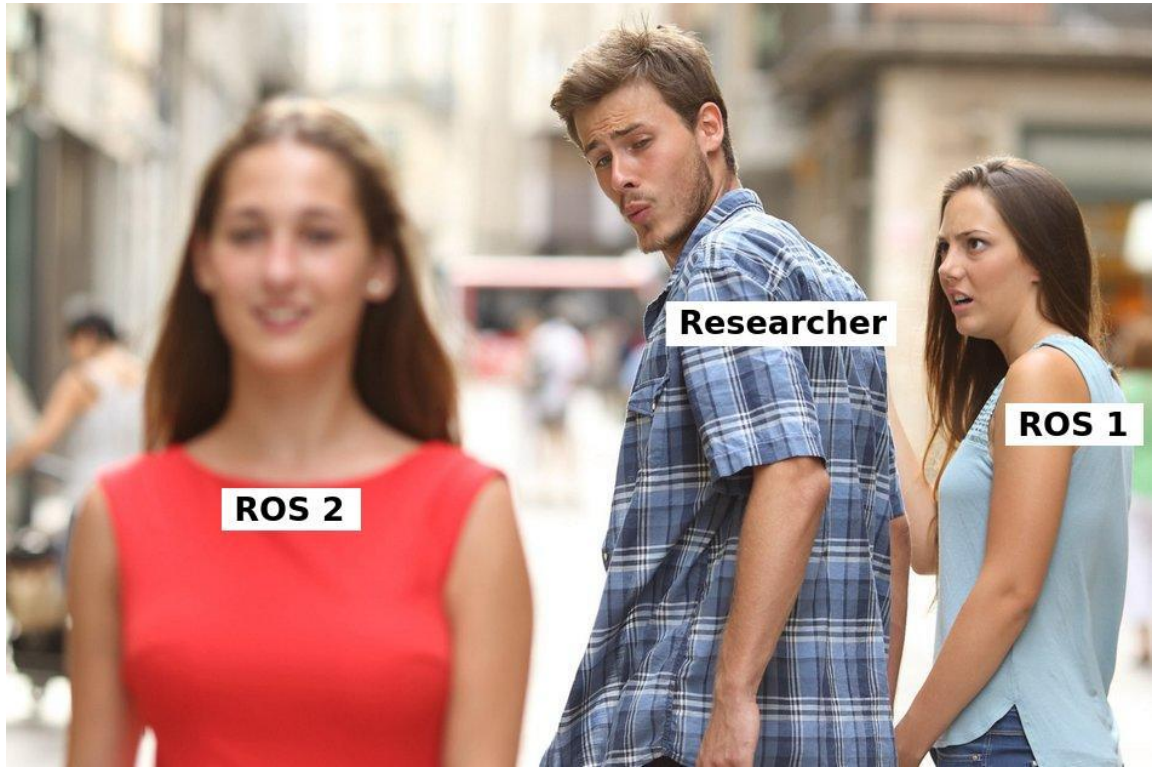
CEG2001 – Tutorial 2

ROS2 Communication

What is ROS 2?

- **Framework:**
  - Robot Operating System 2 is **not an OS** like Windows/Linux.
  - Framework that provides tools and libraries to write **reliable, modern robot** applications.
- **Modular Design:**
  - Encourages breaking a large robot program into many small, independent programs.
- **Decentralized:**
  - ROS 2 uses **DDS (Data Distribution Service)** to allow these programs to talk to each other over a network.
  - Doesn’t need a central server (like ROS 1 did).



Nodes (Your Program)

