Communication infrastructure in ROS

Session 02

Kalana Ratnayake

24/10/2020

Session Plan

Session 01

Robotics and ROS

- Introduction to basic concepts of Robotics
- Introduction to ROS
- When and How to use ROS in robotics

Session 02

Communication infrastructure in ROS

- Getting started with ROS
- Publisher Subscriber (C++)
- Publisher Subscriber (Python)

Session Plan (cont..)

Session 03

Communication infrastructure in ROS (Part 2)

- Standard and Custom message, service and action definitions
- Client Server(C++)
- Client Server (Python)
- Action client Action server (C++)
- Action client Action server (Python)

Session 04

Robot specific infrastructure of ROS

- Introduction to Robot Geometry library
- Introduction to Robot Description language
- Introduction to Gazebo

planned based on slides titled 240AR060 Master's degree in Automatic Control and Robotics – Introduction to ROS by Jan Rosell / Carlos Rosales

Session Plan (cont..)

Session 05

Development tools available in ROS

- rosrun, roslaunch
- rostopic, rosservice
- rqt_graph
- rqt_tf_tree
- Catkin build system

planned based on slides titled 240AR060 Master's degree in Automatic Control and Robotics – Introduction to ROS by Jan Rosell / Carlos Rosales

Communication infrastructure in ROS

Getting started with ROS

Getting started with ROS

- Communication infrastructure of ROS mainly consist of

  1. ROS nodes
  2. ROS messages
  3. ROS core

- When there are many ROS nodes that communicate with each other using ROS messages it is called a ROS application.

Getting started with ROS

- ROS nodes are independent processes that perform computations or connect with hardware devices.
- ROS messages are used for inter-process communication (Data packet)
- ROS core in the process that keeps track of live ROS nodes and the meta data of ROS messages those nodes accept.

Getting started with ROS

ROS Core

1. **ROS Master**

   Negotiates communication connections

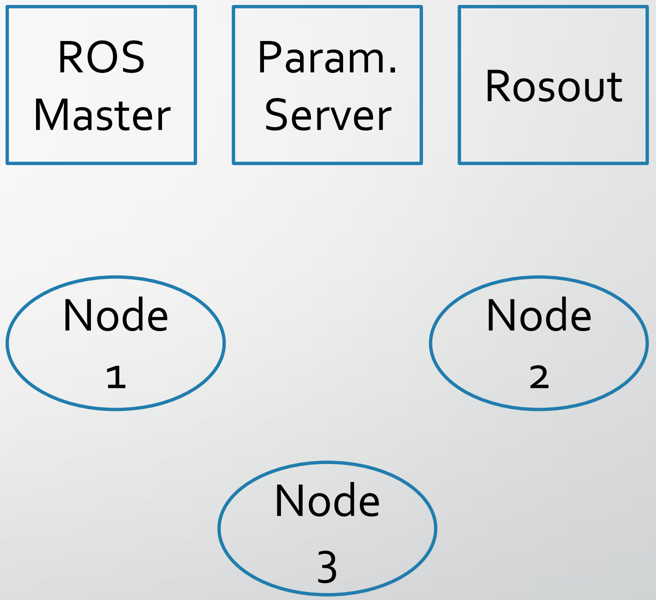
   Registers and looks up names for ROS nodes

