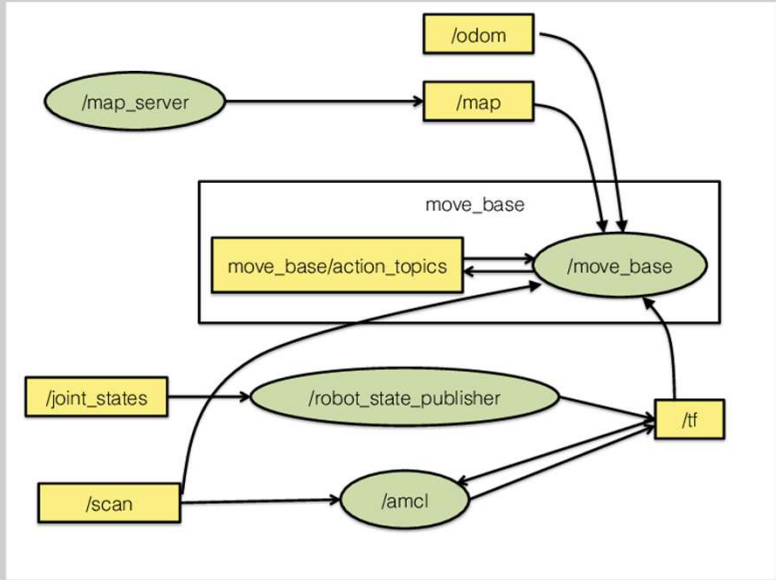
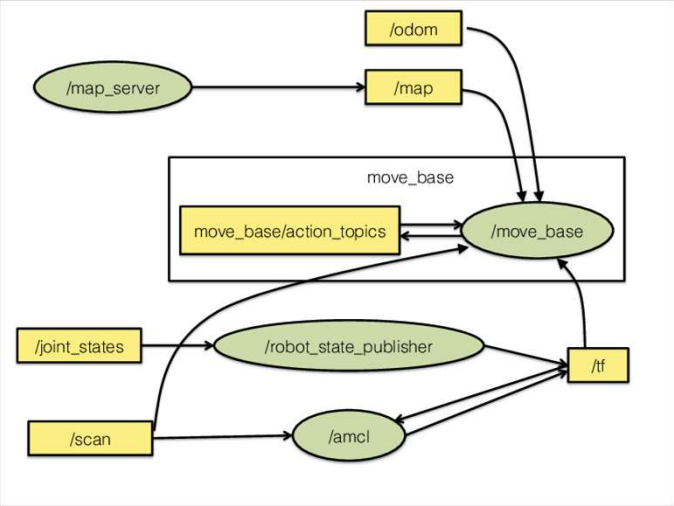


What are the different ROS Concepts?

Discussing the important ROS concepts

List of Topics

- ROS Computation Graph
- Important ROS terminologies
  - ROS Nodes
  - ROS Master
  - ROS Messages
  - ROS Topics
  - ROS Parameter Server
  - ROS Services
  - ROS Action
  - ROS Bags
- Secret of ROS inter-process communication

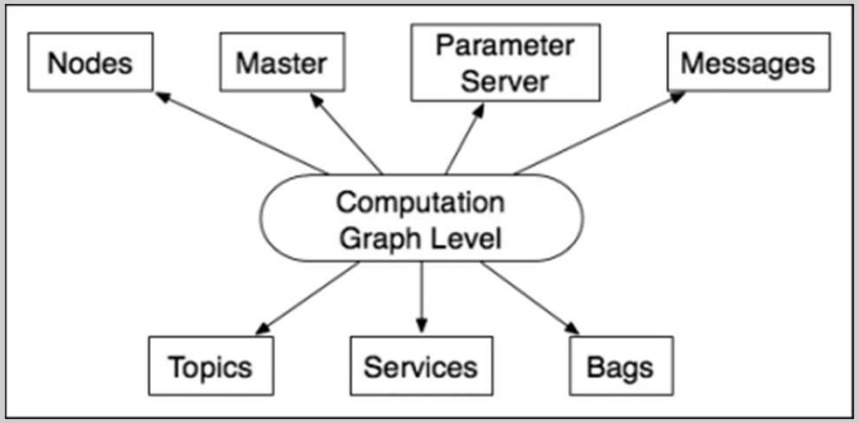


What is a ROS Computation Graph?

What is a ROS Computation Graph?

