MSAI Robotics course

Seminar 2

File System, First Package, Communication Types

Vladislav Goncharenko

MIPT, 2022

girafe ai

Outline

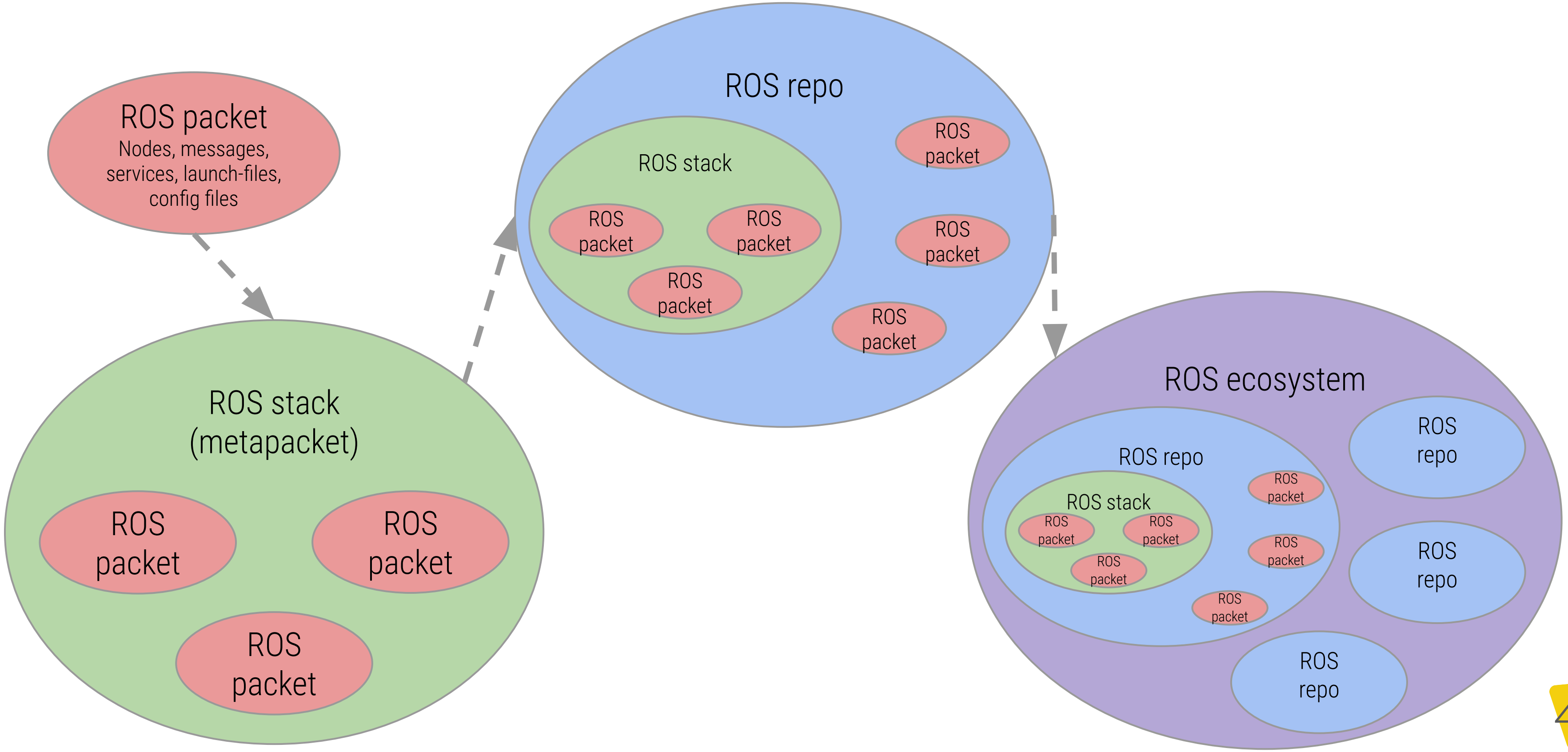
1. What is the ROS file system?
2. How to create a first packet?
   a. CMakeLists.txt
   b. package.xml
3. ROS communication types
4. Writing simple nodes: Publisher and Subscriber
   a. Using standard message types
   b. Creating our own msg-file

What is the ROS file system?

