BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF EEE

EEE-404

INTRODUCTION TO

Prepared BY

Angkon Deb, MSc Student, EEE, BUET, Dhaka-1000

TOPICS

THEORETICAL

- What is ROS
- Why ROS
- ROS and OS
- ROS Topics
- ROS SUBSCRIBER
- ROS PACKAGE
- ROS PUBLISHER
- ROS MESSAGE
- ROS RQT-GRAPH
- ROS NODES
- ROS SERVICE
- ROS CLIENT

PRACTICAL

- INSTALLATION
- VERSIONS
- LISTENER TALKER
- 2D TURTLE-BOT
- GAZEBO
- OBSTACLE AVOIDANCE ROBOT

ROS

What is ROS?

•Full Form: Robot Operating System

•A flexible framework for writing robot software.

•Provides libraries and tools to help developers create robot applications.

Why Use ROS?

•Simplifies complex robot programming.

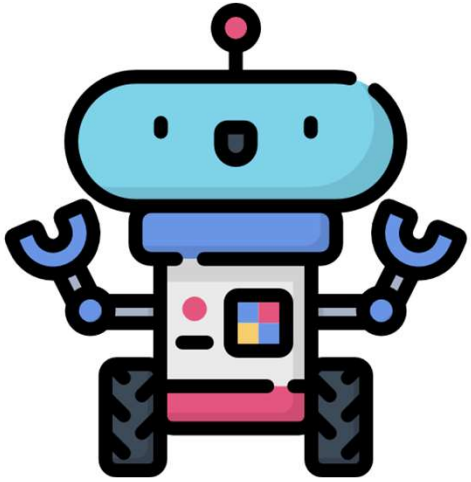
•Supports multiple programming languages (e.g., Python, C++).

•Seamless integration with simulators (e.g., Gazebo, RViz).

•Scalable for multi-robot systems.

•Used in cutting-edge research and industrial applications

Think About a Obstacle Avoidance Robot!



What type of FUNCTIONALITES it has? And which sensors it uses?

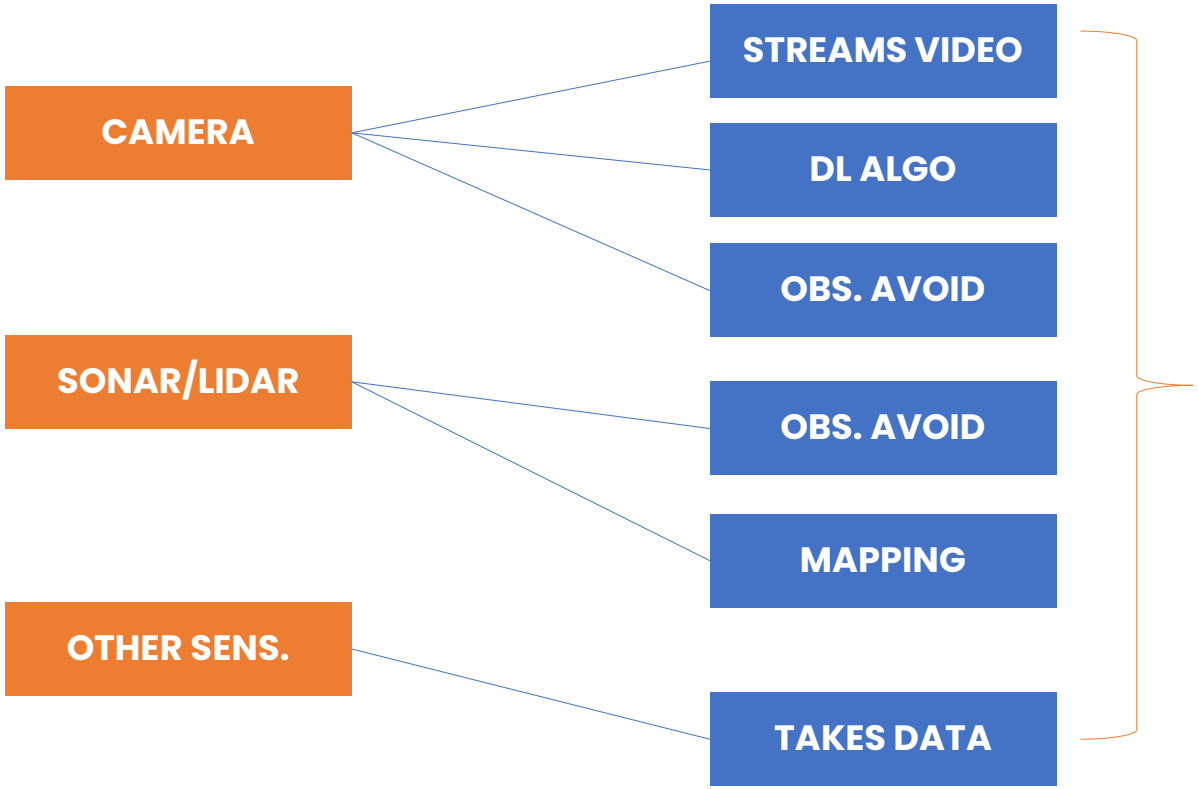
Let’s set some features!