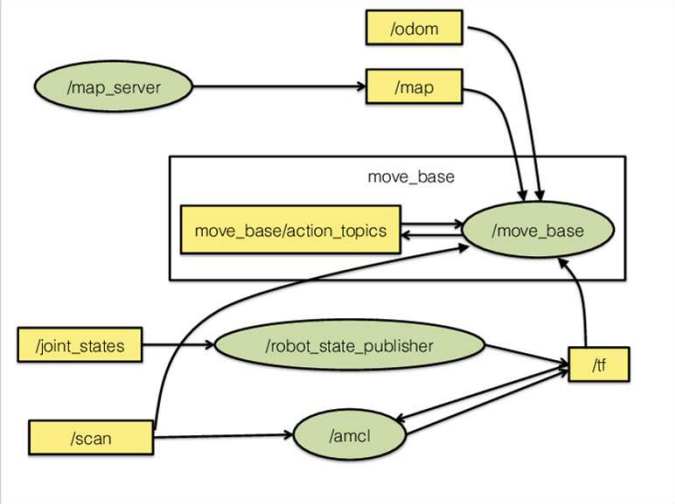
- **ROS Computation Graph:**
  - *It is the network of ROS process which is doing the computation in ROS together by connecting peer-to-peer*

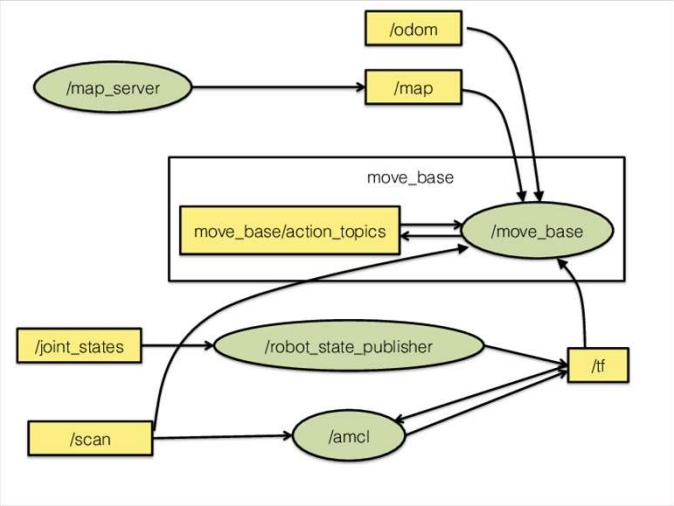
  - Various concept in the graph are
    - **Nodes**
    - **Master**
    - **Parameter Server**
    - **Messages**
    - **Topics**
    - **Services**
    - **Bags**



What is a ROS Computation Graph?

- **ROS Computation Graph:**
  - The ROS graph concepts are implemented in ***ros_comm*** repository http://wiki.ros.org/ros_comm
  - E.g.: ROS graph diagram



