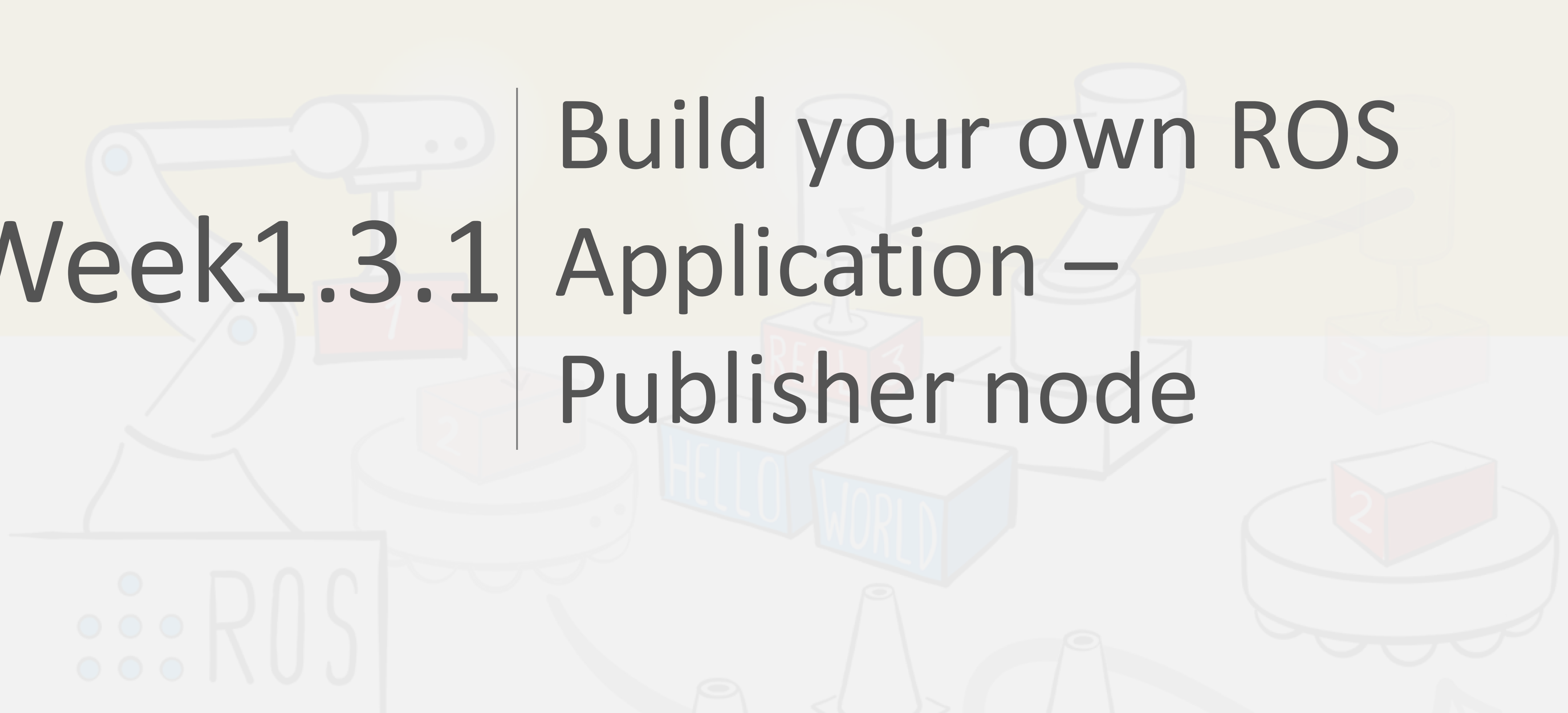


Week1.3.1

Build your own ROS Application – Publisher node

Publisher node

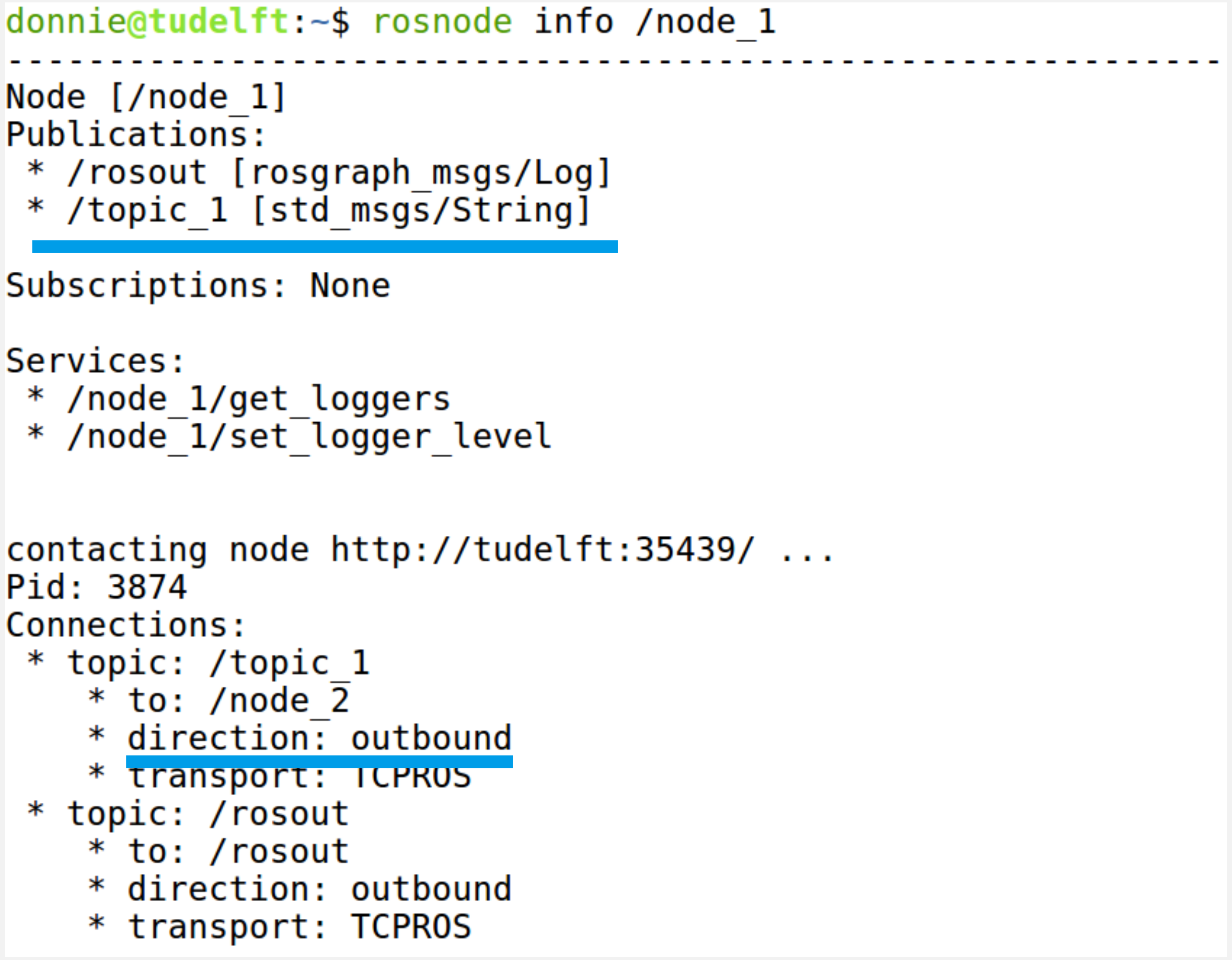
- Nodes are information/data processing units
    - Where does the information come from? Who generates the information?

- A publisher node generates information
    - uses ROS Topic(s) to communicate to other nodes.

- ROS-based robotic applications
    - process raw sensory information from camera, encoders, etc.
    - **publish** processed information **to a topic**.

