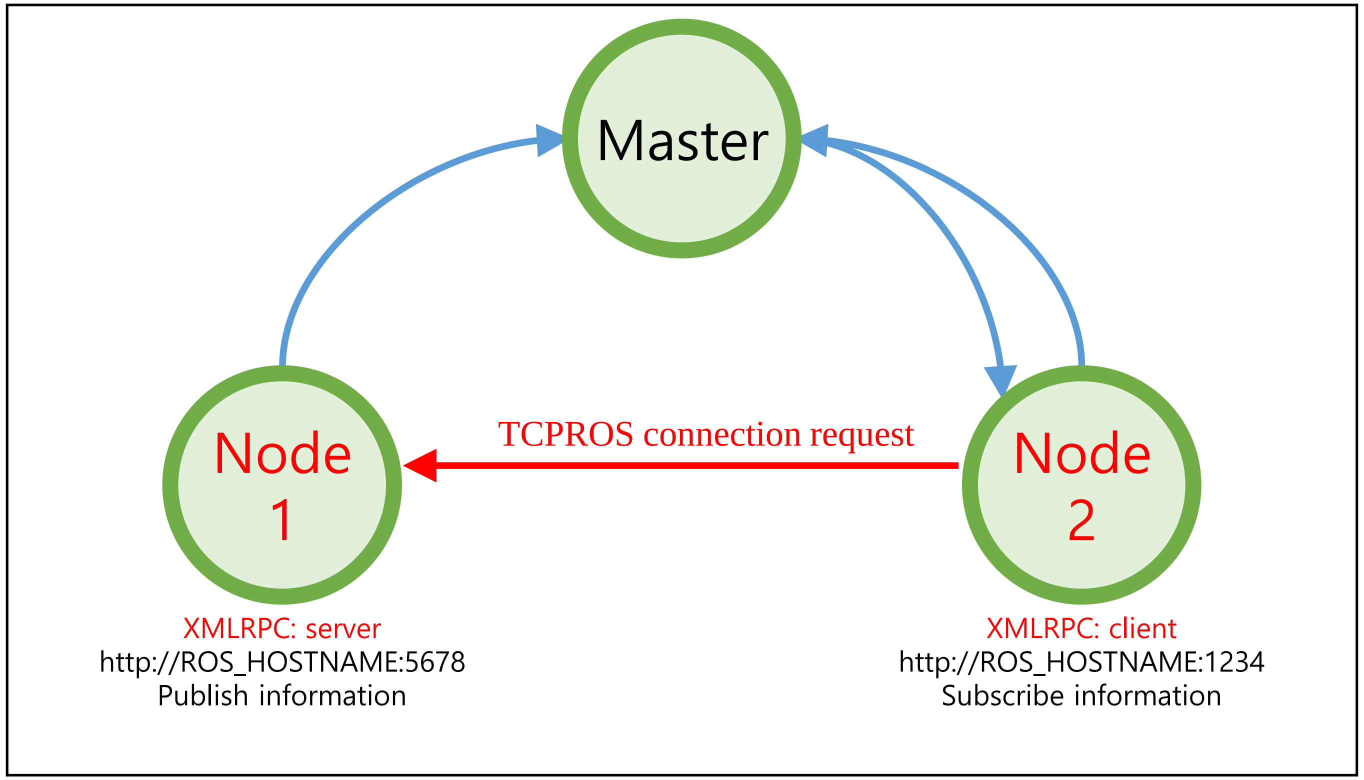


Publisher node information:
/publisher_node_name,
/topic_name,
message_type,
http://ROS_HOSTNAME:5678

Understanding message communication

5. Request access to the publisher node

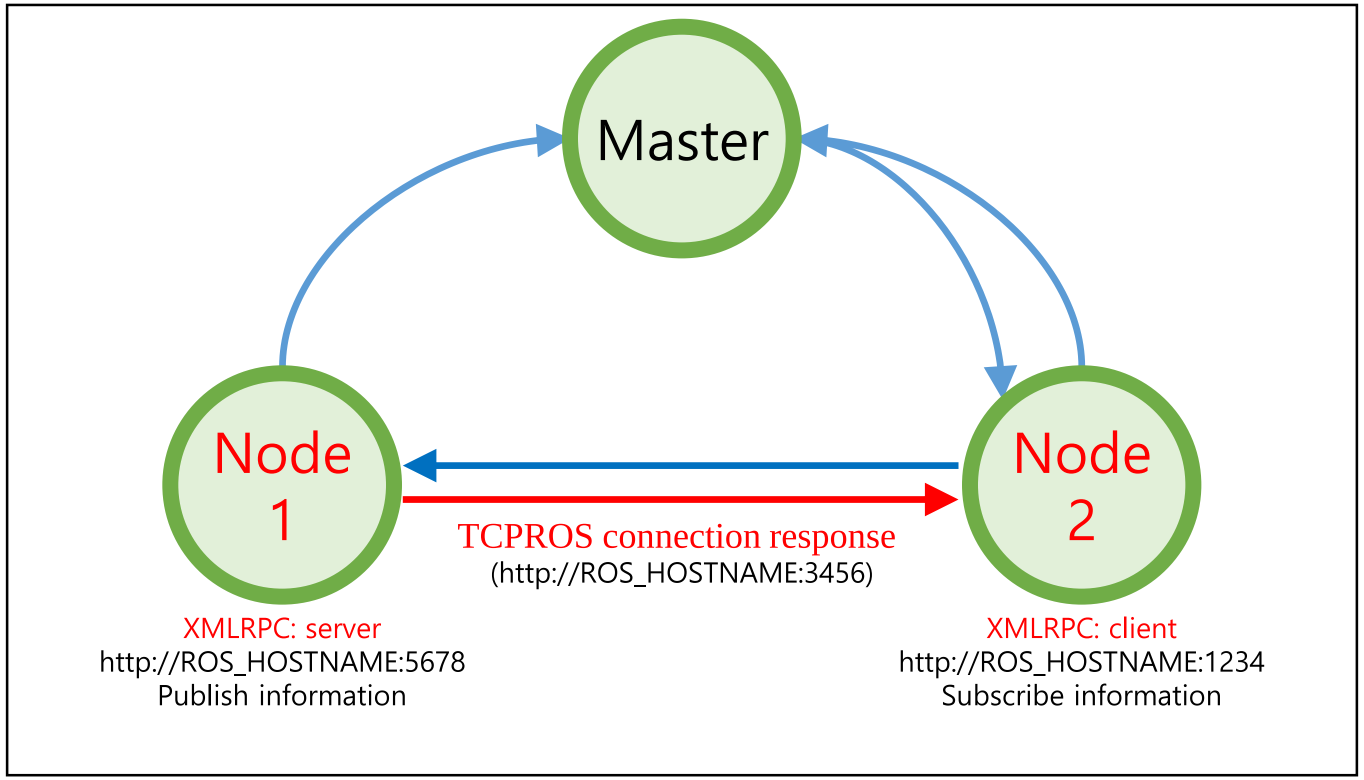
- Request TCPROS connection using the publisher information from the master