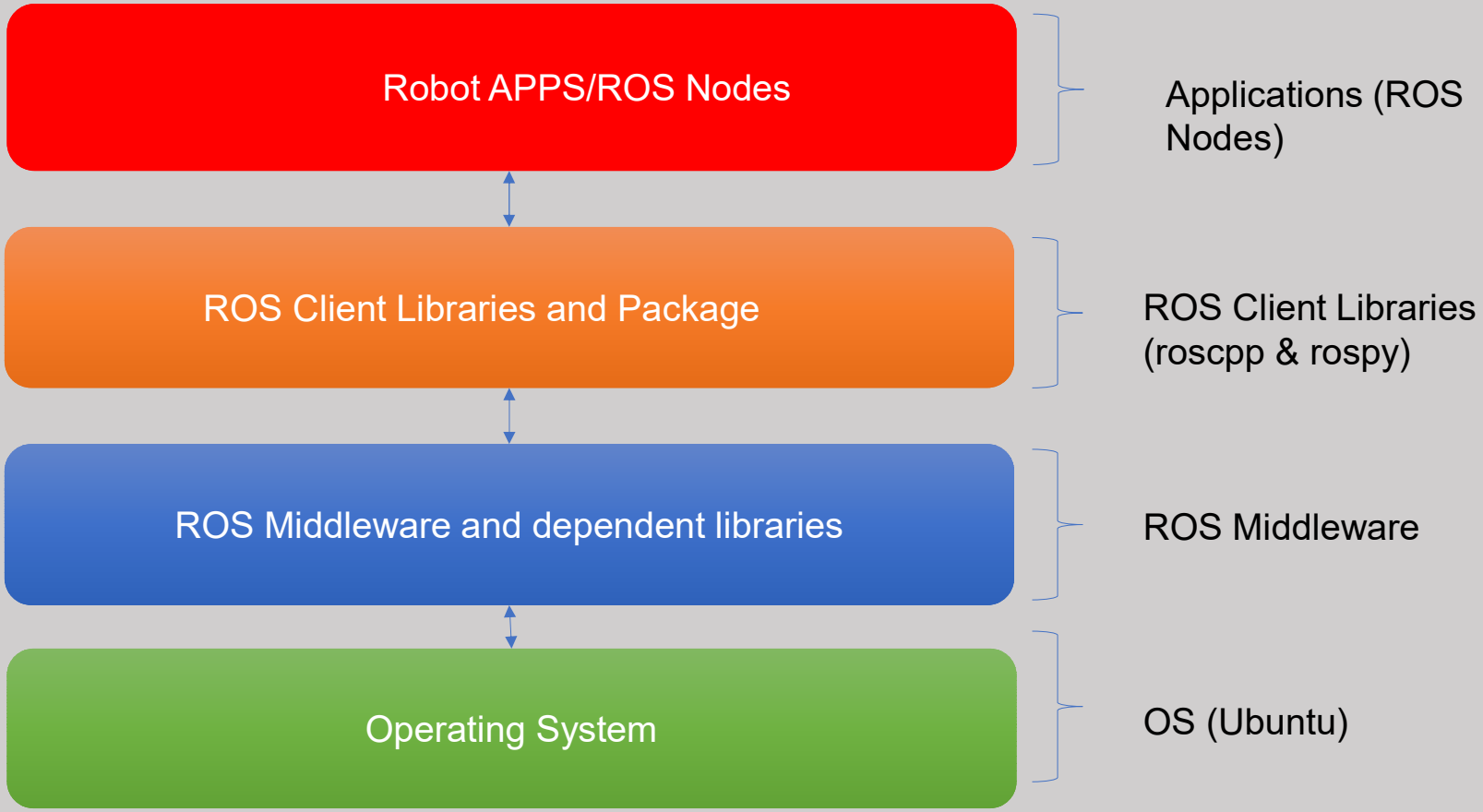


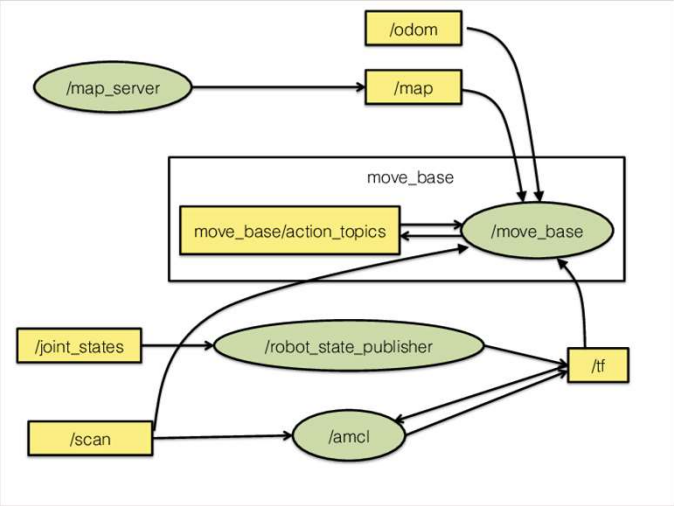
What is a ROS Node?

What is a ROS Node?

- A Node is a executable program which perform computation
- Nodes are ROS program created using ROS Client libraries like **roscpp** & **rospy**
- Nodes are the atomic computing unit of ROS framework
- The Nodes can communicate each other nodes using ROS concepts like Topics, services etc.

ROS Nodes

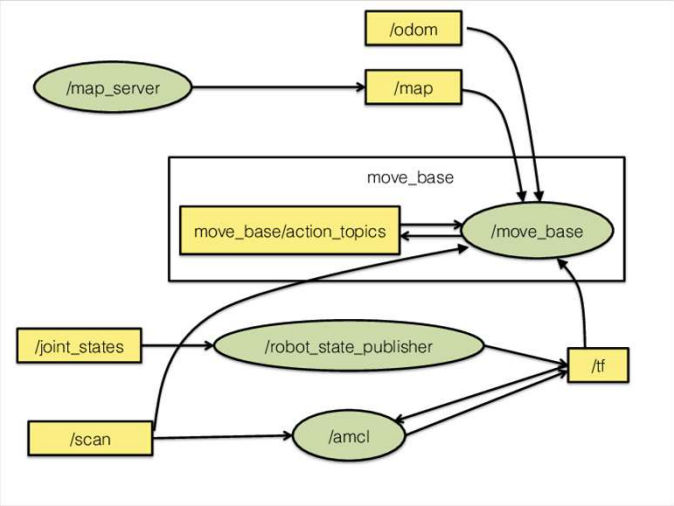

Robot APPS/ROS Nodes
ROS Client Libraries and Package
ROS Middleware and dependent libraries
Operating System
Applications (ROS Nodes)
ROS Client Libraries (roscpp & rospy)
ROS Middleware
OS (Ubuntu)



What is a ROS Master?

What is a ROS Master?

