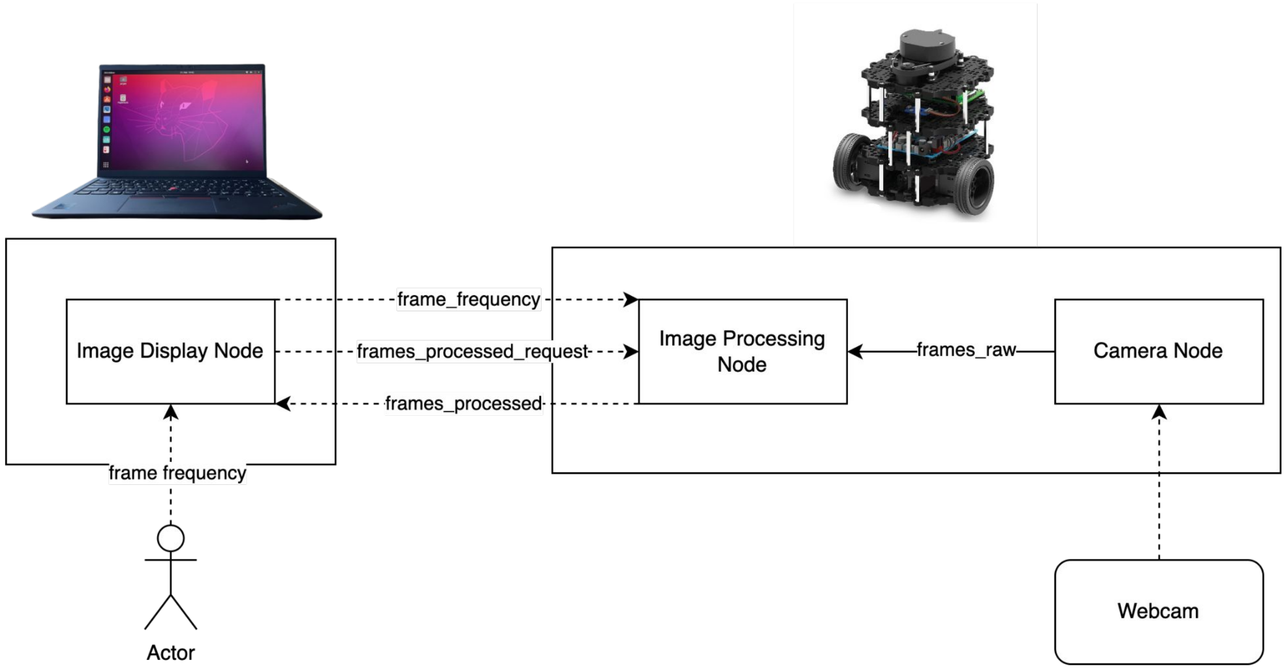
Assignment 1

Architecture

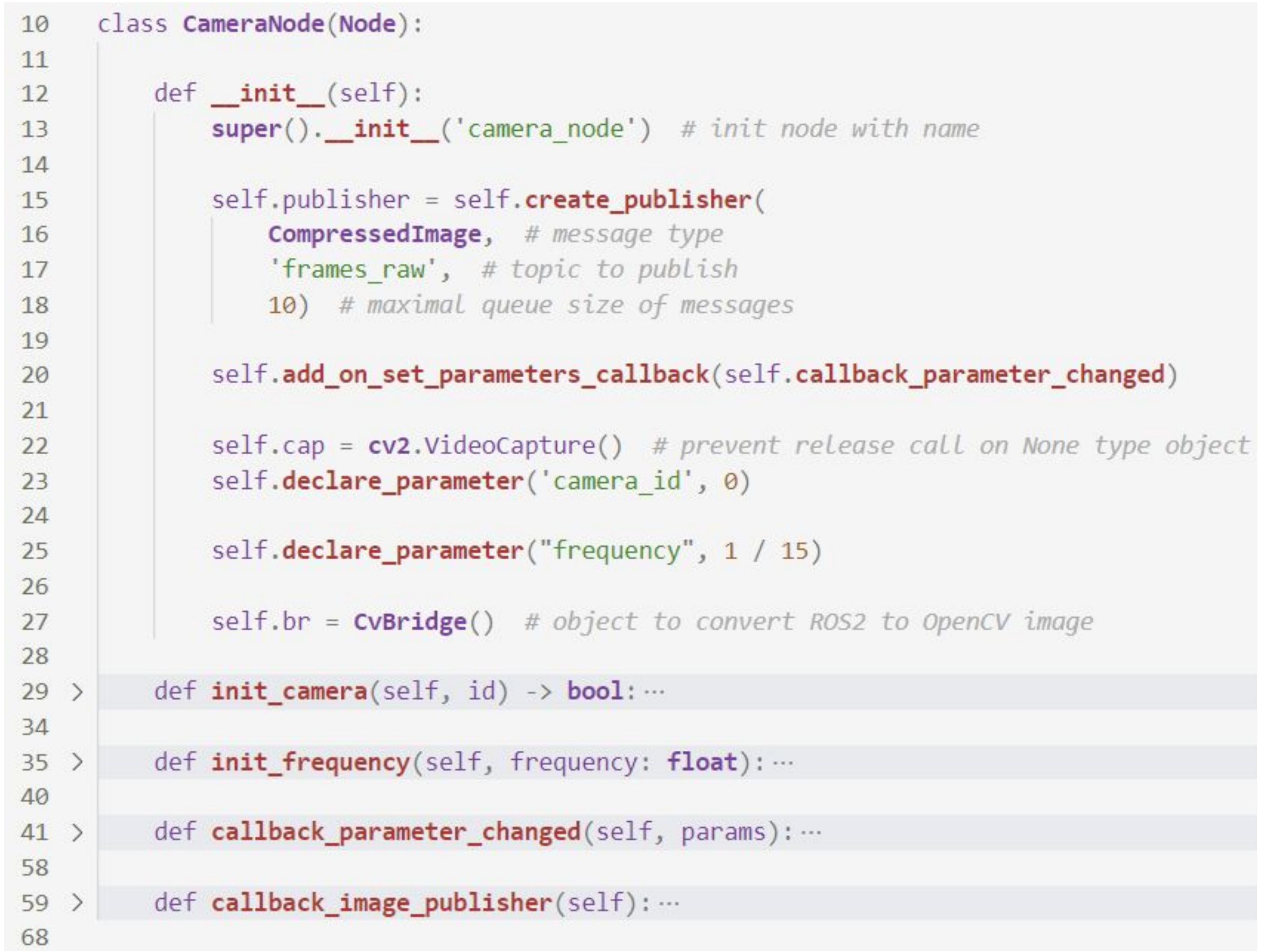


Problems & Challenges

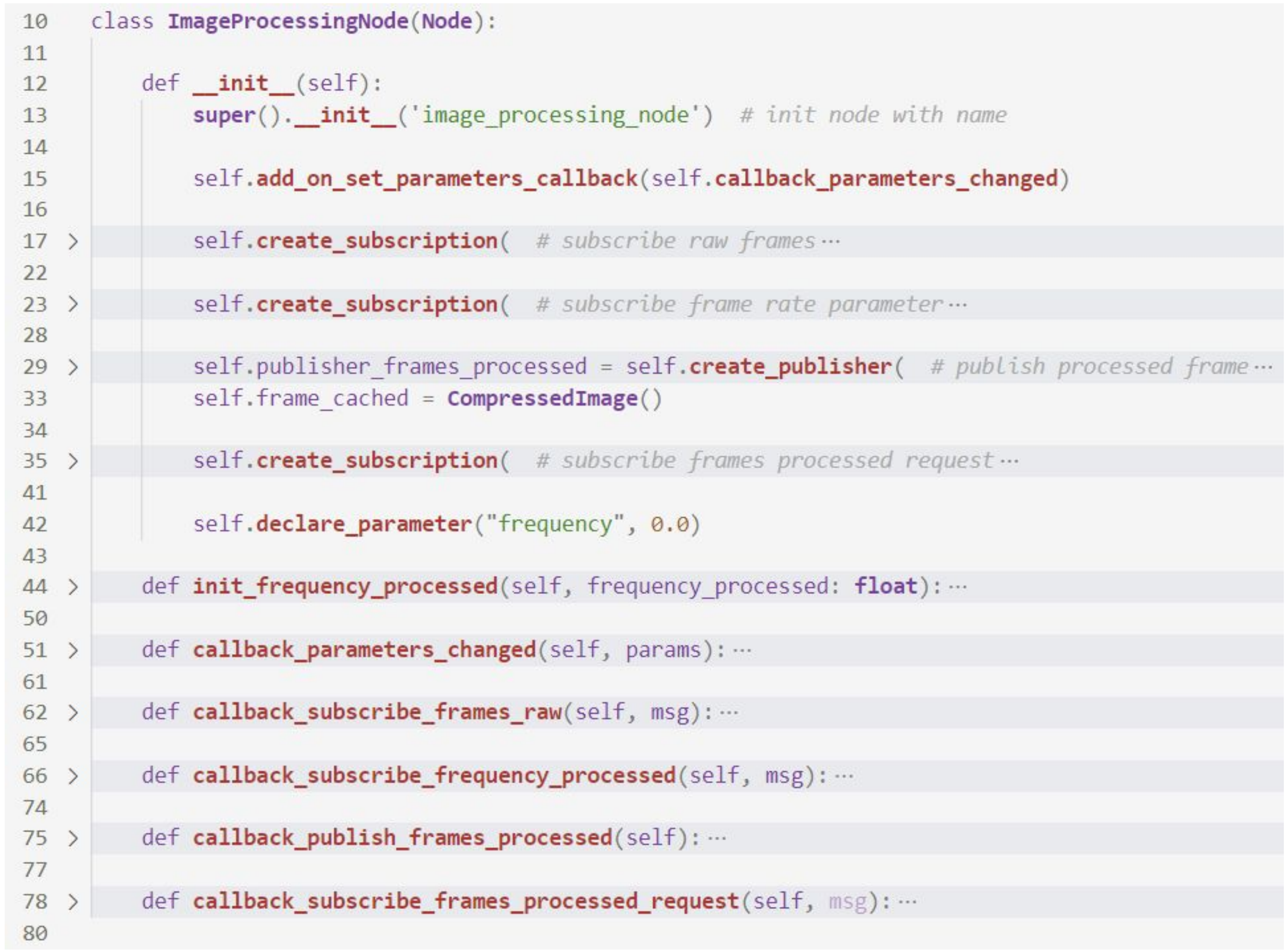
- Computer network communication → trial and error (VMs & Networkadapter)
- Camera Access → OpenCV
- Message Type → sensor_msgs.Image
- OpenCV incompatible with sensor_msgs.Image → cvBridge
- Network Speed → changed Message to sensor_msgs.CompressedImage
- Changing status of one node inside another node → exchanging parameters with publisher/subscriber