1. It should be able to move
2. It should avoid the obstacles
3. It should find a way to move away from the obstacles
4. It will be able to capture video
5. It will be able to process video do some task, lets say it will collect trash, or it will hold something.

Now think about the Sensors and activators you need

1. Sonar < - For Obstacle detection
2. LIDAR < - For Obstacle detection
3. CAMERA < - For Obstacle detection/ Video Streaming
4. MOTOR Controller < - For Directional movement
5. Arm < - For Specific Tasks
6. Other sensors < - For any other task or data collections etc.
7. Image processing Capability < - For video/Image processing, say a deep learning algorithm

Lets assume you have tested all the components separately and they are all working just fine.



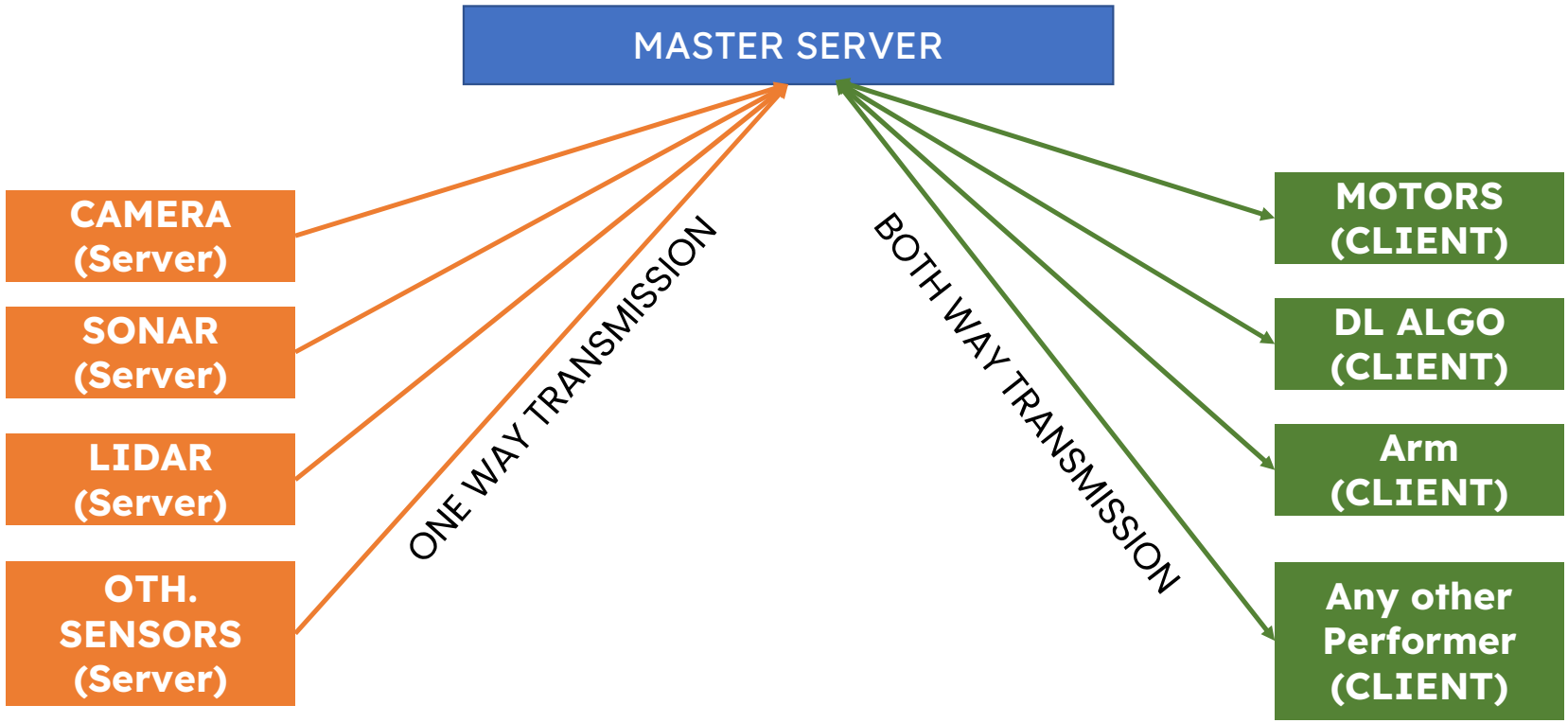
How to integrate all of these?