Understanding message communication

6. Connection response to subscriber node

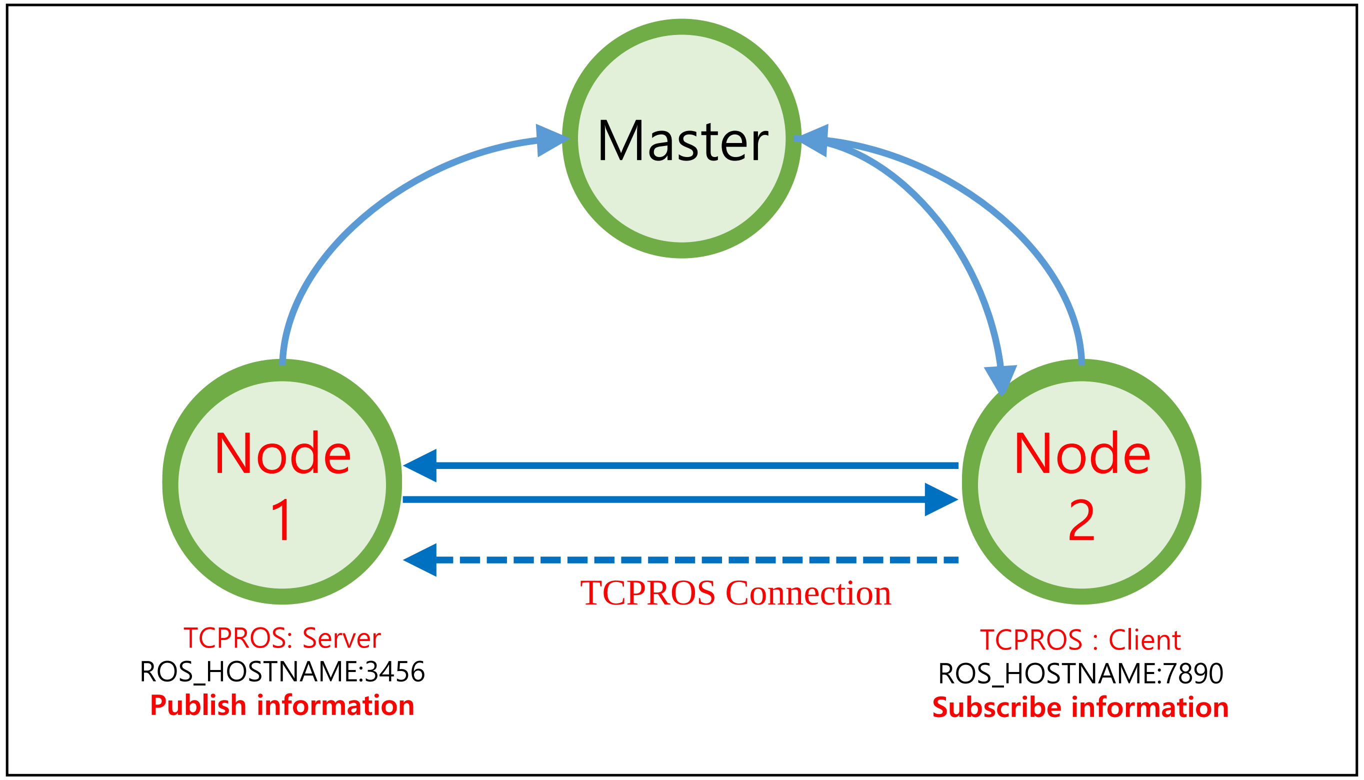
- Return TCP URI address and port number corresponding to the connection response



Understanding message communication

7. TCP Connection

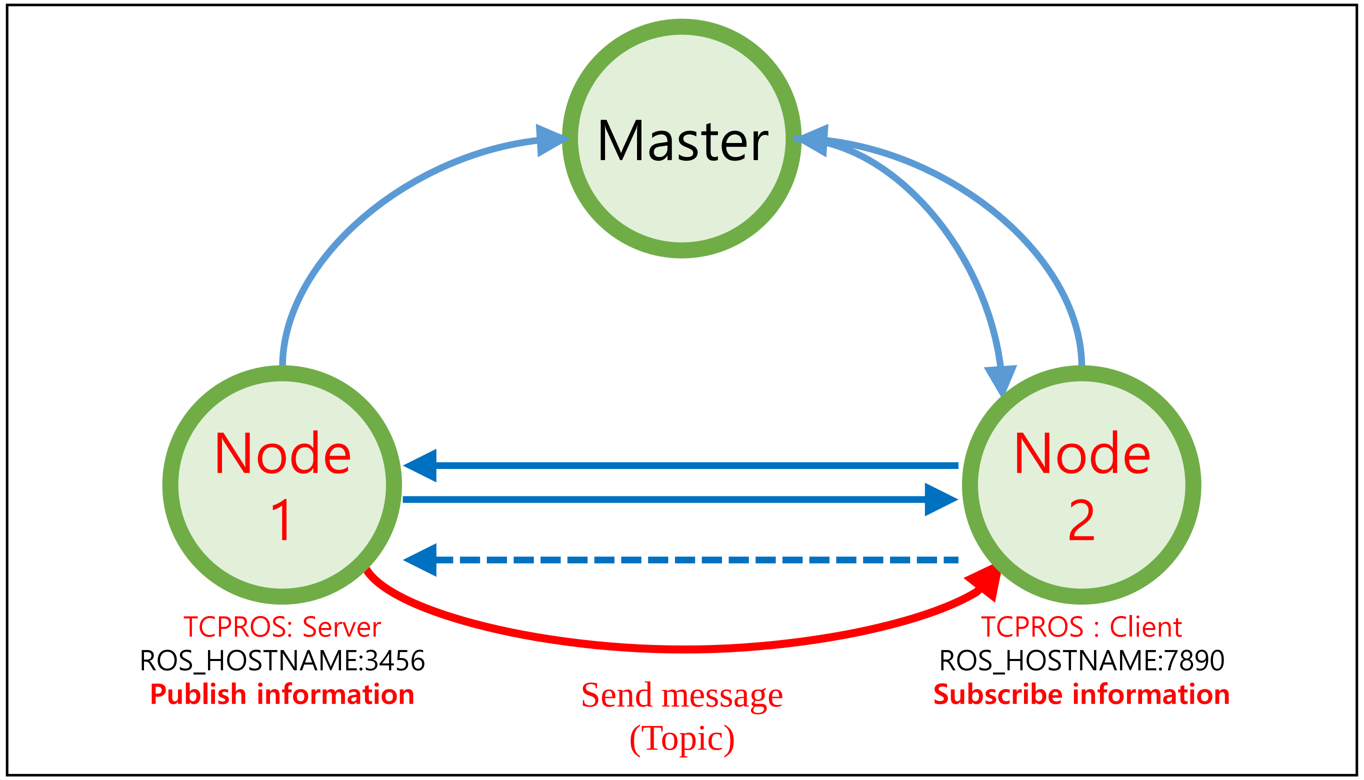
- Establish connection with the publisher node using TCPROS.