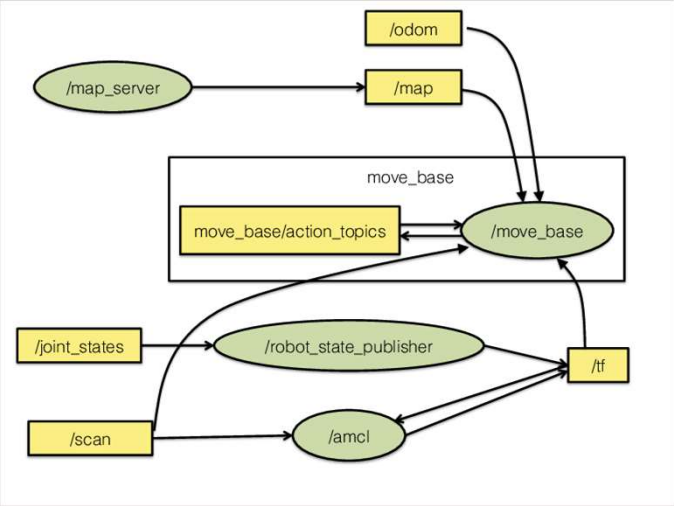
- **ROS Master:**
  - **Helps the ROS nodes to communicate and exchange messages**
  - Keep on tracking all ROS nodes and act as a bridge between two nodes to find each other
  - After the discovery of nodes, the two nodes will connect each other
  - The exchange of message between two nodes start after the connection.



What is a ROS Messages?

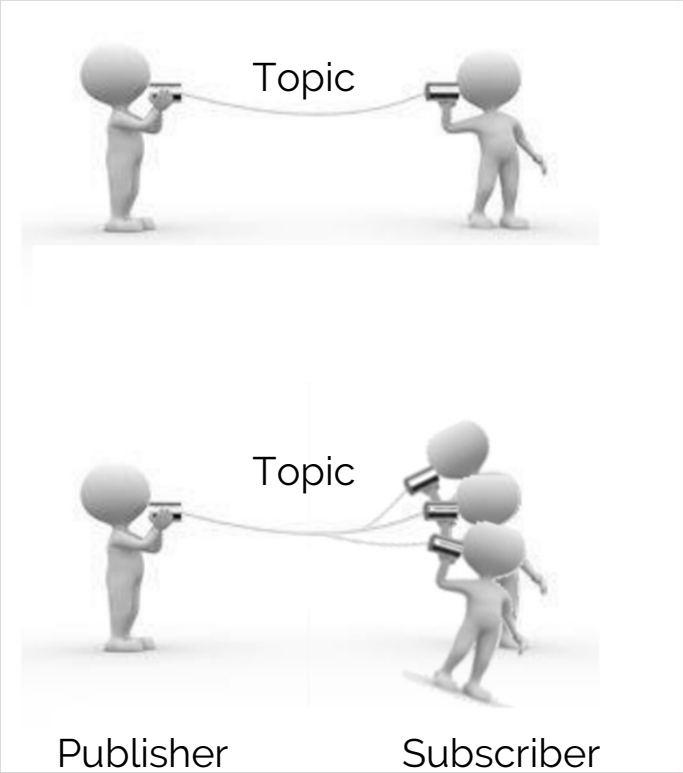
What is a ROS Message?

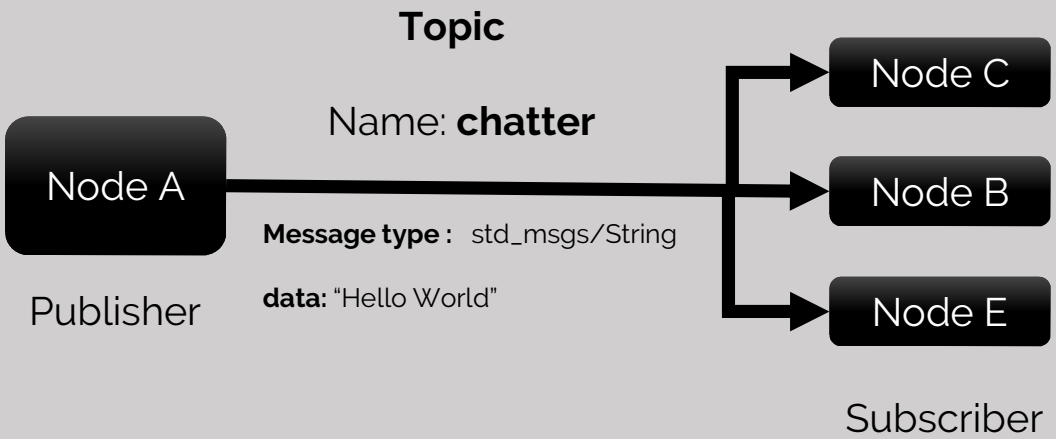
- **ROS Message:**
  - The ROS nodes are communicating with each other by passing ROS messages.
  - ROS message (**.msg**) is a data structure which can be of different types such as *Integer*, *Float*, *Boolean* etc.
  - It can hold same or different types of data type in a single message.
  - It can even hold an array of values of different data types.
  - We can use prebuilt messages in ROS or can create our own ROS messages
  - **E.g.: std_msgs/Int32, std_msgs/String, sensor_msgs/Image**
  - http://wiki.ros.org/Messages



