ROS

Communication infrastructure in ROS

Session 04

Kalana Ratnayake

7/11/2020

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)

Session 04

Robot specific infrastructure of ROS

- Action client Action server (C++)
- Action client Action server (Python)
- Introduction to Gazebo
- Introduction to Robot Description language
- Introduction to Robot Geometry library

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

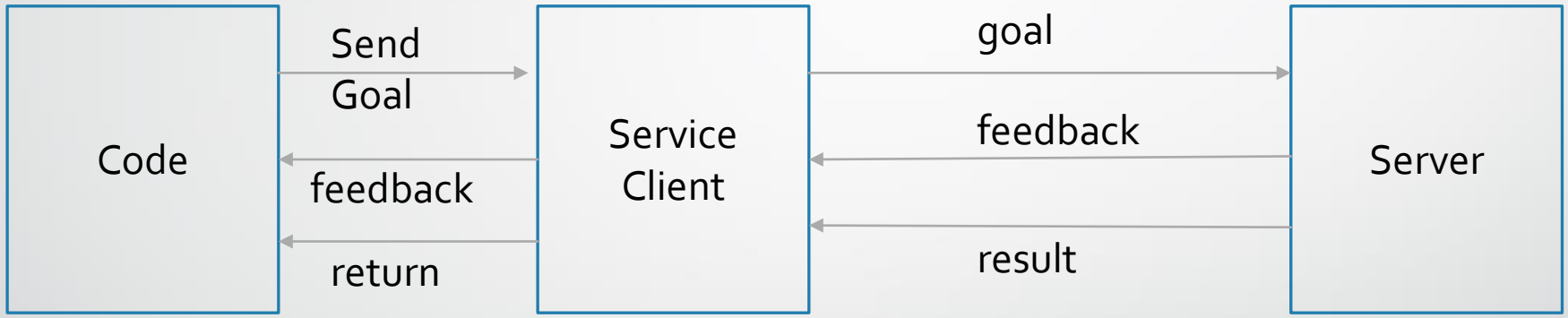
Development tools available in ROS

- 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

Action Server &
Action Client


Code
Send
Goal
feedback
return
Service
Client
goal
feedback
result
Server

Action server & Action Client

- Create a workspace for this workshop

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

- Finally

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

- Create a package

```
catkin_create_pkg session4_action std_msgs rospy roscpp actionlib
```

Action server & Action Client

- Create action folder inside session3_action

  session4_action/action

- Create a custom.action inside

  int32 order

  ---

  int32[] sequence

  ---

  int32[] sequence

- Modify CmakeLists.txt

Action Server (C++)

- Create a file in -> session4_action/src as ac_server .cpp
- Open the -> Action/C++/ ac_server.txt
- And copy the content

Action Client (C++)

- Create a file in -> session4_action/src as ac_client.cpp
- Open the -> Action/C++/ ac_client.txt
- And copy the content

Building the nodes

- Open session4_action/CMakeLists.txt:

```
add_executable(ac_server src/ac_server.cpp)
target_link_libraries(ac_server ${catkin_LIBRARIES})
add_dependencies(ac_server ${session4_action_EXPORTED_TARGETS})

add_executable(ac_client src/ac_client.cpp)
target_link_libraries(ac_client ${catkin_LIBRARIES})
add_dependencies(ac_client ${session4_action_EXPORTED_TARGETS})
```

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 session4_action ac_server
  ```

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

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

  ```
  rosrun session4_action ac_client
  ```

Action Server (python)

- Create a file in -> session4_action/scripts ac_server.py
- Open the -> Action/python/ac_server.txt
- And copy the content
- Open a terminal and Run

  chmod +x ac_server.py

Action Client (python)

- Create a file in -> session4_action/scripts ac_client.py
- Open the -> Action/python/ac_client.txt
- And copy the content
- Open a terminal and Run

  chmod +x ac_client.py

Building the nodes

- Open session4_action/CMakeLists.txt:

```
catkin_install_python(PROGRAMS scripts/ac_client.py scripts/ac_server.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 session4_action ac_client.py
  ```

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

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

  ```
  rosrun session4_action ac_server.py
  ```

Robot Specific infrastructure in ROS

Gazebo

Gazebo

- Open a terminal and run