Understanding message communication

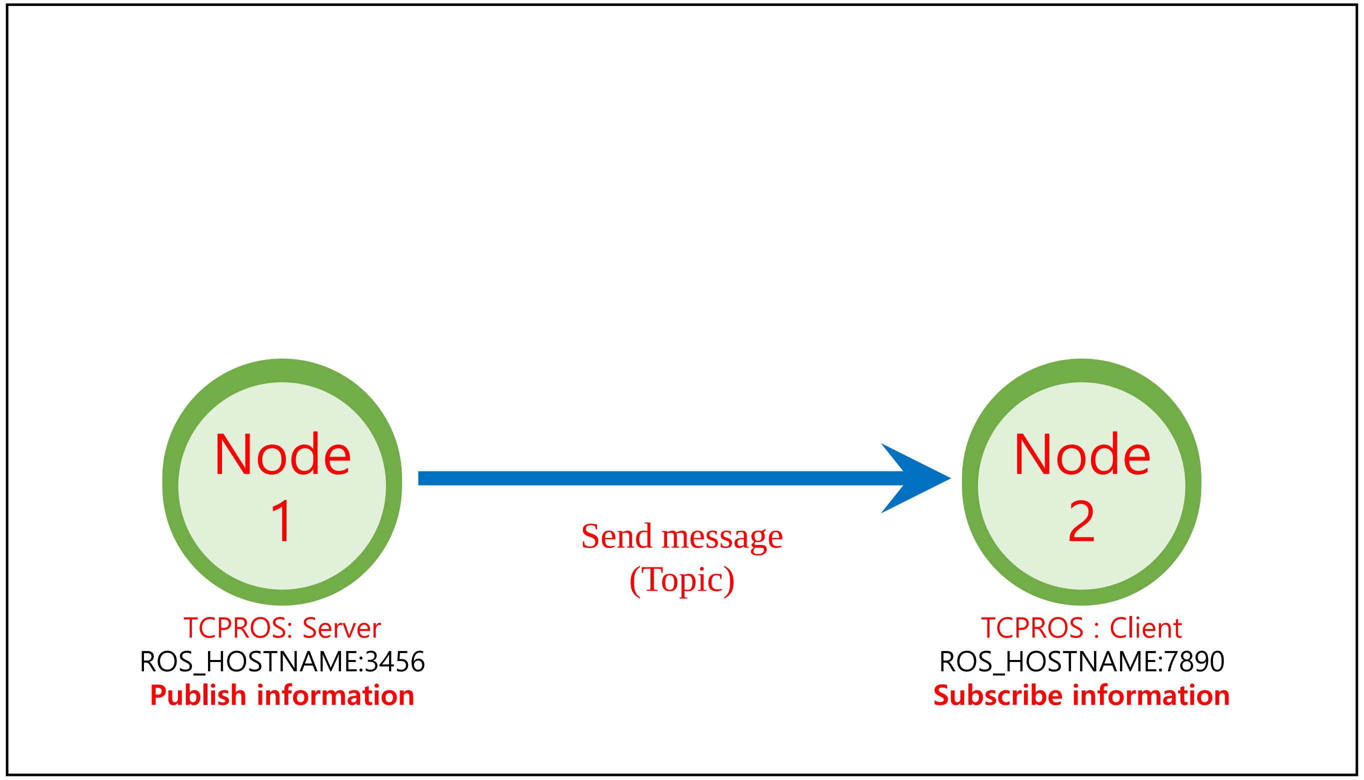
8. Send message

- The publisher node sends a message to the subscriber node (topic)



Understanding message communication

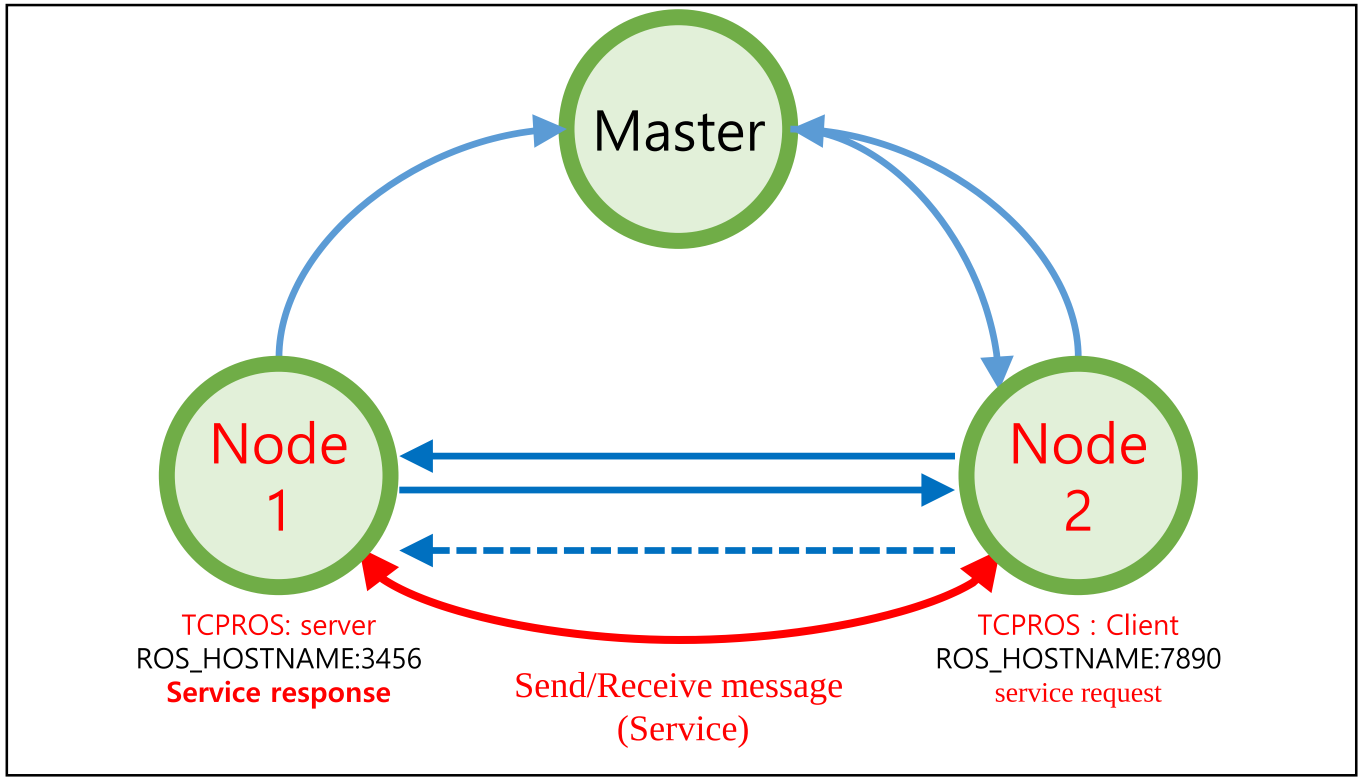
- In Topic mode, messages are continuously transmitted unless the connection is terminated. That is, continuity.