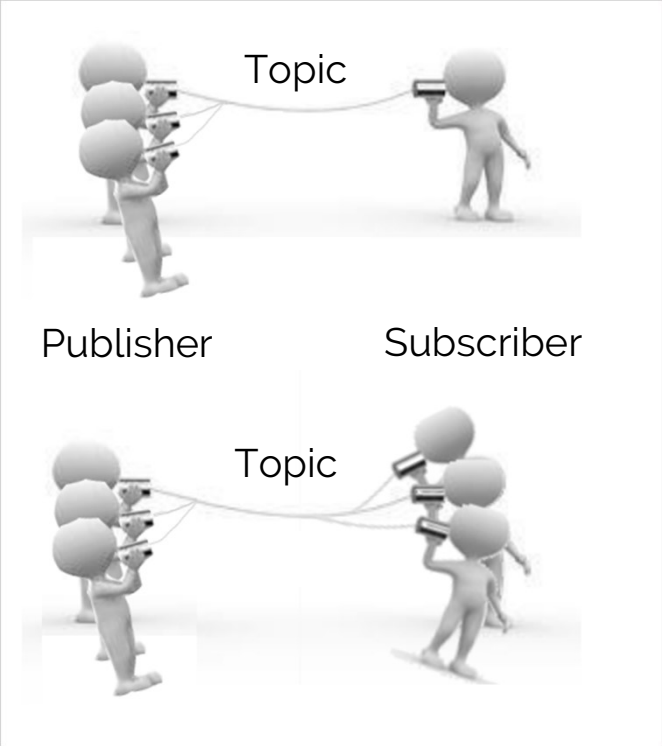
What is a ROS Topic?

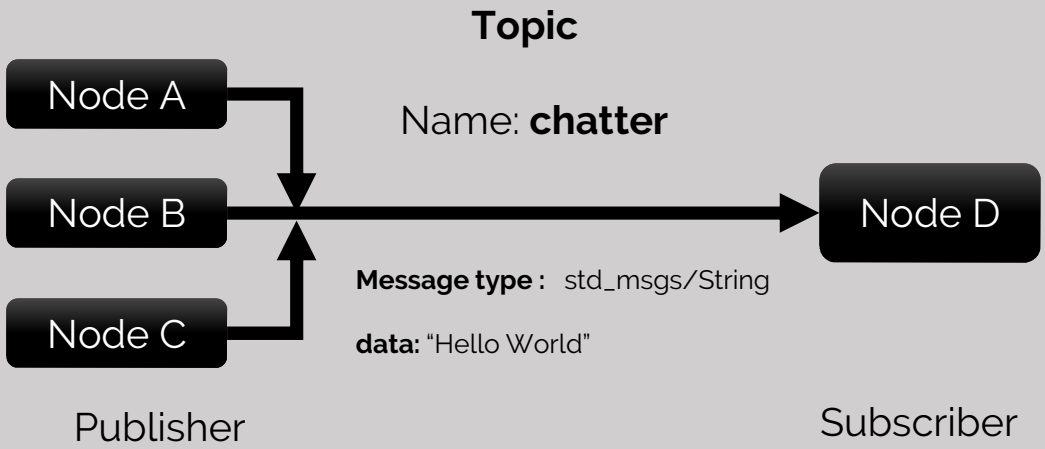
What is a ROS Topic?

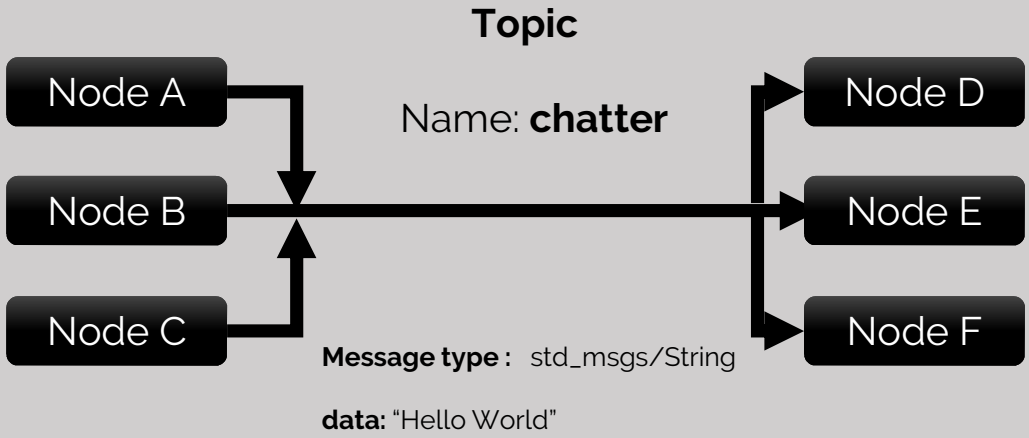




What is a ROS Topic?