girafe ai

01

BACK TO THE STRUCTURE

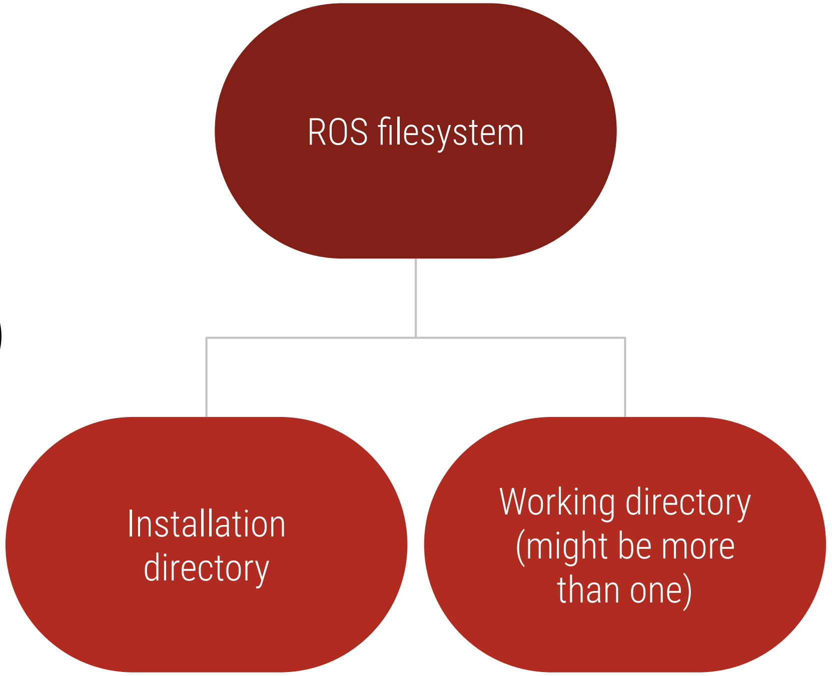


Source: https://web2.qatar.cmu.edu/~gdicaro/16311-Fall17/slides/Lab-1-ROS-Intro.pdf Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

ROS FILE SYSTEM

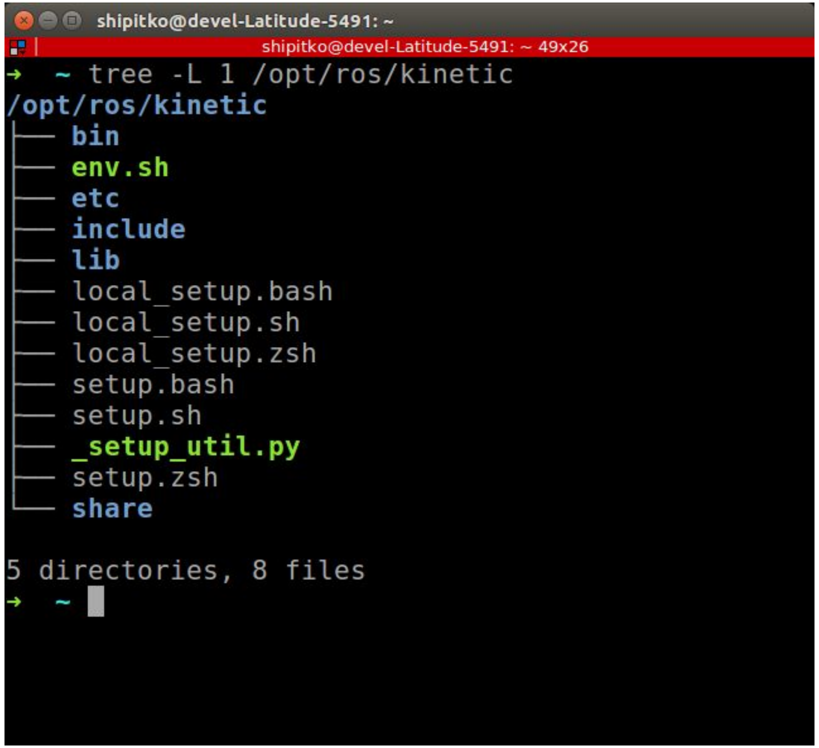
https://www.ros.org/reps/rep-0122.html

- ❑ Contains of two elements:
  - ❑ Installation directory (usually **/opt/ros/<distrib>**)
  - ❑ User’s working directory (workspace)



ROS INSTALLATION DIRECTORY

- Usually **/opt/ros/<distrib>**
  - /bin - executable files
  - /etc - ROS and Catkin config files
  - /include - header files
  - /lib - libraries
  - /share - ROS packets
  - setup.* - scripts for shell environment configuration



```
shipitko@devel-Latitude-5491: ~
shipitko@devel-Latitude-5491: ~ 49x26
➜  ~ tree -L 1 /opt/ros/kinetic
/opt/ros/kinetic
├── bin
├── env.sh
├── etc
├── include
├── lib
├── local_setup.bash
├── local_setup.sh
├── local_setup.zsh
├── setup.bash
├── setup.sh
├── _setup_util.py
├── setup.zsh
└── share

5 directories, 8 files
➜  ~
```

Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

USER’S WORKSPACE FOLDER

http://wiki.ros.org/catkin/workspaces
https://www.ros.org/reps/rep-0128.html

```
user_catkin_workspace_folder/        – Working directory
    src/                             – Source files
        CMakeLists.txt               – Top level CMake file
        package_1/
            CMakeLists.txt           – CMake file of package_1
            package.xml              – Manifest file of package_1
            ...
        package_n/
            CMakeLists.txt           – CMake file package_n
            package.xml              – Manifest file of package_n

    build/                           – Build files

    devel/                           – Compiled executables, header files, libraries, MSG and SRV files

    install/                         – Installation directory
```

WORKSPACE INITIALISATION

http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

❏ Creating and initilisation of the new workspace:

```
mkdir -p ~/my_ros_ws/src
cd ~/my_ros_ws
catkin_make
```

Makes parent directories as needed

Workspace name can be set arbitrary
Usually – ***catkin_ws***

❏ After initialization (and before every run of ROS in a new terminal) you have to use this command:

```
source ~/my_ros_ws/devel/setup.bash
```

It sets environment variables and adds workspace to the ***$ROS_PACKAGE_PATH***

❏ To ensure ROS ability to start your packages just print it's value:

```
echo $ROS_PACKAGE_PATH
```

It has to contain your workspace ***~/my_ros_ws/src***

How to create a first packet?

girafe
ai

02

LET’S CREATE OUR FIRST PACKET

http://wiki.ros.org/ROS/Tutorials/CreatingPackage

❏ First of all we have to change directory to the src:

```
cd ~/my_ros_ws/src
```

❏ We will use ***catkin_create_pkg*** to automatically create a packet. It also gets a list of dependencies as a parameter

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

```
catkin_create_pkg test_package rospy std_msgs
```

❏ You can create packets without this tool. Just create a packet directory and add **CMakeLists.txt** and **package.xml** there.

PACKET STRUCTURE

- ❑ .../<catkin workspace>/src/<package name>
  - ❑ **/include** – header (**.h**, **.hpp**) file
  - ❑ **/node** (**/scripts**) – python scripts
  - ❑ **/launch** – launch file (**.launch**), used by roslaunch
  - ❑ **/msg** – message files (**.msg**)
  - ❑ **/src** – source files
  - ❑ **/srv** – service files (**.srv**)
  - ❑ **/action –** action files (**.action**)
  - ❑ **CMakeLists.txt** – build configuration files
  - ❑ **package.xml** – manifest file
  - ❑ (optional) **setup.py** – installation script for python-modules

PACKAGE.XML

http://wiki.ros.org/catkin/package.xml
https://www.ros.org/reps/rep-0140.html

- ❏ Package.xmldefines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages
- ❏ Packet definitions on wiki.ros.org are generated from these files

Minimal example of package.xml

```
<package format="2">
    <name>foo_core</name>
    <version>1.2.4</version>
    <description>
        This package provides foo
        capability.
    </description>
    <maintainer
    email="ivana@osrf.org">Ivana
    Bildbotz</maintainer>
    <license>BSD</license>
    <buildtool_depend>catkin
    </buildtool_depend>
</package>
```

CATKIN

http://wiki.ros.org/catkin/conceptual_overview

- ❑ **`catkin`** – build automation system created for ROS. It is responsible for generating 'targets' from raw source code that can be used by an end user.
- ❑ **`catkin`** combines CMake macros and Python scripts to provide some functionality on top of CMake's normal workflow.

WORKSPACE INITIALIZATION

http://wiki.ros.org/catkin/CMakeLists.txt

- **Important!** In CMakeLists.txt instruction order matters.
- CMakeLists.txt instructions may vary from one package to another but all of them have to comply following template:
  1. Required CMake Version (**cmake_minimum_required()**)
  2. Package name (**project()**)
  3. Find other CMake/Catkin packages needed for build (**find_package()**)
  4. Enable Python module support (**catkin_python_setup()**)
  5. Message/Service/Action Generators (**add_message_files()**, **add_service_files()**, **add_action_files()**)
  6. Invoke message/service/action generation (**generate_messages()**)
  7. Specify package build info export (**catkin_package()**)
  8. Libraries/Executables to build (**add_library()**/**add_executable()**/**target_link_libraries()**)
  9. Tests to build (**catkin_add_gtest()**)
  10. Install rules (**install()**)

CMAKELISTS.TXT

http://wiki.ros.org/catkin/CMakeLists.txt

https://makefiletutorial.com/

Minimal CMake version

Project(packet) name. Has to be the same as in packege.xml. Saved in ${PROJECT_NAME}

Searching for dependencies. All of the ROS packages depends on catkin

Searching for ROS independent libraries

```
cmake_minimum_required (VERSION 2.8.3)
project (my_first_ros_pkg)
find_package (catkin REQUIRED COMPONENTS
      Roscpp
      std_msgs
)
find_package (Boost REQUIRED COMPONENTS system)
catkin_python_setup ()
add_message_files (
      FILES
      Message1.msg
      Message2.msg
)
add_service_files (
      FILES
      Service1.srv
      Service2.srv
)
generate_messages (
      DEPENDENCIES
      std_msgs
)
catkin_package (
      INCLUDE_DIRS include
      LIBRARIES my_first_ros_pkg
      CATKIN_DEPENDS roscpp std_msgs
message_runtime
      DEPENDS system_lib
)
```