Understanding message communication

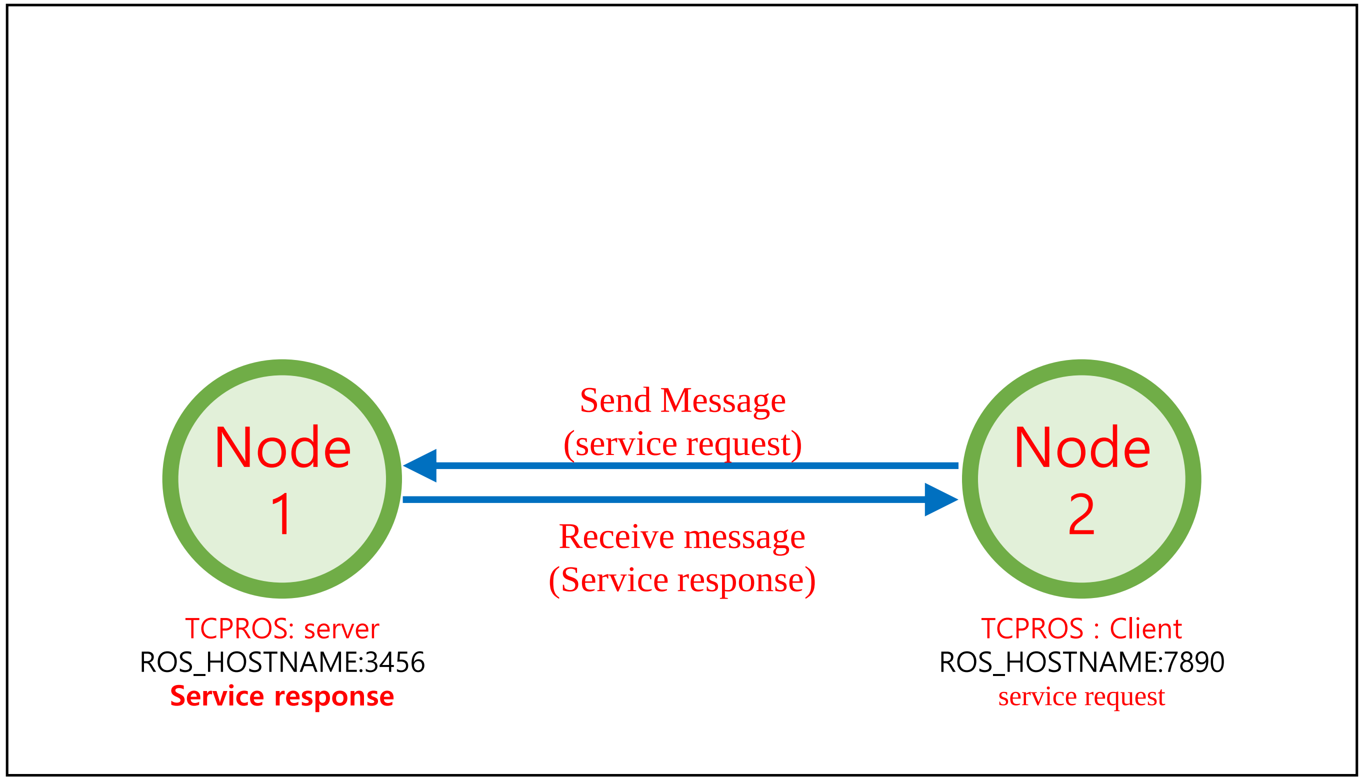
9. Service Request and Response

- For only once, service request and service response are performed and disconnected from each other.



Understanding message communication

- Unlike the topic, the service connects only once and disconnected after a service request and a service response are performed. That is, it is one-time.

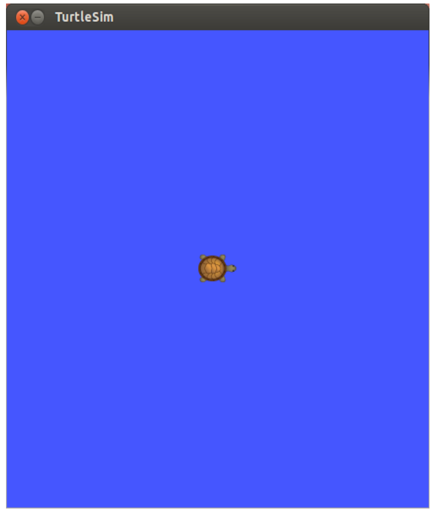


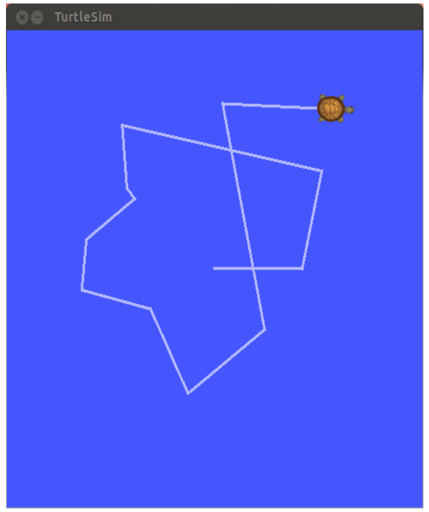
Summing up again!

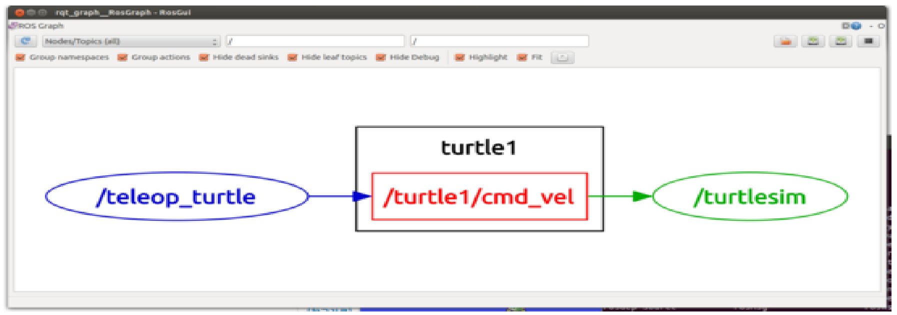
Understanding the message communication concept!