Challenges:

1. All of them should work simultaneously
2. No serial program will help
3. Need Parallel program
4. One sensor data can perform multiple task. For example: Camera. It does video streaming, on-board Image Proc., and obstacle avoidance if needed

SOLUTION

USE CLIENT-SERVER



That is what ROS do under the hood!
It is basically a client-server module that can perform multiple objective at the same time(simultaneously) .

However ROS has its own naming system for this. It does not call it server or client only.

ROS MASTER

Remember the master server we talked about! That is the ROS Master

It is also knows as ROS Core.

It connects all the program.

If the ROS CORE does not work no program can work. They are all dependent to ROS CORE.

Before initiating any program you have to make sure ROS Core is on!

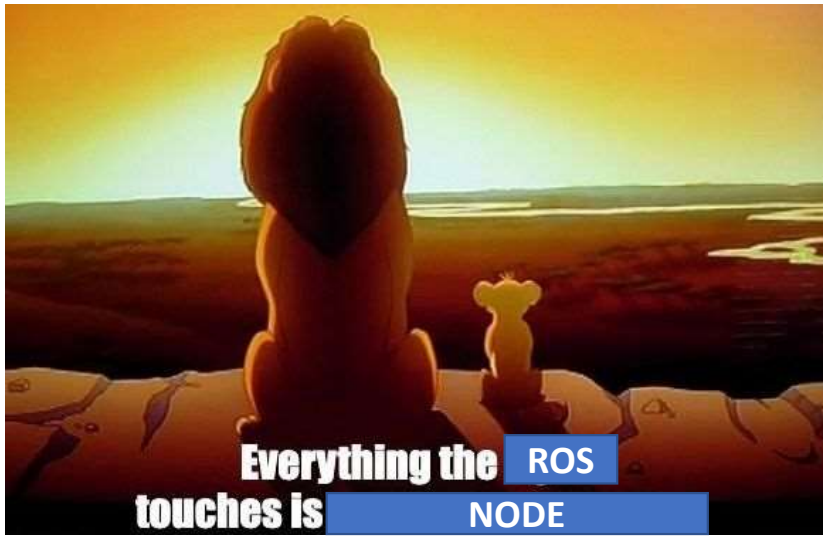
ROS NODE

All the programs you need to write for getting things done are nodes.