FIND_PACKAGE()

http://wiki.ros.org/catkin/CMakeLists.txt

- ❏ Finding a packet using `find_package()` results a creation of several CMake variables which can be used later in CMake file.
- ❏ Variable names match to the template `<PACKAGE NAME>_<PROPERTY>` :
  - ❏ `<NAME>_FOUND` – sets `True`, if package has been found
  - ❏ `<NAME>_INCLUDE_DIRS` or `<NAME>_INCLUDES` – path to include directory of the packet
  - ❏ `<NAME>_LIBRARIES` or `<NAME>_LIBS` – exported libraries
- ❏ Why all ROS packets added as a `catkin components` ? For convenience. In this case all of these packages have corresponding `catkin` related environment variables (ex. `catkin_INCLUDE_DIRS`)

CMAKELISTS.TXT

http://wiki.ros.org/catkin/CMakeLists.txt

Use it if your package exports python modules. Requires packet to have setup.py. Has to be invoked before generate_messages() и catkin_package().

Adds user defined messages, services and action files.
Invoke before catkin_package()

Specifies catkin-specific information to the build system which in turn is used to generate pkg-config and CMake files. ***Must be called before*** declaring any targets with add_library() or add_executable().

```
cmake_minimum_required (VERSION 2.8.3)
project (my_first_ros_pkg)
find_package (catkin REQUIRED COMPONENTS
      Roscpp
      std_msgs
)
find_package (Boost REQUIRED COMPONENTS system)
catkin_python_setup ()
add_message_files (
      FILES
      Message1.msg
      Message2.msg
)
add_service_files (
      FILES
      Service1.srv
      Service2.srv
)
generate_messages (
      DEPENDENCIES
      std_msgs
)
catkin_package (
      INCLUDE_DIRS include
      LIBRARIES my_first_ros_pkg
      CATKIN_DEPENDS roscpp std_msgs
message_runtime
      DEPENDS system_lib
)
```

SETUP.PY

http://docs.ros.org/api/catkin/html/user_guide/setup_dot_py.html

- ❑ `setup.py` has to be used if ROS package contains scripts and modules, which will be installed to the system (ex. They will be used in other packets). Python uses libraries `distutils` и `setuputils` for that.
- ❑ If `CMakeLists.txt` contains `catkin_python_setup()` catkin searches the root of the workspace for `setup.py` and runs it. Also `setup.py` can get an access to information in `CMakeLists.txt`.
- ❑ Use `generate_distutils_setup()` function to access data in `package.xml`.

```
from setuptools import setup
from catkin_pkg.python_setup
import
generate_distutils_setup

d = generate_distutils_setup(
    packages=['mypkg'],
    scripts=['bin/myscript'],
    package_dir={'': 'src'}
)

setup(**d)
```

Message generation: frequent problems

http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

- If you add message generation to your packet:
  - Don't forget to update dependencies in package.xml

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

  - Add `message_generation` to the list of the required components

```
find_package(catkin REQUIRED COMPONENTS
   roscpp
   rospy
   std_msgs
   message_generation
)
```

  - Add `message_runtime` dependency

```
catkin_package(
  ...
  CATKIN_DEPENDS message_runtime ...
  ...
)
```

  - Add message files

```
add_message_files(
  FILES
  Num.msg
)
```

  - Add `generate_messages` command

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

CMAKELISTS.TXT

http://wiki.ros.org/catkin/CMakeLists.txt

Creating targets, adding dependencies to create proper order of generation of messages/services and linking target to the libraries.

```
add_executable (my_first_ros_pkg_node src/main.cpp)
add_dependencies (my_first_ros_pkg_node
          ${${PROJECT_NAME}_EXPORTED_TARGETS}
          ${catkin_EXPORTED_TARGETS}
)
target_link_libraries (my_first_ros_pkg_node
          ${catkin_LIBRARIES}
)
```

(Optional) Adding unit tests

```
if(CATKIN_ENABLE_TESTING)
          catkin_add_gtest (myUnitTest test/utest.cpp)
endif()
```