- **turtlesim package**

  - roscore
  - rosrun turtlesim turtlesim_node
  - rosrun turtlesim turtle_teleop_key
  - rosrun rqt_graph rqt_graph

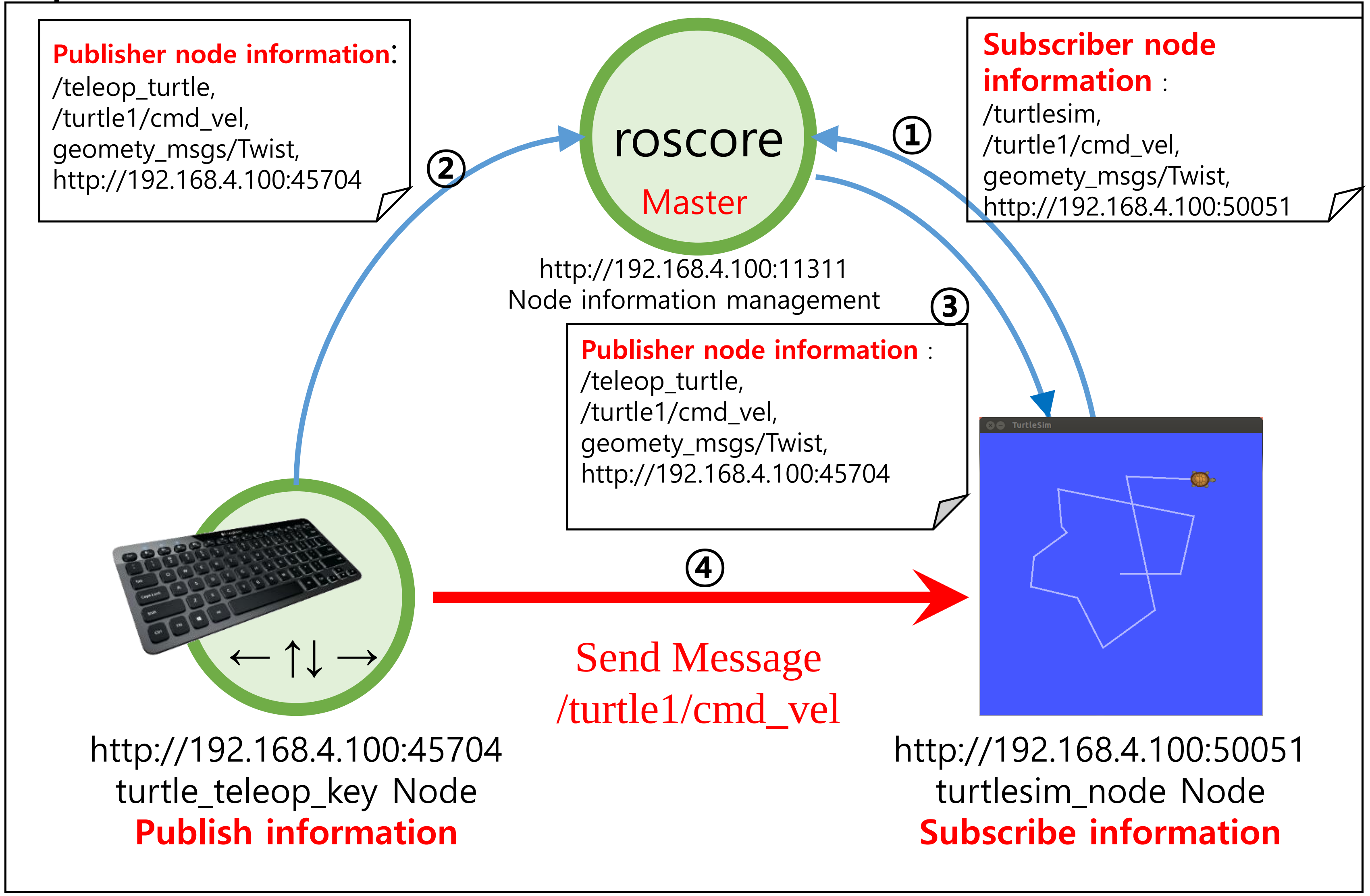






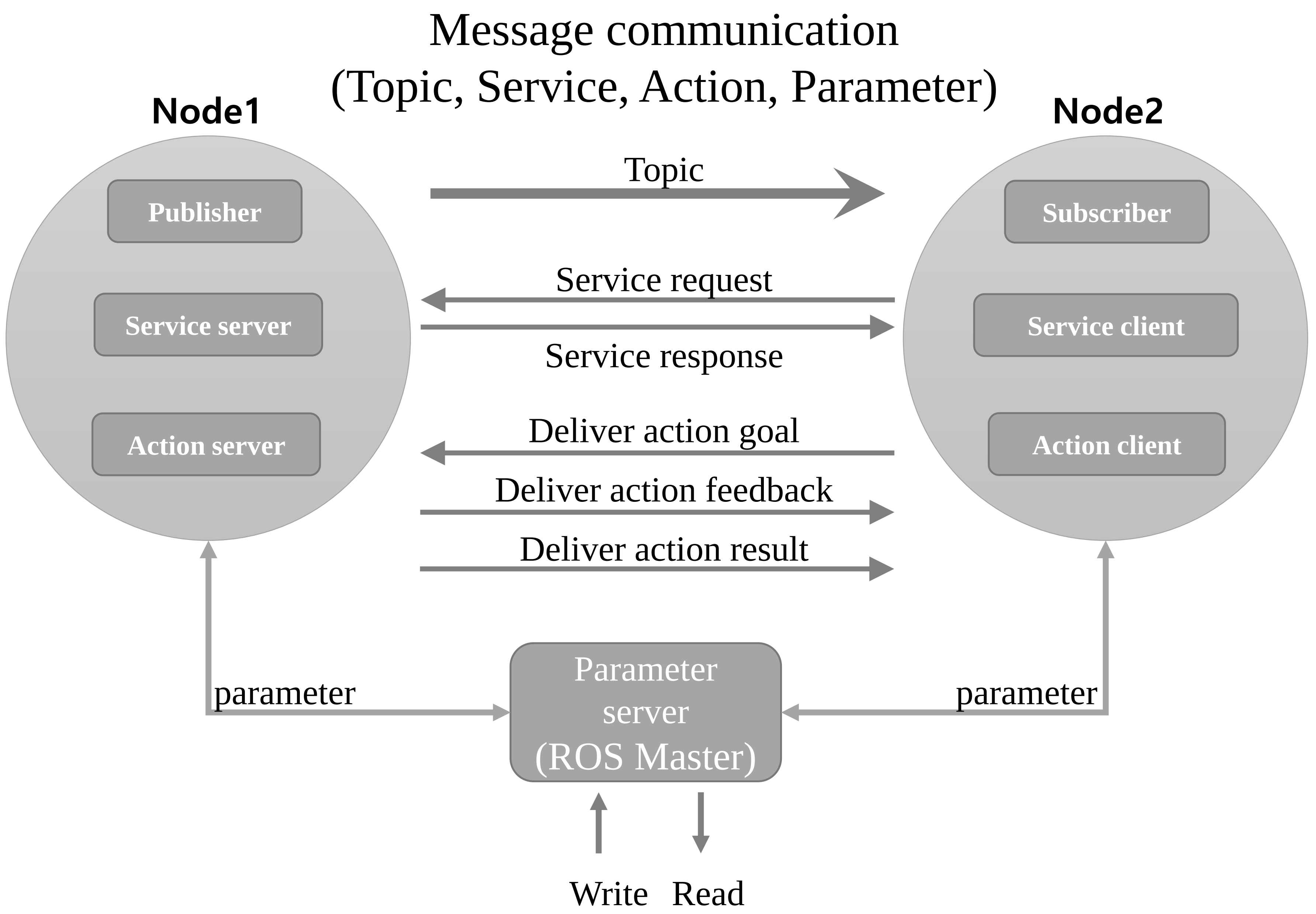
Catching the message communication concept!