  ```
  Gazebo
  ```

- This is a Physics Simulator.
- There is gravity, light effects and friction.

Robot Specific infrastructure in ROS

Robot Description Language

Robot Description Language

- Move to the src folder and clone husarion/rosbot_description
- This is the format of robot models in ROS
  - Robot Description (urdf files and drivers)
  - Gazebo worlds
  - Navigation and Controls
- Build the ROS workspace

Robot Description Language

- ROS Description
  - URDF Files
  - Meshes
  - Communication and control drivers

The Unified Robotic Description Format (**URDF**) is an XML file format used in ROS to describe all elements of a robot.

Meshes can be generated from any 3d modelling software.

Robot Description Language

- Gazebo worlds
  - World Files

World files are another type of xml files. They can be directly opened from gazebo and directly modified. Meshes and height maps can also be imported.

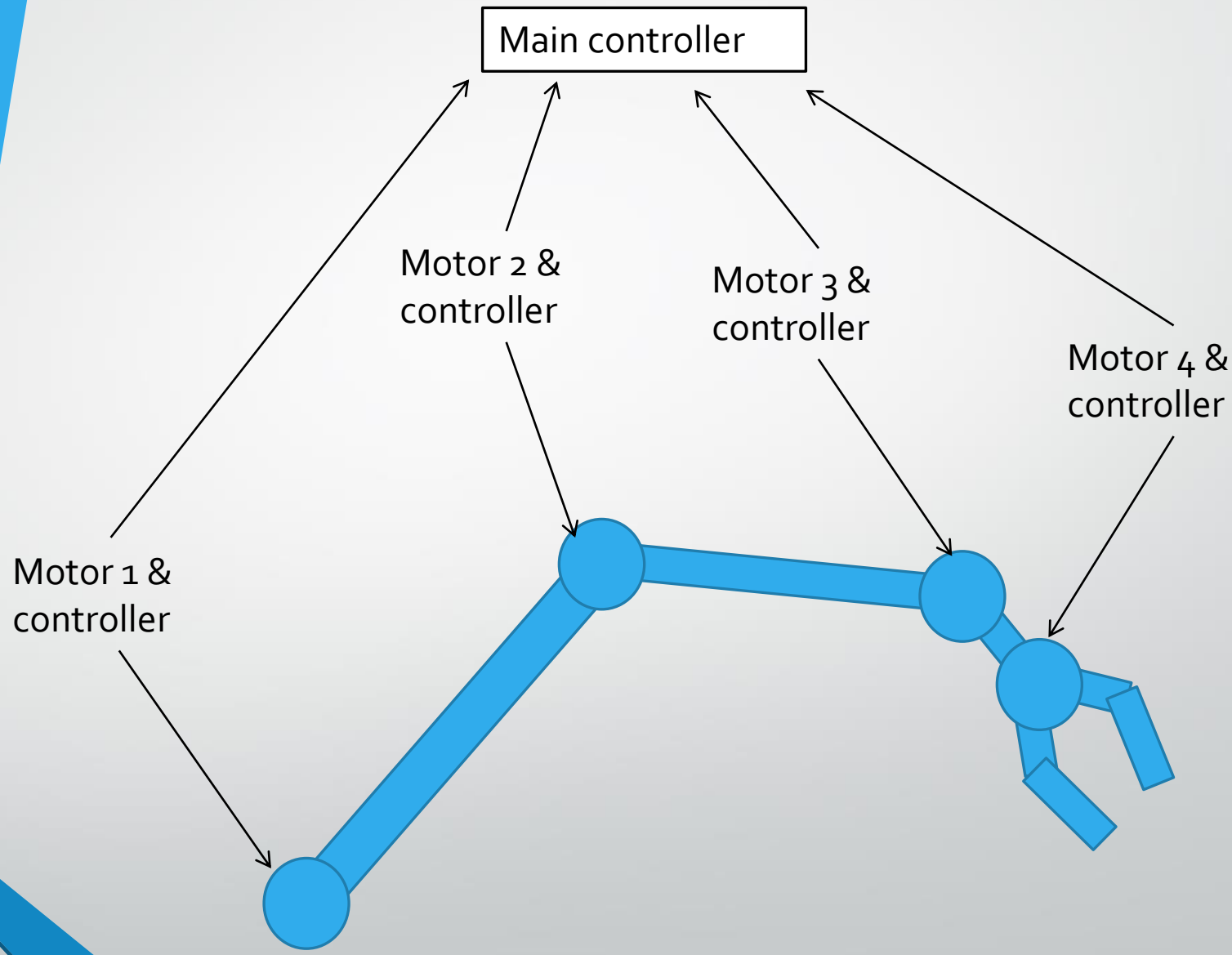
Robot Description Language

- Navigation and Control
  - Teleoperation
  - Camera access
  - Mapping
  - Localization

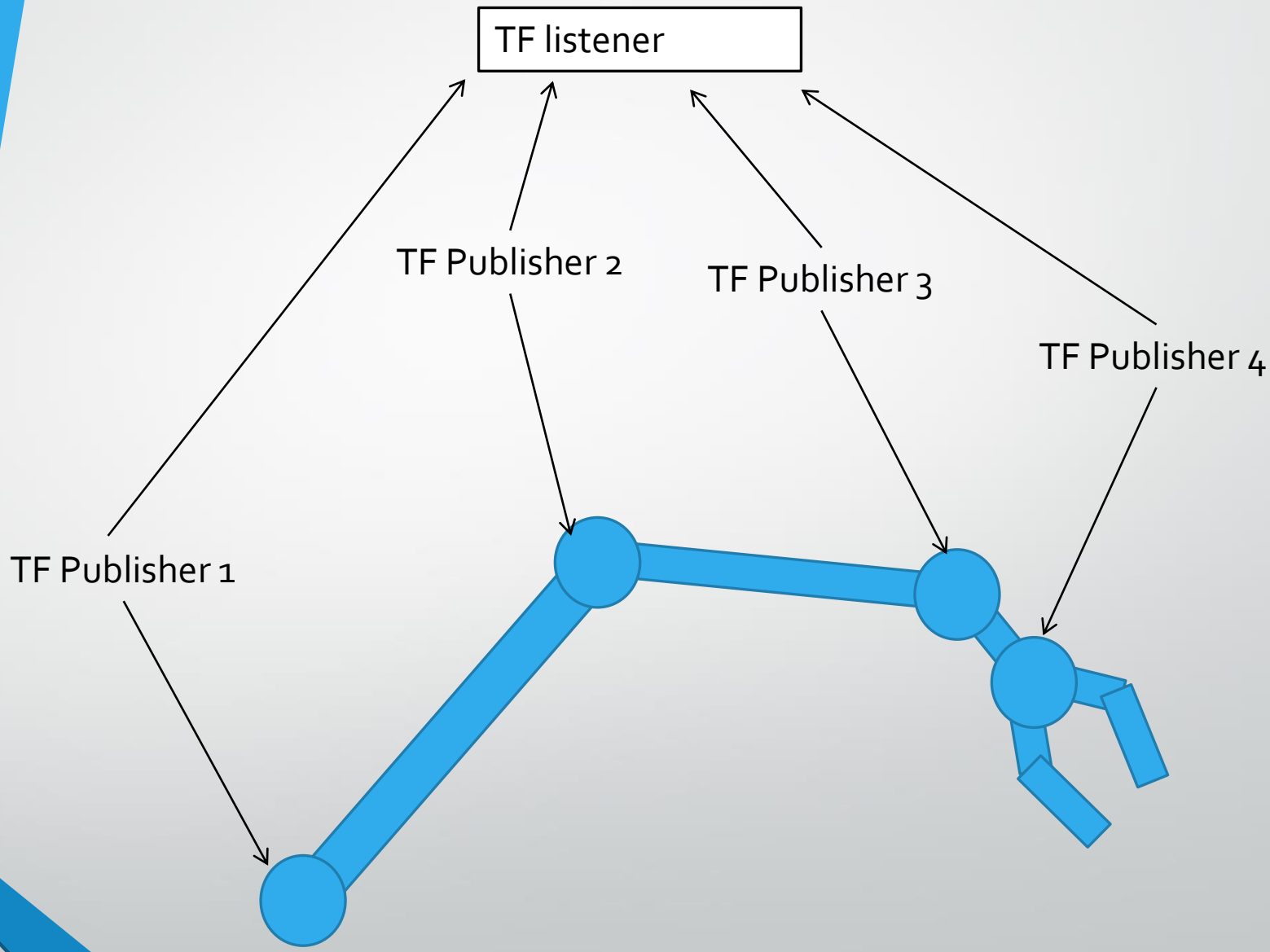
Robot Specific infrastructure in ROS

Robot Geometry Library

Robot Geometry Librabry



Robot Geometry Librabry



Robot Geometry Librabry

- Tf Listener
- Tf Broadcaster

Can be used to publish angle information and calculate kinematics

Thank you