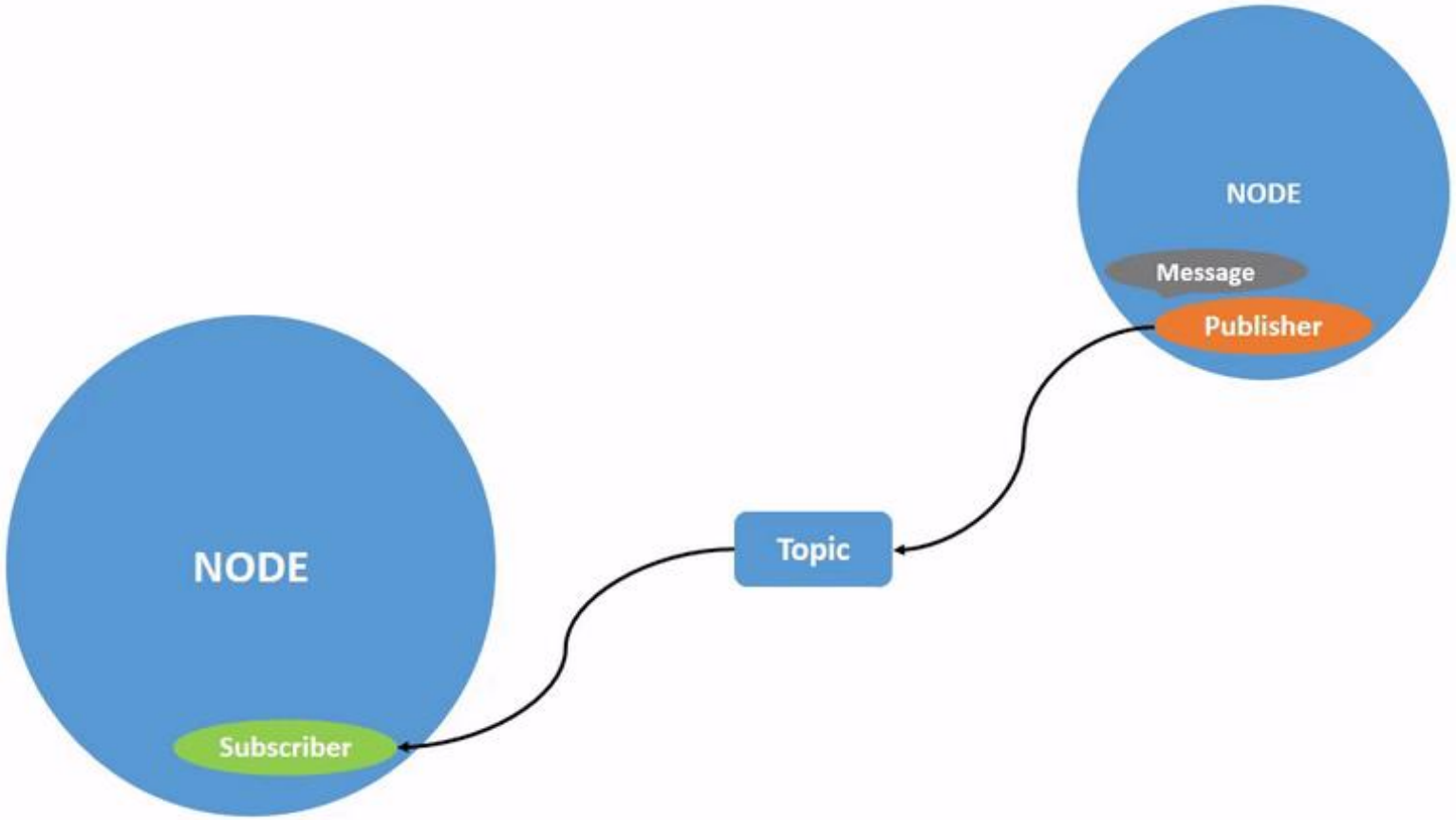
- A **single**, **executable** program in the ROS 2 environment.
- Every distinct job (e.g., reading a camera, calculating a path, sending commands) should be its own Node.
- **Safety:** If one Node crashes, the other parts of the robot can often keep working because they are separate programs (processes).
- **Example Jobs:**

Job	Node Name Example
Seeing	`camera_driver`
Thinking	`behavior_tree`
Moving	`motor_controller`

Topics (Streaming Data)

- **Channels** for constant, one-way streams of data. They are the most common way Nodes communicate.
- **The Publisher-Subscriber Model:**
  - **Publisher:**
    - A Node that continuously sends information onto a Topic
    - e.g., a laser sensor publishing distance readings.
  - **Subscriber:**
    - A Node that receives and acts on that information
    - e.g., a safety system subscribing to sensor data.
  - **Asynchronous:**
    - Data flows freely;
    - the sender doesn't wait for the receiver.
  - **Best for:**
    - High-frequency, continuous data like sensor streams,
    - video feeds,
    - robot pose updates.

The Publisher-Subscriber Model



Services (Request and Reply)

- Used for simple, immediate interactions where one Node needs an answer from another.
- **The Request-Response Model:**
  - **Client:** Asks a specific question or requests an immediate action.
  - **Server:**
    - Receives the request,
    - performs the action,
      - and sends back a single reply.
  - **Synchronous:** The Client waits (is blocked) until the Server responds.
  - **Best for:** Quick tasks like querying the robot's battery level or turning a light on/off.

Parameters (Configuration Settings)

- Simple **configuration values** that belong to a specific Node. They act like settings or variables you can change without touching the Node's main code.
- **Flexible Configuration:** Use parameters to set things like a robot's maximum speed, the frequency of a sensor, or a color threshold.
- **Runtime Changes:** You can change these settings while the Node is running (e.g., turning on "debug mode" instantly).
- **Local to Node:** Parameters only affect the Node they belong to; they are not shared globally.
- **Best for:** Fine-tuning the behavior of a single Node without having to recompile the whole system.

Actions (Long-Running Tasks)

- Actions are for complex, **long-duration tasks**, like "Go to the kitchen."
- **The Action Model adds two features to Services:**
  - **Feedback:** The Server sends back continuous updates on the task's progress (e.g., "I am 50% there").
  - **Preemption:** The Client can send a command to cancel the task while it is running.
- **Best for:** Navigation, complex manipulation, or any goal that takes more than a few seconds to complete.

Summary

Pattern	Data Flow	Purpose	When to Use
Topics	Streaming	Continuous, one-way data flow	High-frequency sensor or state updates.
Services	Request/Reply	Single, immediate function call	Quick queries or instantaneous commands.
Actions	Goal/Feedback/Result	Long-running task with monitoring	Navigation, complex movement routines.
Parameters	Configuration Metadata	Adjusting Node settings at runtime	Setting speeds, colors, or frequency values.

References

- https://www.youtube.com/watch?v=HJAE5Pk8Nyw
- https://docs.ros.org/en/kilted/Tutorials/Beginner-Client-Libraries/Writing-A-Simple-Py-Publisher-And-Subscriber.html