- 10. Example! turtlesim



Message?

ROS Message

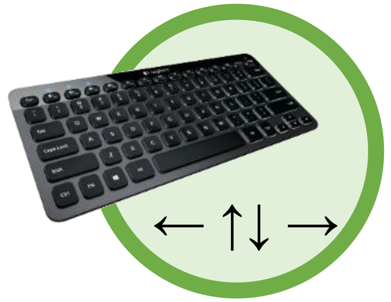


Message communication
(Topic, Service, Action, Parameter)
Node1
Node2
Publisher
Service server
Action server
Subscriber
Service client
Action client
Topic
Service request
Service response
Deliver action goal
Deliver action feedback
Deliver action result
parameter
parameter
Parameter server (ROS Master)
Write
Read

ROS Message

- Message is a type of data travel around nodes
  - Topics, services, and actions all use messages
    - http://wiki.ros.org/msg
    - http://wiki.ros.org/common_msgs
  - **Simple type**
    - ex) integer, floating point, boolean
    - http://wiki.ros.org/std_msgs
  - **A simple data structure containing messages in a message**
    - ex) geometry_msgs/PoseStamped
    - http://docs.ros.org/api/geometry_msgs/html/msg/PoseStamped.html
  - **An array data structure in which messages are listed**
    - ex) float32[ ] ranges
    - ex) sensor_msgs/LaserScan
    - http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html

ROS Message (ex: geometry_msgs/Twist)



http://192.168.4.100:45704
turtle_teleop_key Node
Publish Information

Send message
/turtle1/cmd_vel
(geometry_msgs/Twist)

http://192.168.4.100:50051
turtlesim_node Node
Subscribe information

```
[geometry_msgs/Twist]

Vector3  linear
Vector3  angular
```

```
[geometry_msgs/Vector3]
float64 x
float64 y
float64 z
```

```
[geometry_msgs/Vector3]
float64 x
float64 y
float64 z
```

http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html

Name, TF

Client Library

communication between heterogeneous devices

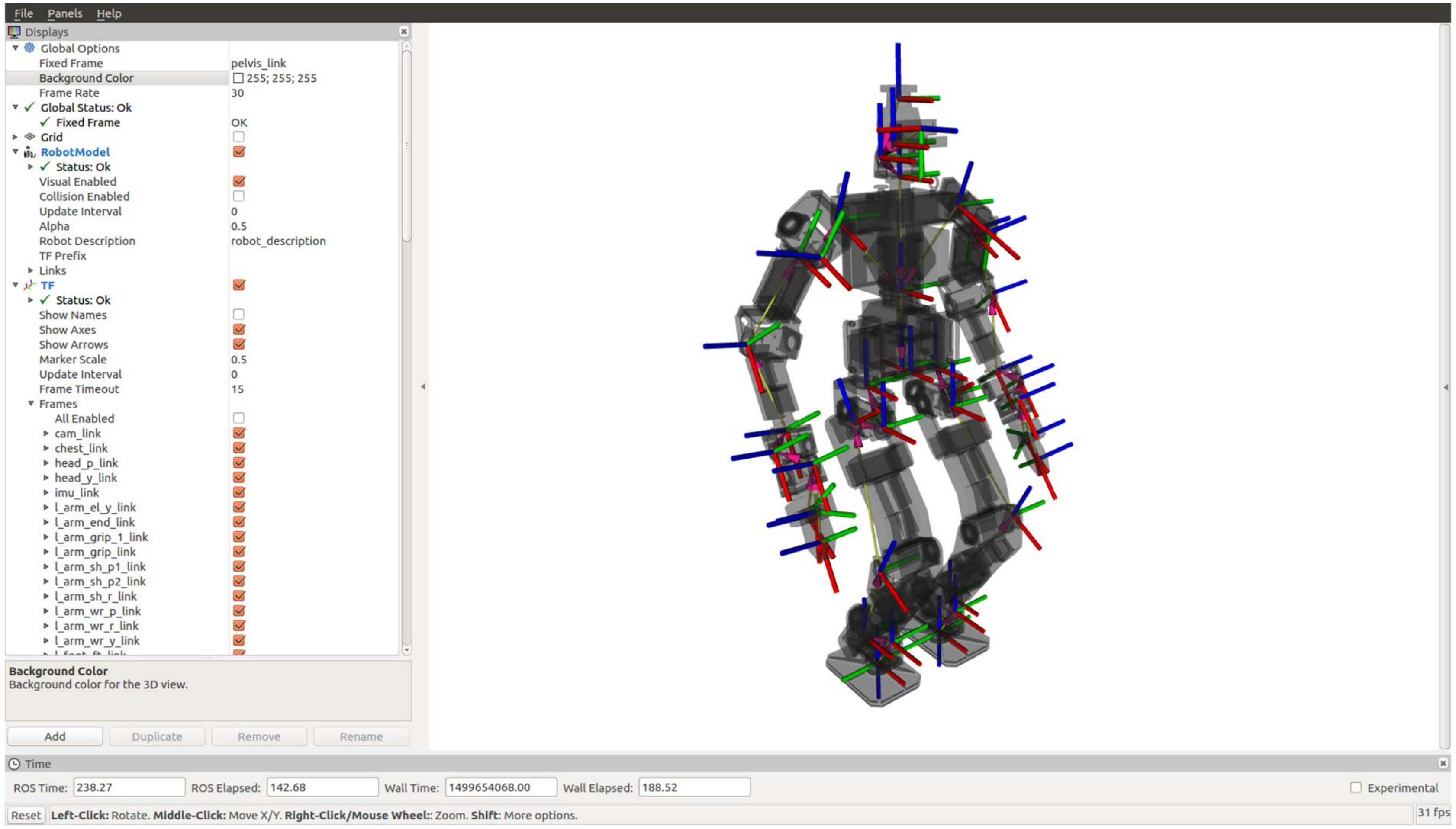
Names

- Name
  - **A unique identifier** for a Node or a message (topic, service, action, parameter)
  - ROS supports abstract data types called **graphs**
  - **Global**
    - Use the name as is or prepend a slash (/) to the name.
  - **Private**
    - Prepend a tilde (~) to the name
  - An example is covered in Chapter 7, ROS Basic Programming, roslaunch.