Design Decisions

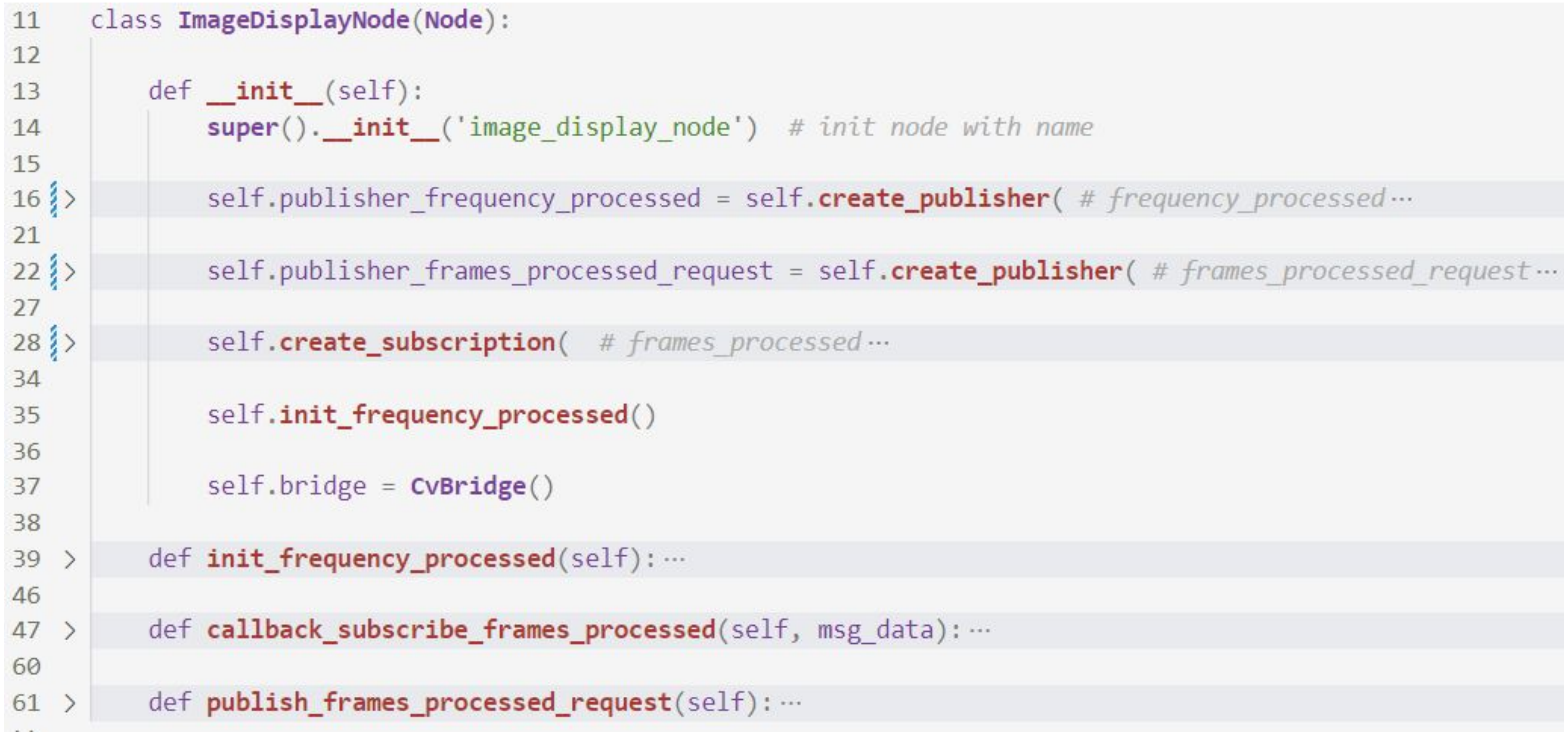
- Publisher / Subscriber instead of Services
- Parameter modifiable at runtime (frame frequency)
- Used standard libraries for solving common problems



```python
class CameraNode(Node):

    def __init__(self):
        super().__init__('camera_node')  # init node with name

        self.publisher = self.create_publisher(
            CompressedImage,  # message type
            'frames_raw',  # topic to publish
            10)  # maximal queue size of messages

        self.add_on_set_parameters_callback(self.callback_parameter_changed)

        self.cap = cv2.VideoCapture()  # prevent release call on None type object
        self.declare_parameter('camera_id', 0)

        self.declare_parameter("frequency", 1 / 15)

        self.br = CvBridge()  # object to convert ROS2 to OpenCV image

    def init_camera(self, id) -> bool: ···

    def init_frequency(self, frequency: float): ···

    def callback_parameter_changed(self, params): ···

    def callback_image_publisher(self): ···

```



```python
class ImageProcessingNode(Node):

    def __init__(self):
        super().__init__('image_processing_node')  # init node with name

        self.add_on_set_parameters_callback(self.callback_parameters_changed)

        self.create_subscription(  # subscribe raw frames···

        self.create_subscription(  # subscribe frame rate parameter···

        self.publisher_frames_processed = self.create_publisher(  # publish processed frame···
        self.frame_cached = CompressedImage()

        self.create_subscription(  # subscribe frames processed request···

        self.declare_parameter("frequency", 0.0)

    def init_frequency_processed(self, frequency_processed: float):···

    def callback_parameters_changed(self, params):···

    def callback_subscribe_frames_raw(self, msg):···

    def callback_subscribe_frequency_processed(self, msg):···

    def callback_publish_frames_processed(self):···

    def callback_subscribe_frames_processed_request(self, msg):···

```



```python
class ImageDisplayNode(Node):

    def __init__(self):
        super().__init__('image_display_node')  # init node with name

        self.publisher_frequency_processed = self.create_publisher( # frequency_processed···

        self.publisher_frames_processed_request = self.create_publisher( # frames_processed_request···

        self.create_subscription(  # frames_processed···

        self.init_frequency_processed()

        self.bridge = CvBridge()

    def init_frequency_processed(self):···

    def callback_subscribe_frames_processed(self, msg_data):···

    def publish_frames_processed_request(self):···
```