Publisher node – example output

Output of rosnode info /node_1 command



```
donnie@tudelft:~$ rosnode info /node_1
--------------------------------------------------------------
Node [/node_1]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /topic_1 [std_msgs/String]

Subscriptions: None

Services:
 * /node_1/get_loggers
 * /node_1/set_logger_level


contacting node http://tudelft:35439/ ...
Pid: 3874
Connections:
 * topic: /topic_1
    * to: /node_2
    * direction: outbound
    * transport: TCPROS
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

Publisher node – example code

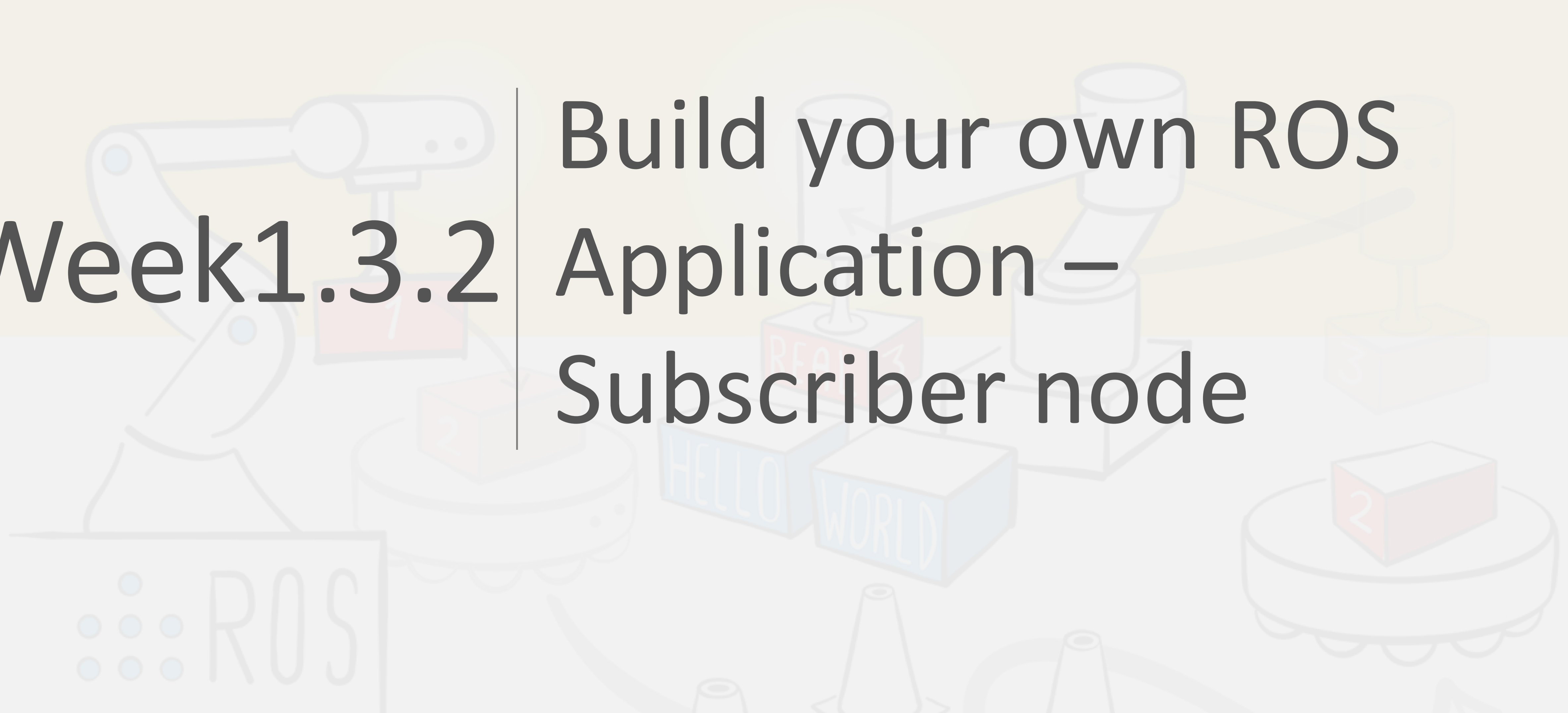
Code for a simple publisher node (1/2)

```
## Node to publish a string topic.


import rospy

from std_msgs.msg import String



def simplePublisher():


    simple_publisher  = rospy.Publisher('topic_1', String,
queue_size = 10)
    rospy.init_node('node_1', anonymous = False)
    rate = rospy.Rate(10)
```

Publisher node – example code

Code for a simple publisher node (2/2)

```
# The string to be published on the topic.
topic1_content = "My first ROS topic"