2. **Parameter Server**

   Stores persistent configuration parameters and other arbitrary data

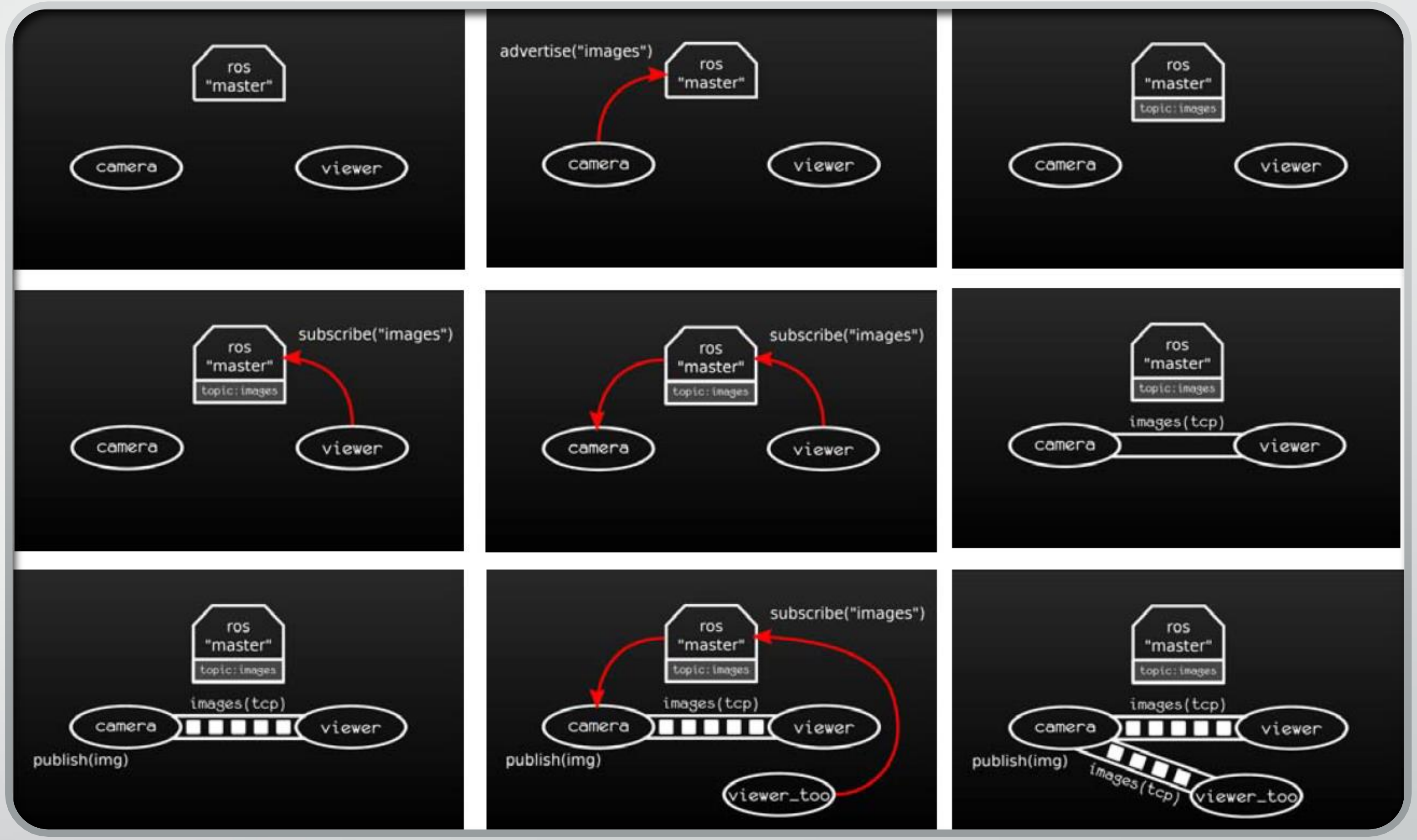
3. **Rosout**

   Essentially a network-based stdout for human-readable messages



Getting started with ROS

- Taken from the slides titled 240AR060 Master's degree in Automatic Control and Robotics – Introduction to ROS by Jan Rosell / Carlos Rosales



Getting started with ROS

- Starting a Workspace

```
mkdir  -p  ~/catkin_ws/src
cd  ~/catkin_ws/
catkin  build
```