A node is a single executable program in ROS responsible for specific tasks.

In our robot----------

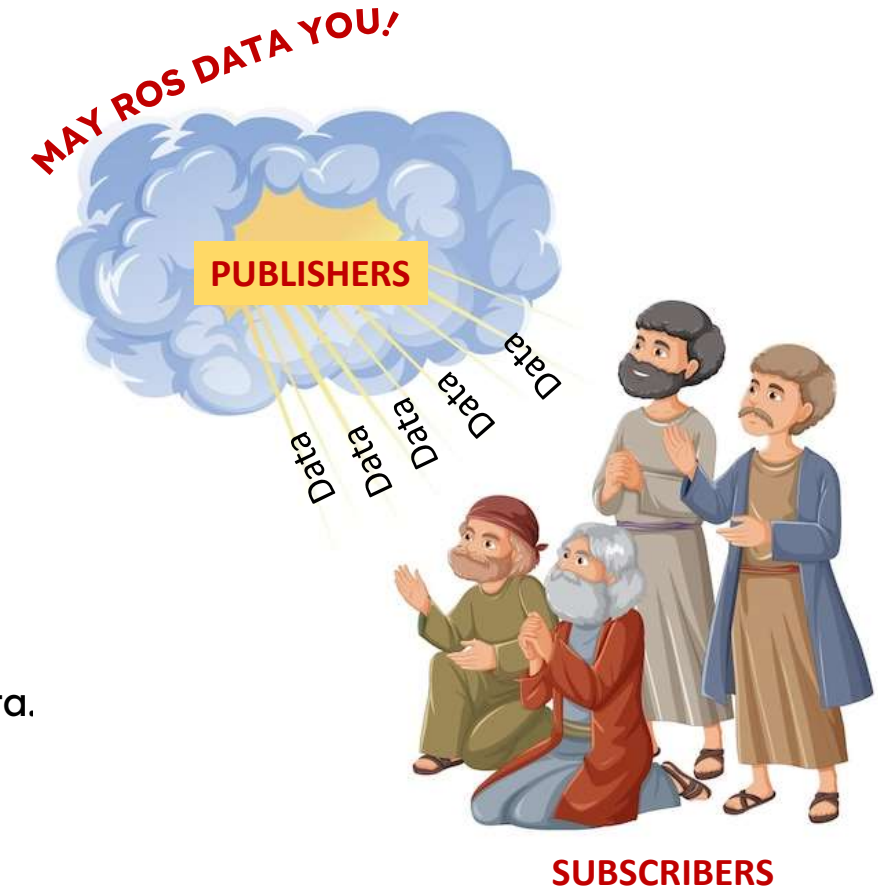
- **Sonar Node:** Reads distance data
- **Lidar Node:** Provides 2D/3D obstacle mapping.
- **Camera Node:** Streams video and processes images for object detection using deep learning.
- **Arm Control Node:** Manages the robotic arm for picking up trash.
- **Motor Control Node:** Controls wheel movements based on navigation logic.



ROS PUBLISHER

Typically we call it the server. But in ROS we say PUBLISHER. It only SENDS data. And does not RECEIVE any.

A publisher is a node that sends data to a topic.



In our Robot:

Sonar Node: Sends data. So it's a publisher.

LIDAR Node: Sends data. So it's a publisher

Camera Node: Stream Footage. Does not receive data.

Other sensors: If you use any other sensor that only takes data is also a publisher.

ROS SUBSCRIBER

It's the END EFFECTOR. It interacts with the environment based on the received data.

A subscriber listens to a topic and acts on the received data.