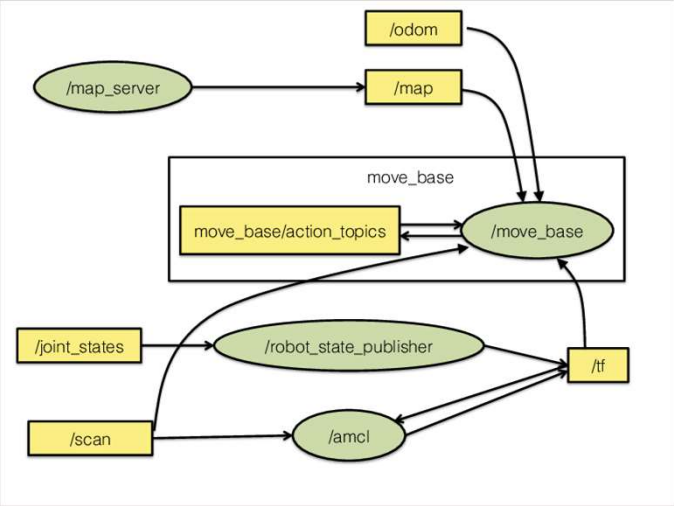


What is a ROS Topic?

- **ROS Topic:**
  - Topic is a named data bus in which ROS nodes send/receive ROS messages
  - Sending a message is called Publishing/Advertising
  - Receiving a message is called Subscribing
  - A ROS topic is identified by a name called Topic name
  - A topic can be created in a ROS node with a specific name and the type of ROS message it should transport

What is a ROS Topic?

- **ROS Topic:**
  - Once a topic is created, multiple nodes can send or receive the ROS message transporting through it
  - The *publishing* nodes and *subscribing* nodes can only exchange a message when they handle same message type, which is given during the creation of ROS Topic.
  - A ROS node can publish or subscribe any number of topics.
  - The publisher nodes and subscriber nodes are not aware of other's existence

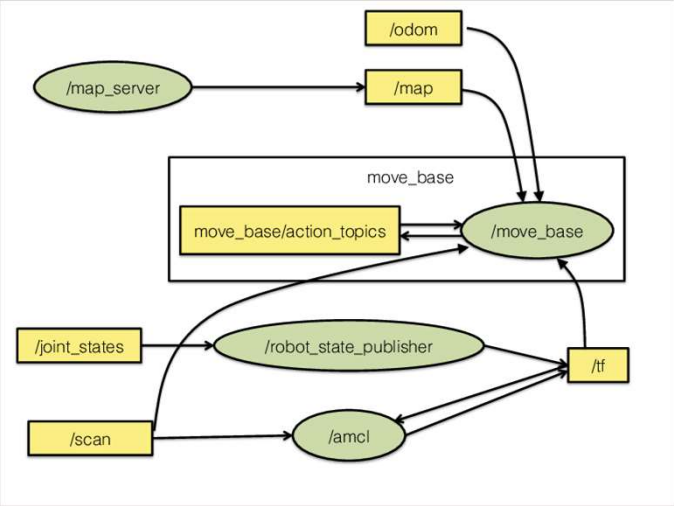


What is a ROS Parameter Server & Parameter?

What is a ROS Parameter Server?

- **ROS Parameter Server:**
  - It is a server program running inside the ROS Master
  - It enables the ROS nodes to store and retrieve variables at runtime
  - These variables are called ***ROS Parameters***.
  - The parameters can be accessed by all nodes in the ROS network
  - We can create private and public ROS parameters
  - It is implemented using XML RPC libraries

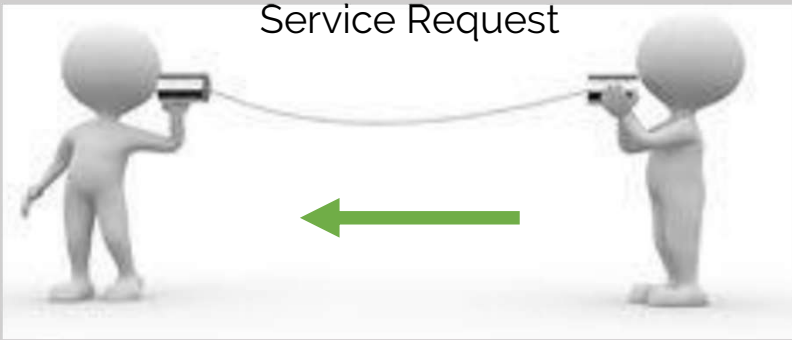
  - Parameter types: 32-bit integers, Booleans, strings, doubles, lists etc.
  - E.g. **/P: 10.0**
    /I: 1.0
    /D: 0.1



What is a ROS Service?

What is a ROS Service?