- Finally

```
source  devel/setup.bash
```

Getting started with ROS

- Creating a package

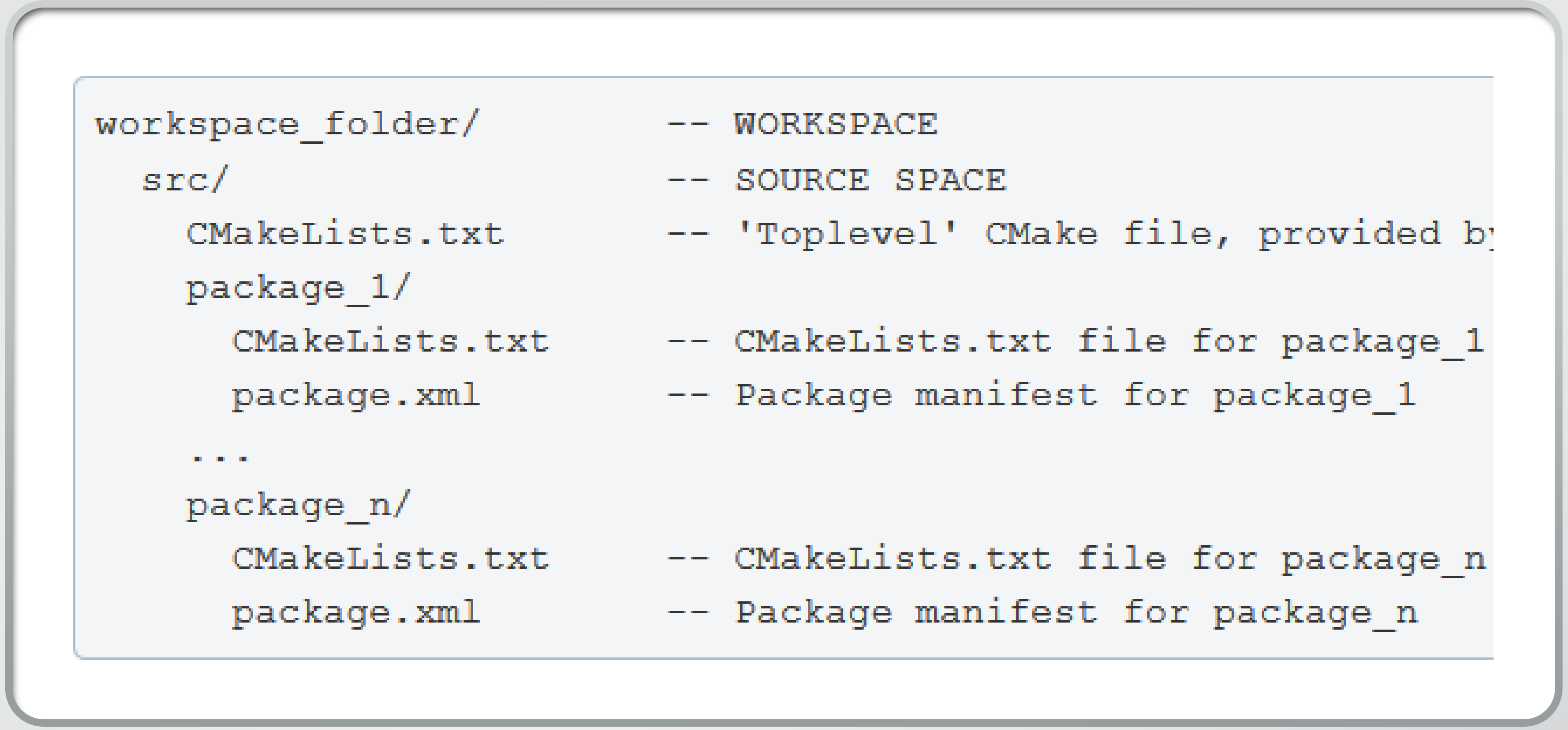
```
catkin_create_pkg  <package_name>  [depend1]  [depend2]  [depend3]
```

Eg: `catkin_create_pkg  session2_tutorials  std_msgs  rospy  roscpp`

Std_msg : basic ROS messages

Rospy : **Python** client library for ROS. Client API

ROSCPP : **C++** client library for ROS. Client API



```
workspace_folder/        -- WORKSPACE
  src/                   -- SOURCE SPACE
    CMakeLists.txt       -- 'Toplevel' CMake file, provided b
    package_1/
      CMakeLists.txt     -- CMakeLists.txt file for package_1
      package.xml        -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt     -- CMakeLists.txt file for package_n
      package.xml        -- Package manifest for package_n
```

Getting started with ROS

Communication infrastructure in ROS

Publisher & Subscriber

Publisher & Subscriber

- Create a workspace for this workshop

```
mkdir  -p  ~/ros_workshop/src
cd  ~/ros_workshop/
catkin  build
```

- Finally

```
source  devel/setup.bash
```

- Create a package

```
catkin_create_pkg  session2_pubsub  std_msgs  rospy  roscpp
```

Publisher (C++)

- Create a file in -> session2_pubsub/src   as publisher.cpp
- Open the   -> Session2/Pub-Sub/C++/publisher.txt
- And copy the content
- Change

  ros::init(argc, argv, "talker");

  TO

  ros::init(argc, argv, “publisher");

Subscriber (C++)