In our Robot:

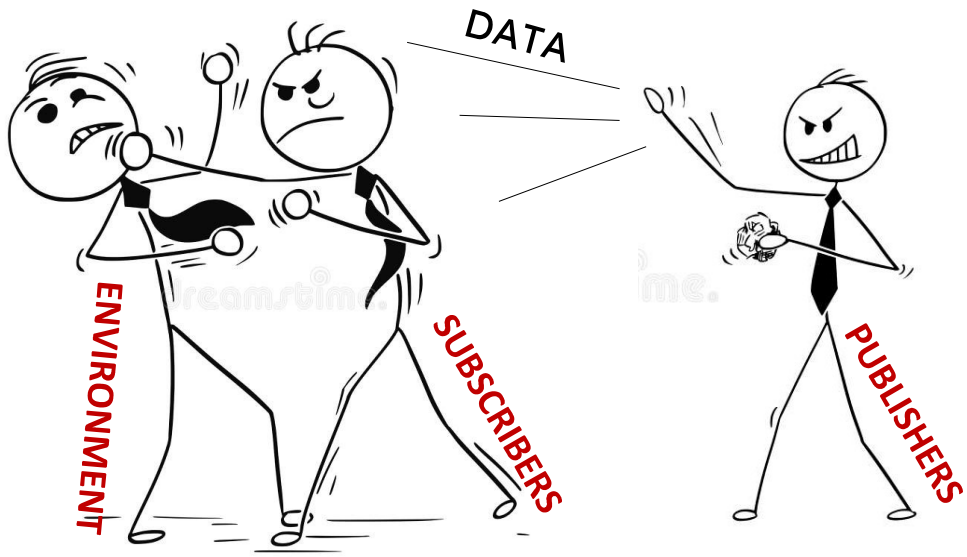
Obstacle avoidance node

Image Processing Node

Arm Node

are SUBSCRIBER.

They can listen and perform.



ROS TOPIC

You may think PUBLISHER directly publishes data to SUBSCRIBER. But it is not the case. What if multiple factors are there for SUBSCRIBERS. TOPIC is the pipeline where the Publisher sends data to. Subscribers takes data from the TOPIC. In this way the PUBLISHERS do not require the info of SUBSCRIBERS.

A topic is a named channel through which nodes exchange data. Publishers send messages, and subscribers listen to them.

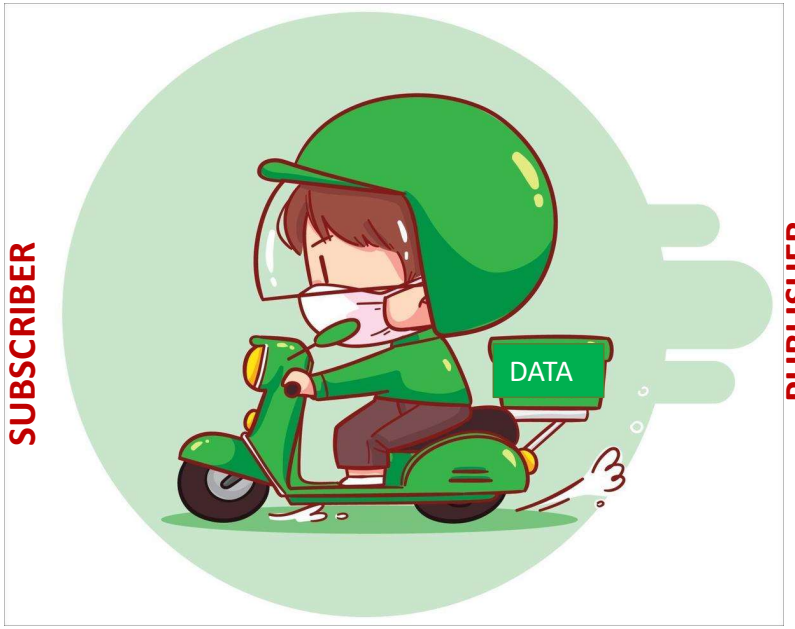
/sonar_data: <-TOPIC

Publisher: Sonar Node

Subscriber: Obstacle Avoidance Node

Message Type: sensor_msgs/Range

Purpose: Shares distance readings from sonar sensors for detecting nearby obstacles.