(Optional) Installation of the package and executable python-scripts

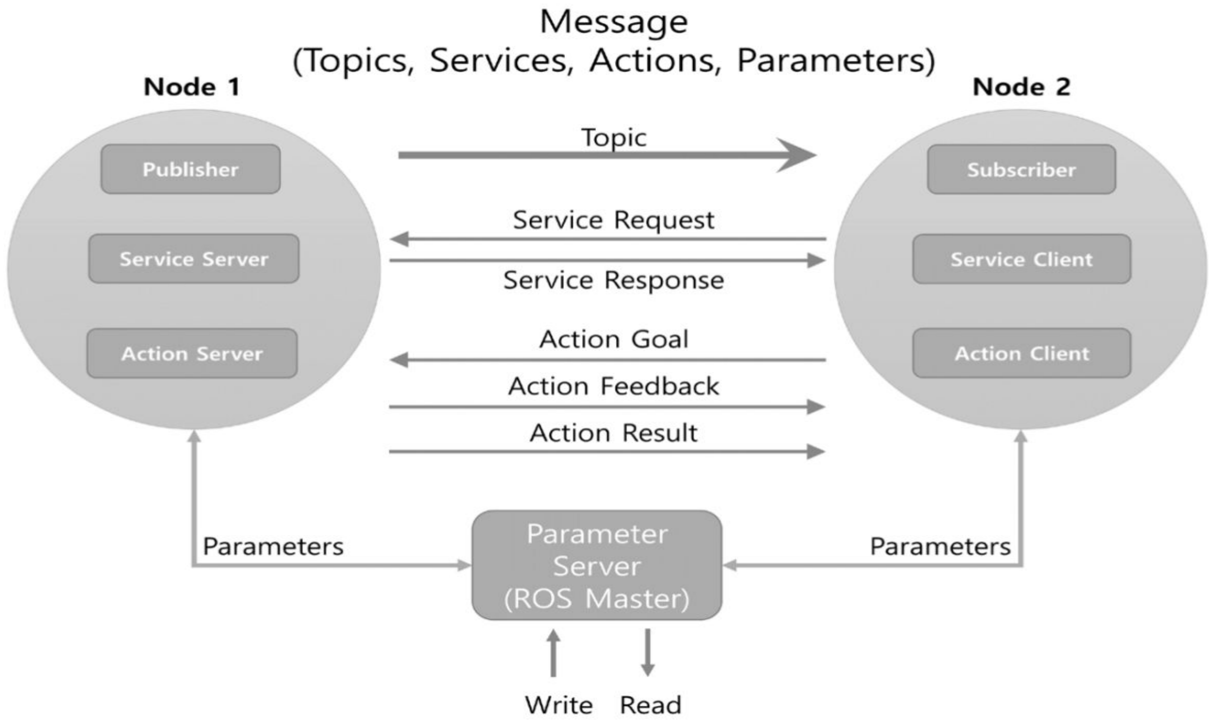
```
install (TARGETS ${PROJECT_NAME}
          ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
          LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
          RUNTIME DESTINATION ${CATKIN_GLOBAL_BIN_DESTINATION}
)


catkin_install_python (PROGRAMS scripts/myscript
          DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

ROS communication types

girafe
ai

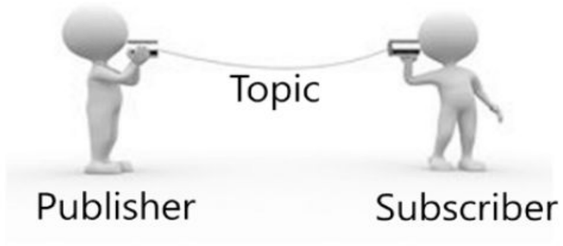
03

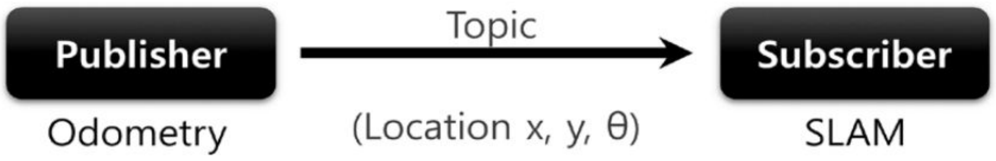
ROS communication types

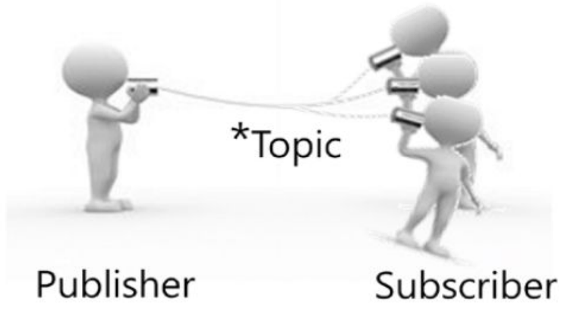


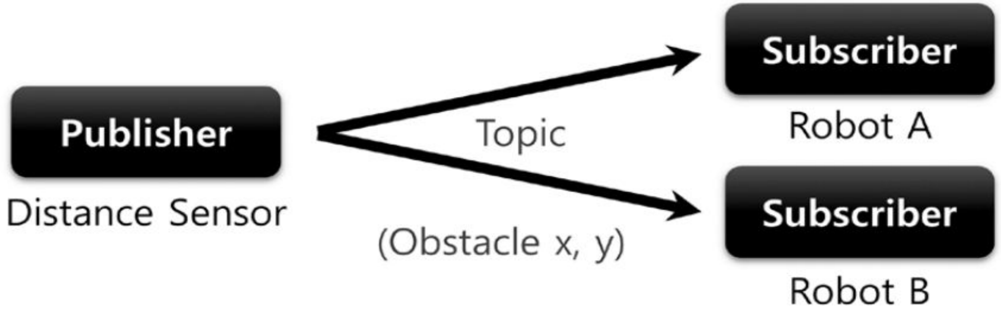
Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

Topics





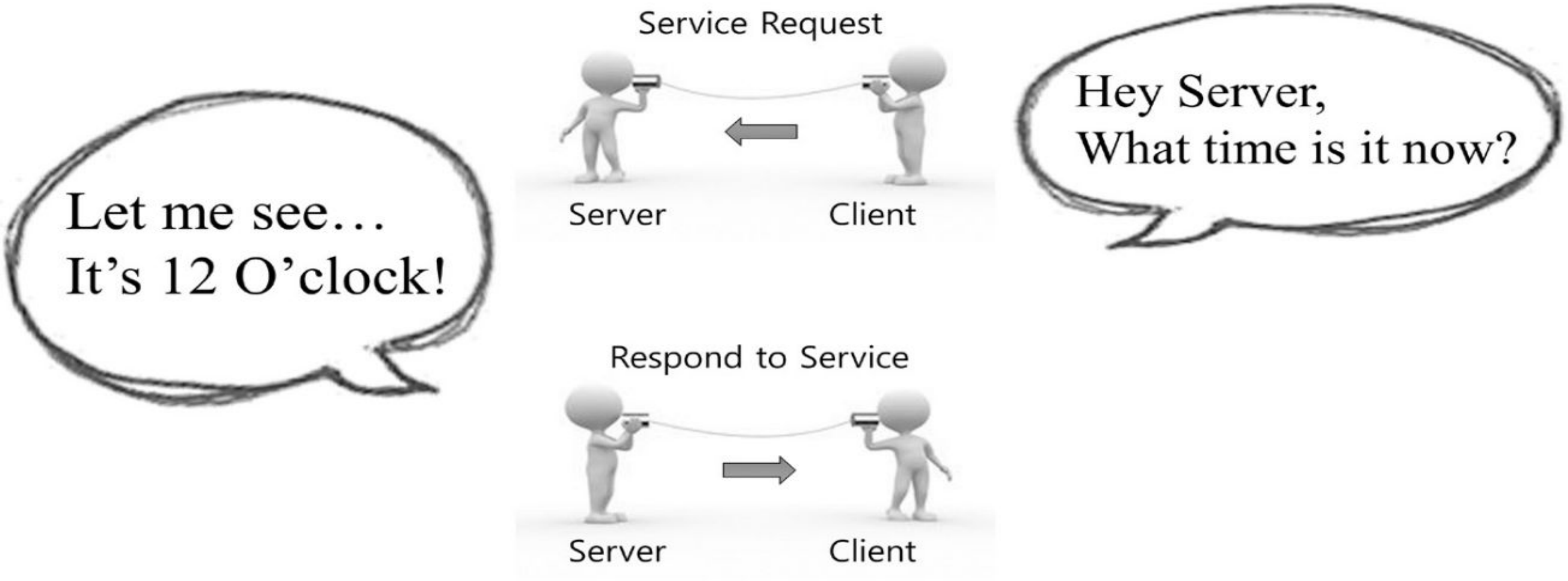