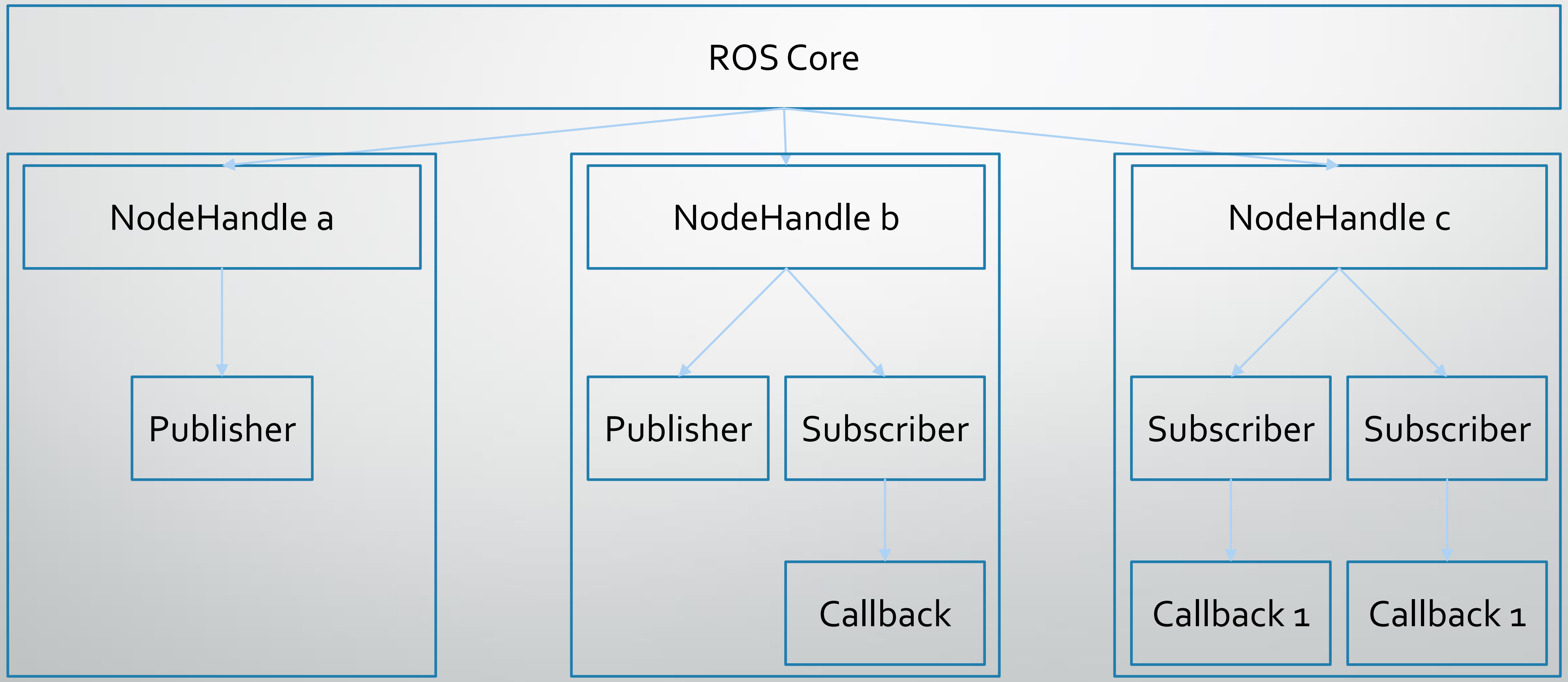
- Create a file in -> session2_pubsub/src   as subscriber.cpp
- Open the   -> Session2/Pub-Sub/C++/subscriber.txt
- And copy the content
- Change

  ros::init(argc, argv, “listener");

  TO

  ros::init(argc, argv, “subscriber");

Structure of ROS nodes



Building the nodes

- Open session2_pubsub/CMakeLists.txt:

```
add_executable(publisher src/ publisher.cpp)
target_link_libraries(publisher ${catkin_LIBRARIES})

add_executable(subscriber src/subscriber.cpp)
target_link_libraries(subscriber ${catkin_LIBRARIES})
```

Building the nodes

- Go to root of the workspace

```
cd  ~/ros_workshop/
catkin  build
```

Running the nodes

- Open a terminal and run

```
roscore
```

- Open a 2nd terminal in the workspace root and run

```
source   devel/setup.bash
rosrun session2_pubsub publisher
```

- Open a 3rd terminal in the workspace root and run

```
source   devel/setup.bash
rosrun session2_pubsub subscriber
```

Publisher (python)

- Create a file in -> session2_pubsub/scripts   as publisher.py
- Open the   -> Session2/Pub-Sub/python/publisher.txt
- And copy the content
- Change

```
rospy.init_node('talker', anonymous=True)
```

TO

```
rospy.init_node(‘publisher', anonymous=True)
```

- Open a terminal and Run

```
chmod +x publisher.py
```

Subscriber (python)

- Create a file in -> session2_pubsub/scripts as subscriber.py
- Open the -> Session2/Pub-Sub/python/subscriber.txt
- And copy the content
- Change

```
rospy.init_node('listener', anonymous=True)
```

  TO

```
rospy.init_node(‘publisher', anonymous=True)
```

- Open a terminal and Run

```
chmod +x subscriber.py
```

Building the nodes

- Open session2_pubsub/CMakeLists.txt:

```
catkin_install_python(PROGRAMS scripts/subscriber.py scripts/publisher.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

- Go to root of the workspace

```
cd  ~/ros_workshop/
catkin  build
```

Running the nodes

- Open a terminal and run

```
roscore
```

- Open a 2nd terminal in the workspace root and run

```
source   devel/setup.bash
rosrun session2_pubsub publisher.py
```

- Open a 3rd terminal in the workspace root and run

```
source   devel/setup.bash
rosrun session2_pubsub subscriber.py
```

Thank you