    while not rospy.is_shutdown():
        simple_publisher.publish(topic1_content)
        rate.sleep()
if __name__ == '__main__':
    try:
        simplePublisher()
    except rospy.ROSInterruptException:
        pass
```



Week1.3.2

Build your own ROS Application – Subscriber node

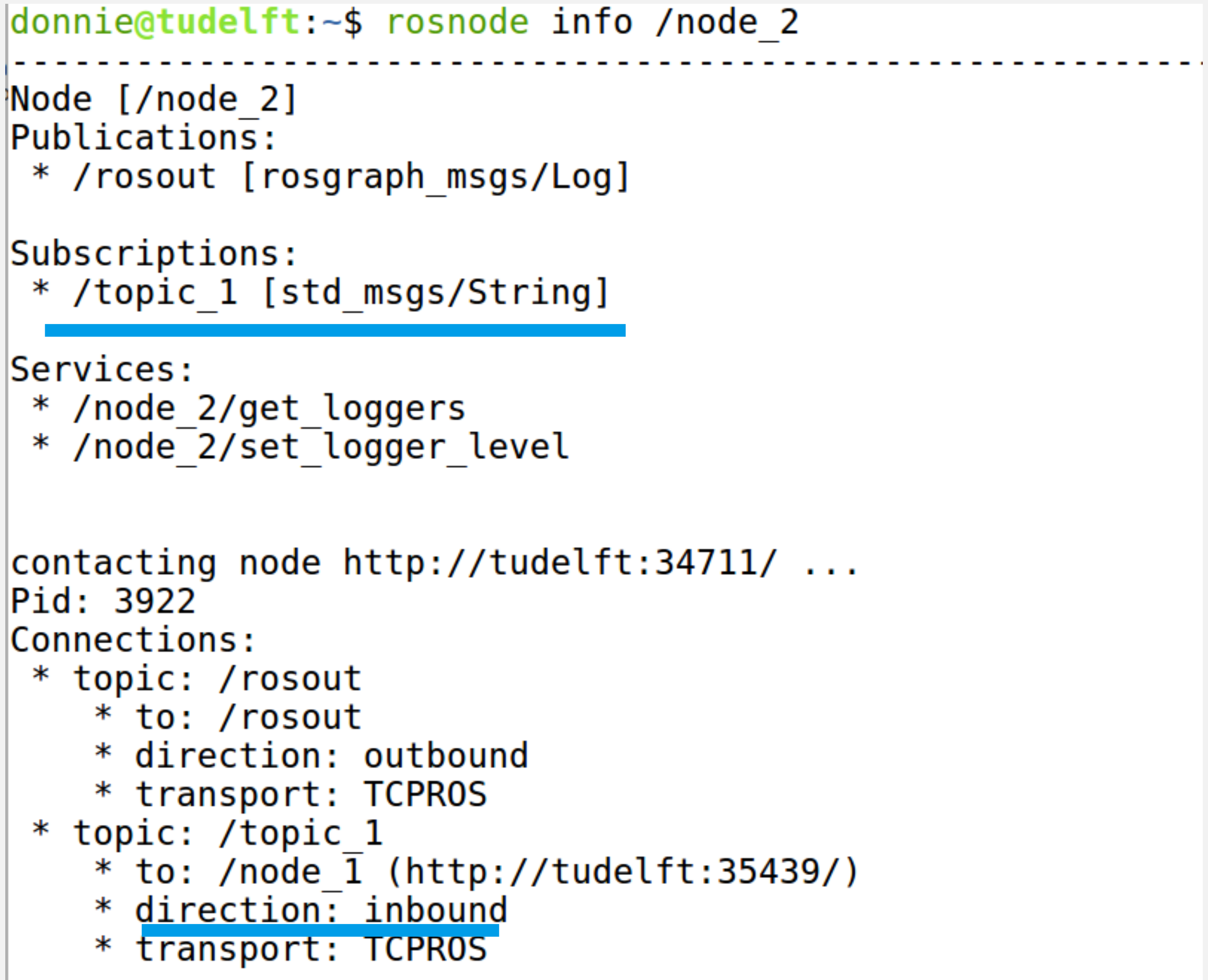
Subscriber node

- A subscriber node receives information
  - **subscribe to** information in **a topic**.
  - uses “topic callbacks” to process received information.

- ROS-based robotic applications
  - monitoring system state such as triggering an alert when close to robot joint limits.

Subscriber node – example output

Output of rosnode info /node_2 command



```
donnie@tudelft:~$ rosnode info /node_2
--------------------------------------------------------------------------------
Node [/node_2]
Publications:
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /topic_1 [std_msgs/String]

Services:
 * /node_2/get_loggers
 * /node_2/set_logger_level


contacting node http://tudelft:34711/ ...
Pid: 3922
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /topic_1
    * to: /node_1 (http://tudelft:35439/)
    * direction: inbound
    * transport: TCPROS
```

Subscriber node – example code

Code for a simple subscriber node (1/2)

```
## Node to subscribe to a string and print the string on
terminal.


import rospy

from std_msgs.msg import String


# Topic callback function.

def stringListenerCallback(data):

    rospy.loginfo('The contents of topic1: %s', data.data)


def stringListener():

    rospy.init_node('node_2', anonymous=False)
```

Subscriber node – example code

Code for a simple subscriber node (2/2)

```
    rospy.Subscriber('topic_1', String, stringListenerCallback)

    # spin() simply keeps python from exiting until this node
is stopped
    rospy.spin()

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

Quick Recap

- Two important building blocks of a ROS application
  - publisher node and subscriber node.
- Publisher node “writes to” one or more ROS topics.
- Subscriber node “reads from” one or more ROS topics and processes the corresponding information in topic callback function.