Node	Relative (default)	Global	Private
/node1	bar -> /bar	/bar -> /bar	~bar -> /node1/bar
/wg/node2	bar -> /wg/bar	/bar -> /bar	~bar -> /wg/node2/bar
/wg/node3	foo/bar -> /wg/foo/bar	/foo/bar -> /foo/bar	~foo/bar -> /wg/node3/foo/bar

Coordinate transformation(TF, transform)

- Relative coordinate transformation of each joint
  - Indicates the relationship between joints in the form of tree structure
  - An example is covered in Chapter 10 TF and Chapter 13 Modeling



Global Options	
Fixed Frame	pelvis_link
Background Color	☐ 255; 255; 255
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☐
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	0.5
Robot Description	robot_description
TF Prefix	
Links	
TF	✓
✓ Status: Ok	
Show Names	☐
Show Axes	✓
Show Arrows	✓
Marker Scale	0.5
Update Interval	0
Frame Timeout	15
Frames	
All Enabled	☐
cam_link	✓
chest_link	✓
head_p_link	✓
head_y_link	✓
imu_link	✓
l_arm_el_y_link	✓
l_arm_end_link	✓
l_arm_grip_1_link	✓
l_arm_grip_link	✓
l_arm_sh_p1_link	✓
l_arm_sh_p2_link	✓
l_arm_sh_r_link	✓
l_arm_wr_p_link	✓
l_arm_wr_r_link	✓
l_arm_wr_y_link	✓

Background Color
Background color for the 3D view.

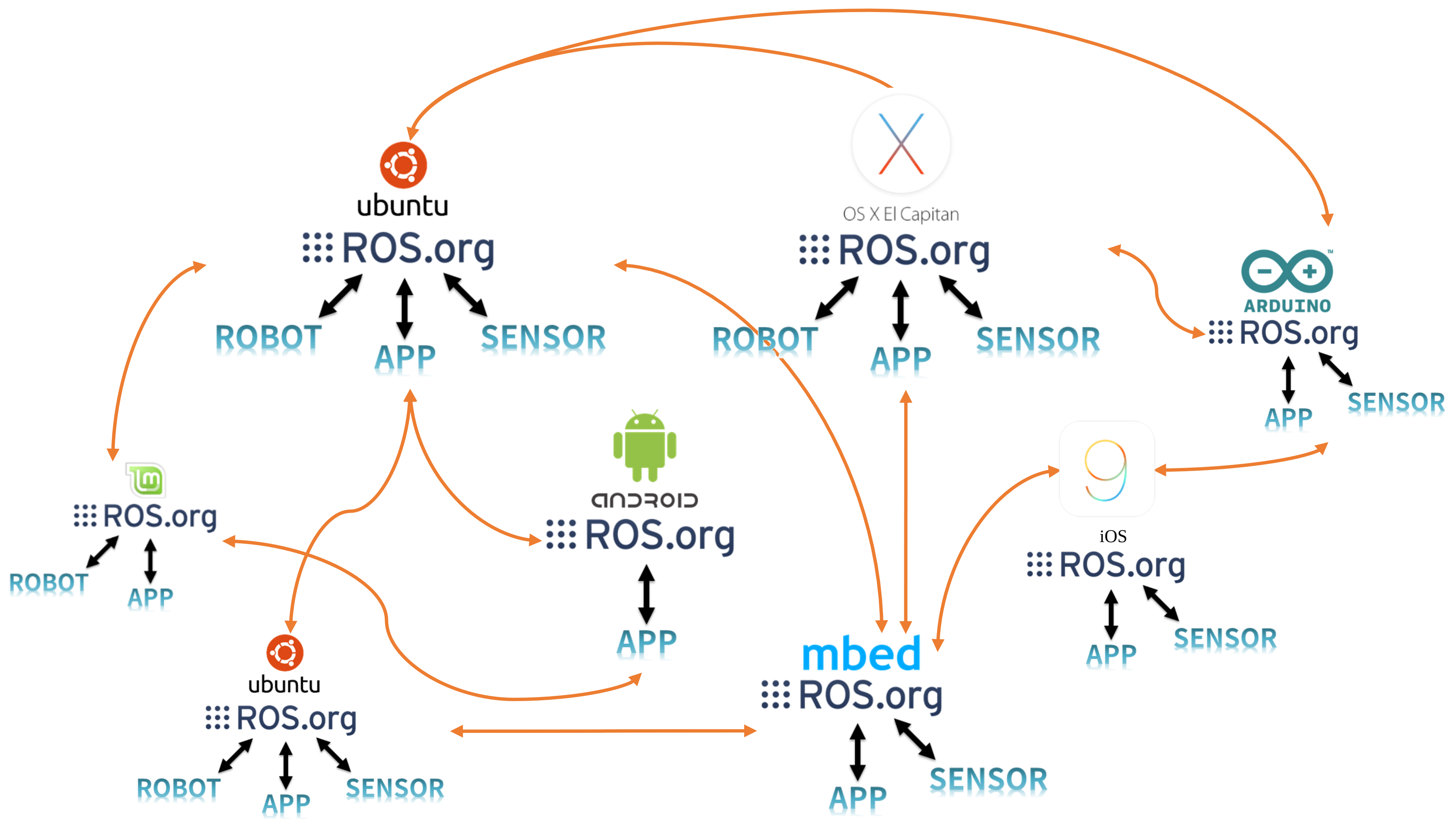
ROS Time: 238.27 ROS Elapsed: 142.68 Wall Time: 1499654068.00 Wall Elapsed: 188.52 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 31 fps

Client Library

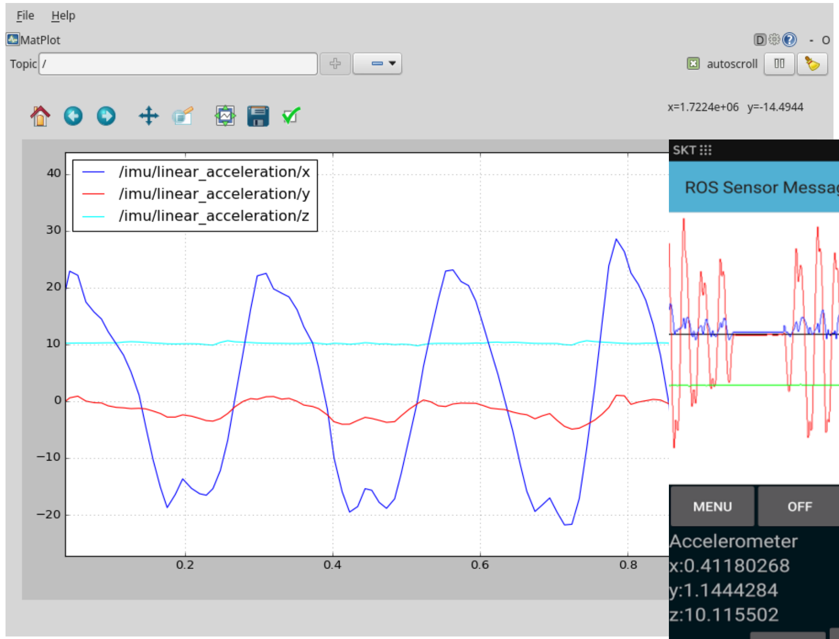
- Supports various programming languages
  - roscpp, rospy,roslisp
  - rosjava, roscs, roseus, rosgo, roshask, rosnodejs, RobotOS.jl, roslua, PhaROS, rosR, rosruby, Unreal-Ros-Plugin
  - MATLAB for ROS
  - LabVIEW for ROS