Topics allow as one-to-one communication so as N-to-N

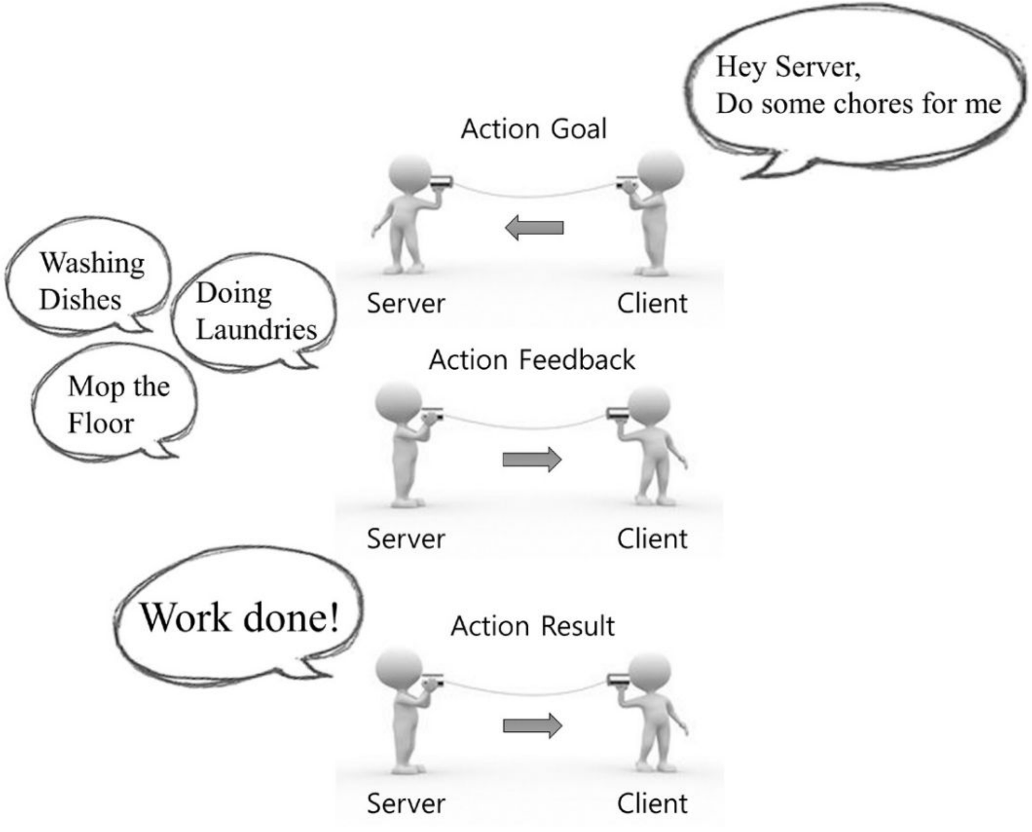
Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

Services



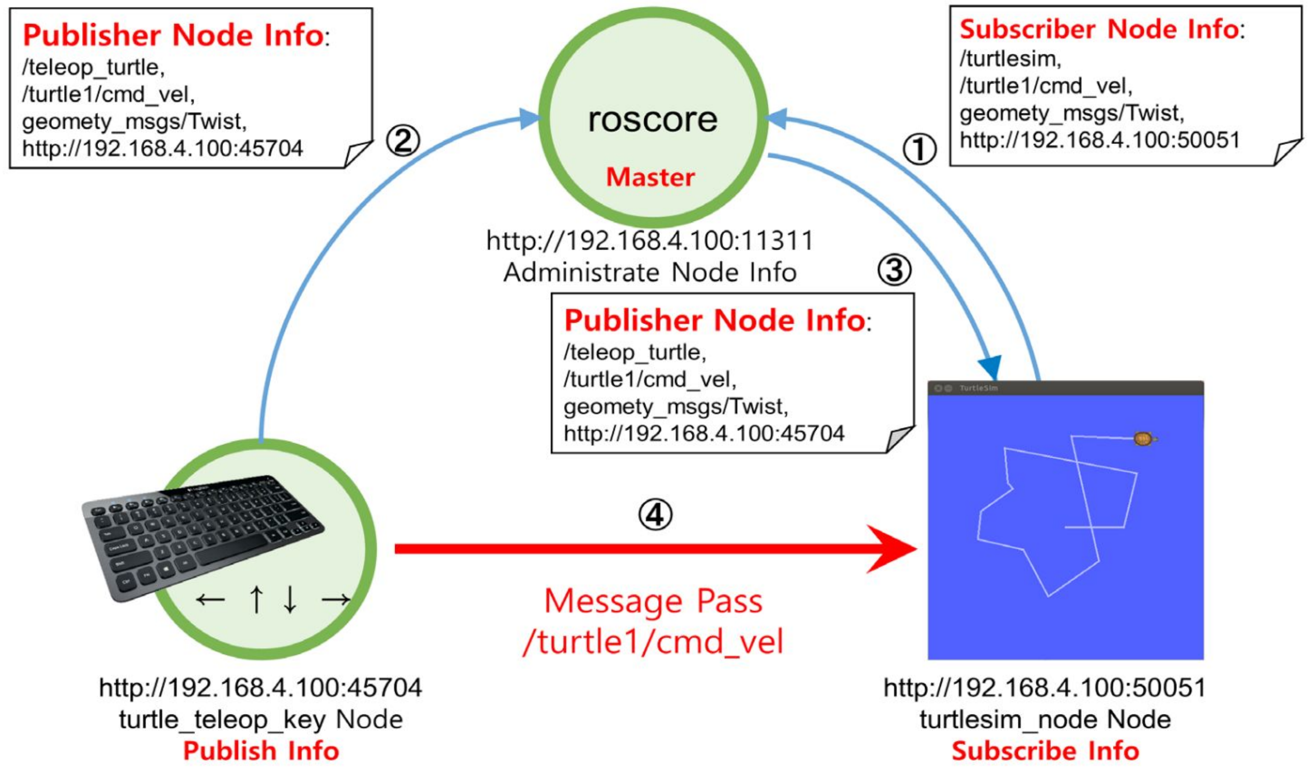
Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

Actions



Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

ROS communication



Source: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim

TYPES OF COMMUNICATION

Type	Features	Use cases
Topic	Asynchronous, unidirectional	Continuous data streams
Service	Synchronous, bidirectional	Request-reply with a fast response
Action	Asynchronous, bidirectional	If Service is too long to response, or if you need a feedback in process

MESSAGES

http://wiki.ros.org/msg

- ROS uses simple language to define messages. From this definitions catking automatically generate code definitions for several target program languages (python, C++, lisp)
- User defined messages usually saved in /msg folder of the packet and has an .msg file extension
- Messages can have two parts:
  - **Data field** (required) – defines fields of message in a form “type + name”
  - **Constants** – helper constants for data interpretation (as enum in C++)

MESSAGES

http://wiki.ros.org/msg

- **Data type** – can be built-in type (ex. `float64`), another message type (`geometry_msgs/Quaternion`), fixed or dynamic size array (`float64[]` или `float64[9] orientation_covariance`), special type `Header` (see `std_msgs/Header`)
- **Constants** – only built-in type (except of `time` and `duration`)

sensor_msgs/Imu