ROS SERVICE

A *service* allows request-response communication between nodes.

In the robot:

•A service can be set up to handle **robotic arm control**, where the Arm Control Node acts as a *server* and responds to requests (e.g., "Pick up trash at (x, y)").

•The Navigation Node might call a service to calculate paths dynamically.

SUBSCRIBER

SERVICE VS TOPIC

Aspect	Topic	Service
Purpose	Continuous data streaming or broadcasting.	On-demand request-response communication.
Communication	One-way (Publisher → Subscriber).	Two-way (Client ↔ Server).
Use Case	Regular updates, like sensor data or commands.	Specific actions, like querying or triggering an operation.
Persistence	Stateless; data is not stored once published.	Stateful; explicitly initiated communication.
Initiation	Publisher continually pushes data.	Client initiates the request, and Server responds.
Timing	Asynchronous and continuous.	Synchronous; requires waiting for a response.
Example Usage	Sensor readings, video streams, robot state updates.	Starting/stopping a task, querying battery level.
Implementation	Topics use publishers and subscribers.	Services use clients and servers.

ROS SERVER AND CLIENT

These terms are not used a lot. They are basically PUBLISHERs and SUBSCRIBERs.

What they do:

The server executes a task and returns a result.

The client sends a request to the server.

In the robot:

The Arm Control Node acts as the server for picking up trash.

The Main Control Node sends client requests to control the arm or fetch robot status.

ROS MESSAGE

Till now when we were taking about data, we were actually refereeing to MESSAGE

It is a structured data type. It reduces the complexities to process the data type from different sensors.

A message is a structured data format used to exchange information in topics or services.

ROS MESSAGE

For Example: Say we are dealing with the Movement or Position data.

We do not need variables to take Linear and Angular data. ROS MESSAGE handles that for us

sensor_msgs/Image

```
header:
  seq: 1234
  stamp: 2024-12-09 12:34:56
  frame_id: "camera_frame"
height: 480
width: 640
encoding: "rgb8"
is_bigendian: 0
step: 1920
data: [255, 0, 0, 255, 255, 255, ...]  # 1D array of pixel values
```

geometry_msgs/Twist

```
linear:
  x: 0.5  # Forward velocity in m/s
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.3  # Angular velocity in rad/s
```

ROS MESSAGE: EXAMPLE

Topic	Message Type	Purpose
`/sonar_data`	`sensor_msgs/Range`	Publish sonar distance readings.
`/lidar_scan`	`sensor_msgs/LaserScan`	Publish LIDAR obstacle data.
`/camera_stream`	`sensor_msgs/Image`	Publish live video stream.
`/detection_data`	`TrashLocation` (custom)	Share trash detection coordinates.
`/movement_cmds`	`geometry_msgs/Twist`	Send navigation commands.
`/robot_status`	`RobotStatus` (custom)	Monitor system status.
`/arm_feedback`	`ArmFeedback` (custom)	Provide feedback on arm actions.
`/path_plan`	`nav_msgs/Path`	Publish planned navigation paths.

ROS PACKAGE

It encapsulates all the nodes so that everything works perfectly even if it is ran in different device with different version

A package organizes nodes, configuration files, and scripts for a specific functionality.

In the robot:

sonar_driver: A package for sonar sensor handling.

lidar_mapping: A package for LIDAR data processing.

video_stream: A package for camera streaming and object detection using a deep learning model.

robot_navigation: A package for path planning, obstacle avoidance, and motor control.

Example Flow for the Robot:

Sensors Publish Data: Sonar and LIDAR nodes publish distance readings to sonar_data and lidar_scan.The Camera Node publishes video data to camera_stream.