Communication between heterogeneous devices

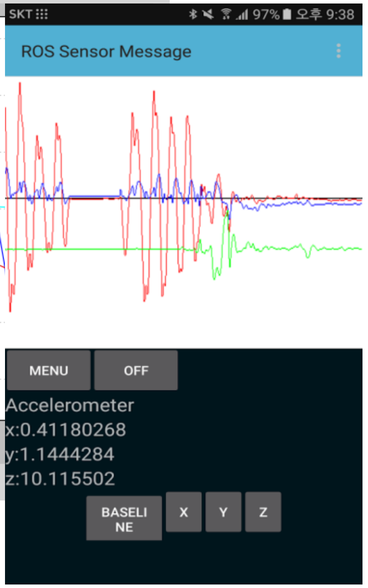


Communication between heterogeneous devices

- Example 1: Transferring images remotely (see Chapter 8, Camera)
- Example 2: Checking the acceleration value of your Android smartphone on your PC (APP)
- Example 3: Controlling TurtleBot with Android Smartphone (APP)

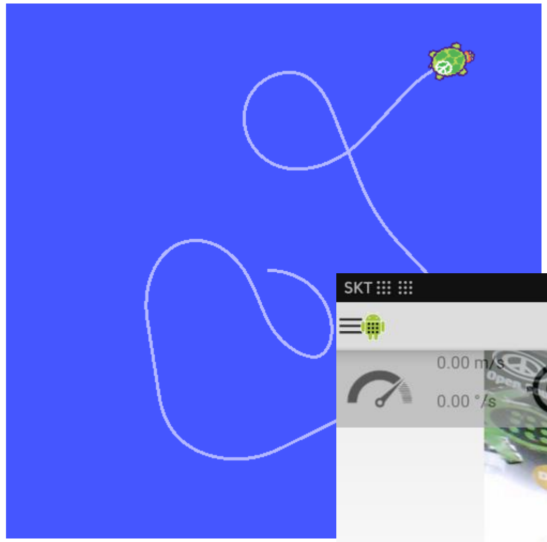


[PC]

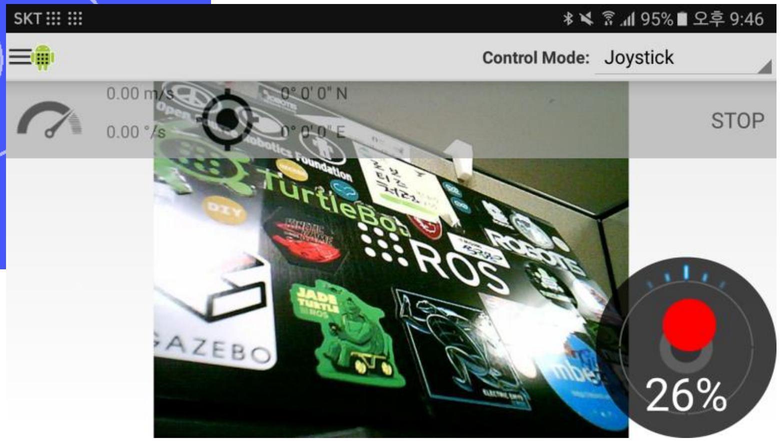


Accelerometer
x:0.41180268
y:1.1444284
z:10.115502

[Smartphone]



[PC]



[Smartphone]

Question Time!

Advertisement #1

✓ Download link

✓ Language:

English, Chinese, Japanese, Korean

"ROS Robot Programming"

A Handbook is written by TurtleBot3 Developers

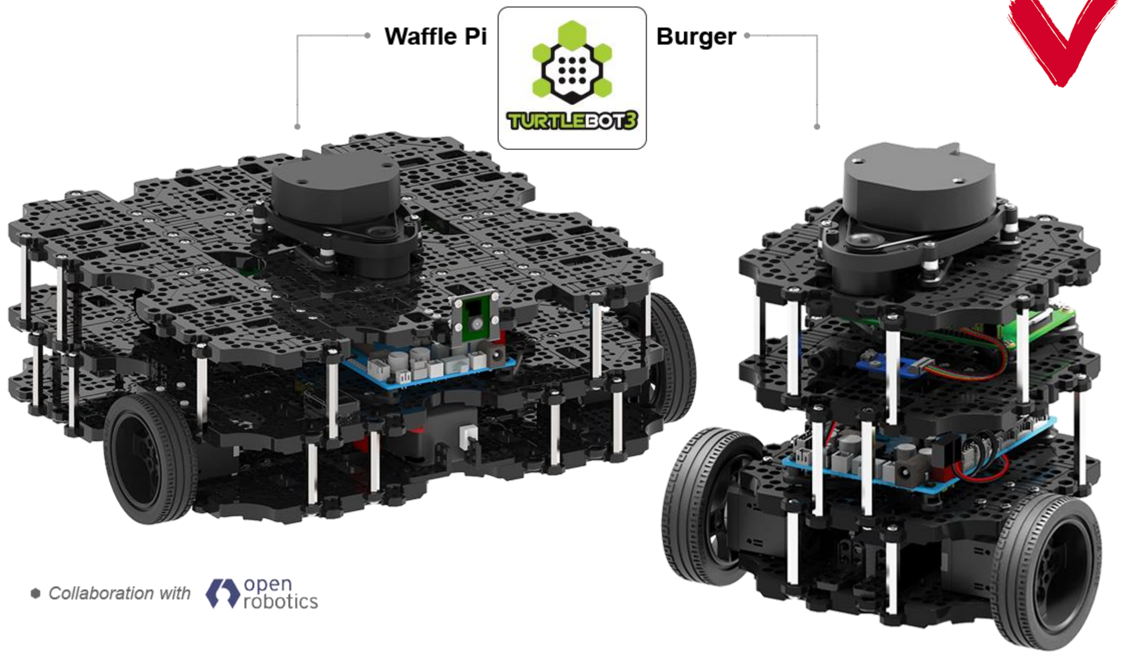
Advertisement #2

TURTLEBOT3

AI Research Starts Here
ROS Official Platform

TurtleBot3 is a new generation mobile robot that’s modular, compact and customizable. Let's explore ROS and create exciting applications for education, research and product development.

✓ Direct Link



Advertisement #3

Robot Source

www.robotsource.org

The ‘RobotSource’ community is the space for people making robots.

We hope to be a community where we can share knowledge about robots, share robot development information and experiences, help each other and collaborate together. Through this community, we want to realize open robotics without disti nguishing between students, universities, research institutes and companies.

Join us in the Robot community ~

END.