Hey Server, What is 2+2



Server Client

Thanks for contacting me, The answer is 4

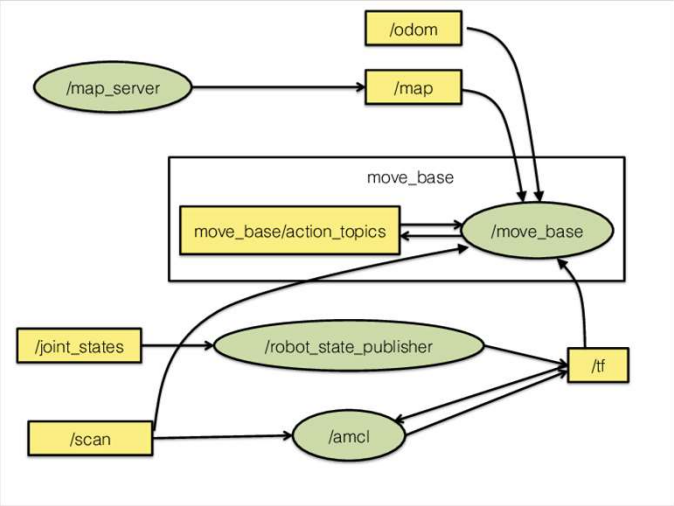
Server Client

What is a ROS Service?

- **ROS Service:**
  - The ROS topics are sending message in many-to-many one direction using publish/subscribe mechanism
  - We may need a *two way* communication like **request/rep**l**y** in a distributed system.
  - Using ROS services, we can implement a **request/reply** system in our nodes
  - Node which sending the request is called ***Service client node***.
  - Node which receive the request and execute the service is called ***ROS Service server.***
  - After the execution, the server node send the results back to the ROS client.

What is a ROS Service?

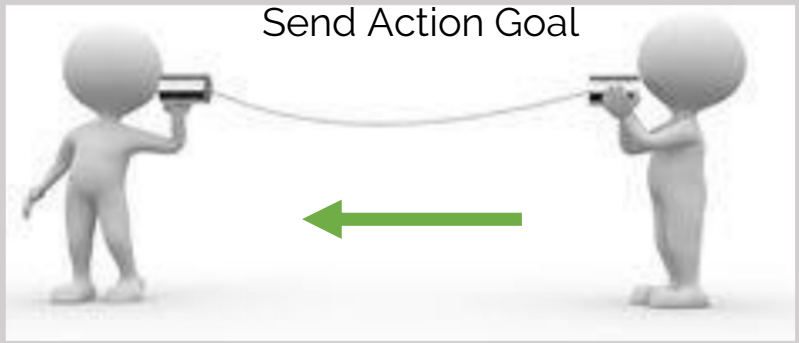
- **ROS Service:**
  - The node is sending request/reply using a *ROS service message*.
  - The service message (**.srv**) will have the definition of ***request*** and ***reply*** datatype
  - The service message type is similar to ROS messages but have different fields for request and reply.
  - E.g.: **std_srvs/SetBool**
  - http://wiki.ros.org/rosservice



What is a ROS Action?

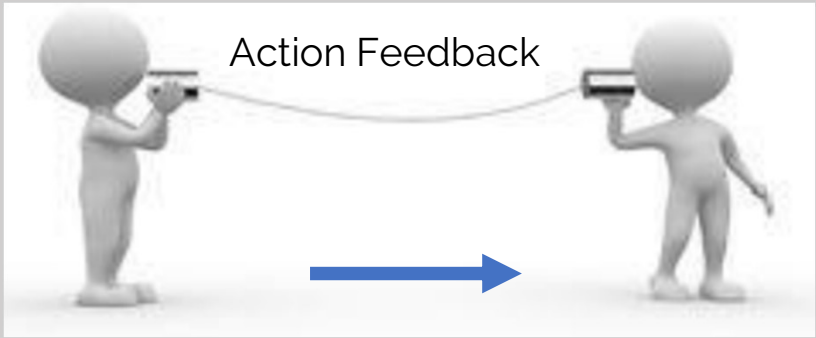
What is a ROS Action?

1)



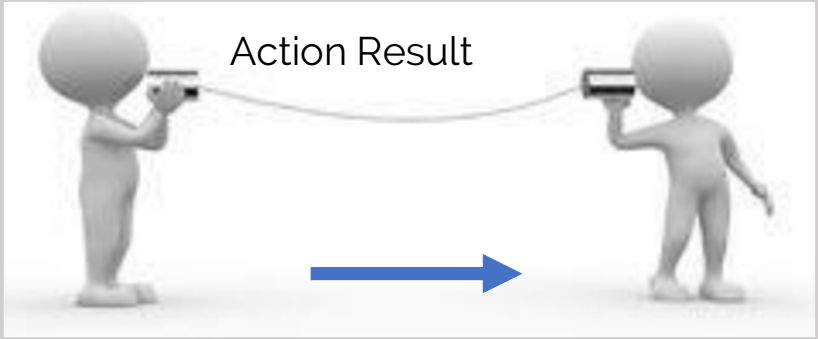
Server Client

2)



Server Client

3)



Server Client

What is a ROS Action?

