ROS Publisher/Subscriber Model

Kiran Vasudev[1]

[1]Hochschule Bonn-Rhein-Sieg

April 3, 2018

Outline

The Publisher/Subscriber Pattern

- Transport system used to route messages
- A node sends a message by publishing it to a topic
- A node that is interested to access/use data certain data will subscribe to the most appropriate topic
- There can be many existing publishers and subscribers for a single topic
- A single node can publish/subscribe to multiple topics

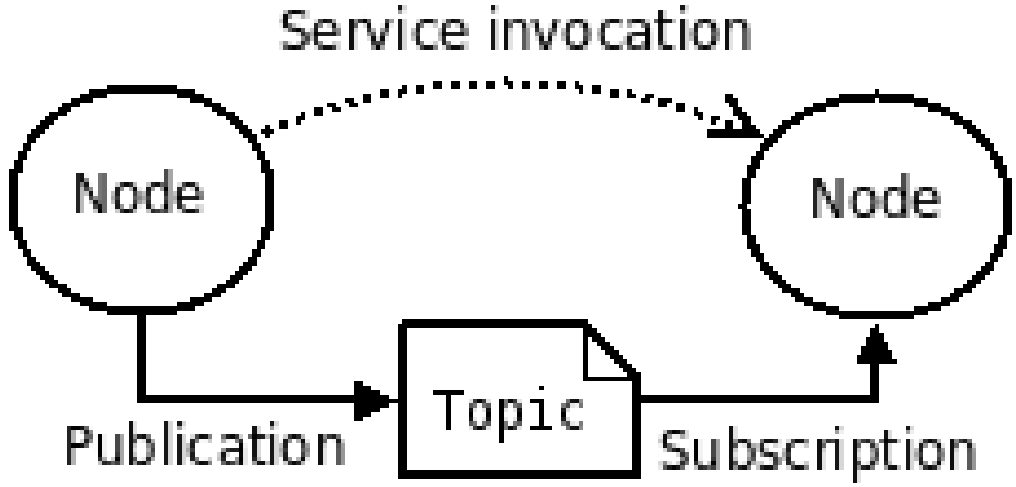

Figure: The publish/subscribe model [1]

Cons of the Publisher/Subscriber Pattern

- The model's many-to-many one-way transport of messages are not appropriate for interactions based on requests/replies.
- This con is overcome by using services (Will be introduced in the next session)

Initializing a publisher and subscriber

- Python
  - **Publisher**
    pub = rospy.Publisher('chatter', String, queue_size=10)
    pub.publish(msg)
  - **Subscriber**
    rospy.Subscriber("chatter", String, callback)
- C++
  - **Publisher**
    ros::Publisher chatter_pub = n.advertise⟨std_msgs::String⟩("chatter", 1000);
    chatter_pub.publish(msg);
  - **Subscriber**
    ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

Task

- Create a new ROS package that has a minimum of two nodes: A publisher and a subscriber
- Define a custom message that contains:
  - Image data
  - LaserScan
  - Pose

  Documentation to create a custom message is found here: **Creating messages**
  Use the documentation of ROS to define the message type. For example, the documentation of the sensor_msgs can be found here: **Sensor messages**
- Play the given rosbag and subscribe to the required topics for your new message in the subscriber node
- Using this information from the bag file, create your new message and publish it
- Once the message is published, subscribe to this message, and display a confirmation message once it has been received

References

1. ROS Wiki
2. Publisher/Subscriber(python)
3. Publisher/Subscriber(cpp)