```
Header header
geometry_msgs/Quaternion orientation
float64[9] orientation_covariance
# Row major about x, y, z axes

geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance
# Row major about x, y, z axes

geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance
# Row major x, y z

# Constants example
int32 X=123
string FOO=foo
```

Writing simple nodes: Publisher and Subscriber

girafe ai

04

ROSPY API

http://wiki.ros.org/rospy

- Import of client ROS library in `python`

```
import rospy
```

- Import message of type Float32 from std_msgs packet. **Warning!** When you import messages don't forget an .msg suffix in the packet name

```
from std_msgs.msg import Float32
from <package>.msg import <Message>
```

- Registering subscription to the specific topic providing its name, message type and processing function (callback)

```
rospy.Subscriber("signal", Float32, signal_callback)
rospy.Subscriber(name, data_class, callback=None, callback_args=None,
queue_size=None, buff_size=65536, tcp_nodelay=False)
```

ROSPY API

http://wiki.ros.org/rospy

- ❏ Registering publication (advertisement) to the specific topic providing its name, message type and processing queue length

```
rospy.Publisher("filtered_signal", Float32, queue_size=10)
rospy.Publisher(name, data_class, subscriber_listener=None, tcp_nodelay=False,
latch=False, headers=None, queue_size=None)
```

- ❏ Logging. There are several levels of logging: `.logdebug, .logwarn, .logerr, .logfatal`

```
rospy.loginfo("I've got {}".format(signal.data))
```

- ❏ Initialization of the node with a specific name

```
rospy.init_node("signal_filter")
rospy.init_node(name, argv=None, anonymous=False, log_level=2,
disable_rostime=False, disable_rosout=False, disable_signals=False)
```

HOW TO SAVE CHANGES IN CONTAINER

https://docs.docker.com/engine/reference/commandline/commit/
https://docs.docker.com/storage/volumes/

- There are several ways to save modified data in container:
  - In order to create new version of the image containing your changes use

    ```
    docker commit <container-id> USER_NAME/IMAGE_NAME
    ```

  - To copy data from container to the host system:

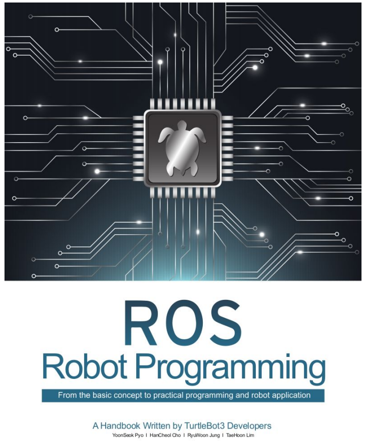
    ```
    docker cp CONTAINER:SRC_PATH DEST_PATH
    ```

  - Mount directory on the host to the container file system:

    ```
    sudo docker run -v [-- volume] HOST_FOLDER:CONTAINER_VOLUME_NAME
    ```

ADDITIONAL RESOURCES

1. Book: ROS Robot Programming. YoonSeok Pyo, HanCheol Cho, Ryu Woon Jung, TaeHoon Lim
2. ROS Officiel Tutorials
3. Clearpath Robotics ROS Tutorial
4. The history of ROS creation


ROS
Robot Programming
From the basic concept to practical programming and robot application
A Handbook Written by TurtleBot3 Developers
YoonSeok Pyo I HanCheol Cho I RyuWoon Jung I TaeHoon Lim

Thanks for attention!

Questions? Additions? Welcome!

girafe
ai