Obstacle Avoidance Node: Subscribes to sonar_data and lidar_scan.Publishes safe movement commands to movement_cmds.

Motor Control Node: Subscribes to movement_cmds and moves the robot accordingly.

Deep Learning Node: Processes the video feed, detects trash, and publishes coordinates to a detection_data topic.

Robotic Arm Node: Uses a Service (e.g., /arm_control_service) to pick up detected trash at specific coordinates.

Central Node: Coordinates the workflow by interacting with nodes via topics and services.

Demonstration

TALKER - LISTENER

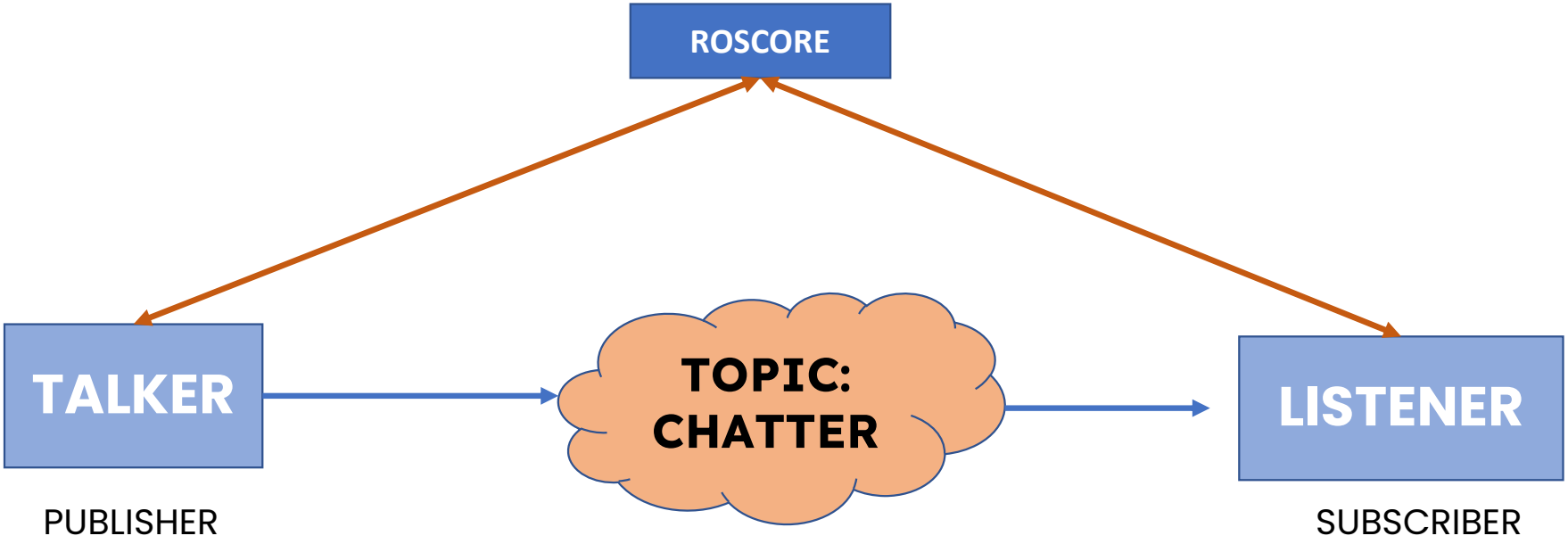
TURTLESIM 2D

TURTLE BOT 3D

ROS Versions are OS specific, you can not run JALISCO in UBUNTU 20.04 LTS, at least, its not practical.

We will be using NOETIC in UBUNTU 20.04 LTS

TALKER- LISTENER



TURTLESIM 2D

Topics:

/turtle1/cmd_vel

/turtle1/pose

SUBSCRIBER:
Sets position

PUBLISHER: Provides
Current position