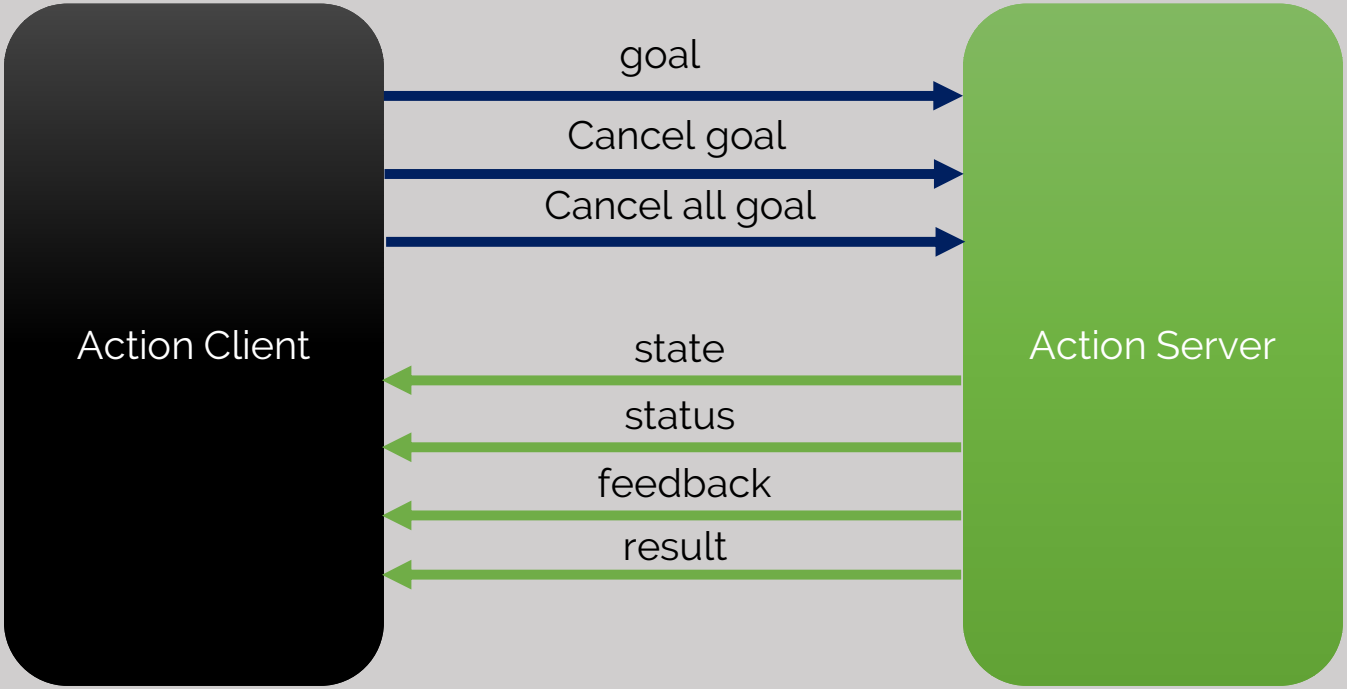
- **ROS Action:**
  - Similar to ROS Service
  - There is a Action Server and Client
  - In ROS service, we can't cancel the service once it requested
  - Using ROS action, we can cancel the current service request
  - The action server can also send feedback of the current goal execution.
  - After completing the goal, the action server will send the final result to the client

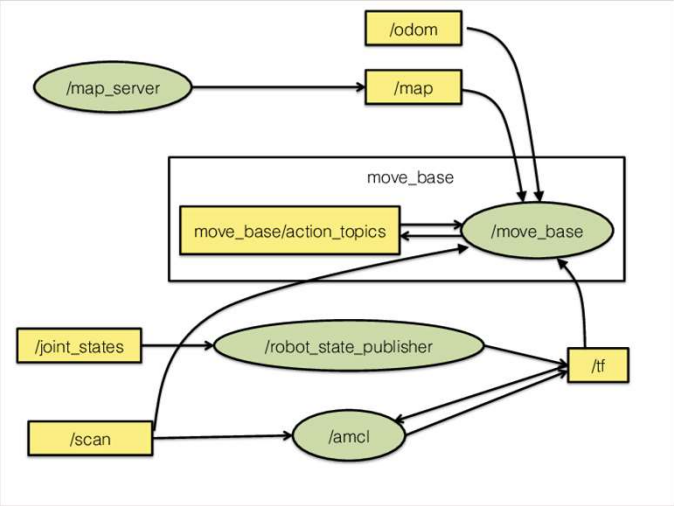
  - **ROS Action = ROS service + feedback + state + status + cancel goal**

What is a ROS Action?

- **ROS Action:**
  - Similar to ROS service, there is ROS Action messages
  - The ROS Action message (**.action**) contain
    - **Goal message type**
    - **Feedback message type**
    - **Result message type**

What is a ROS Action?

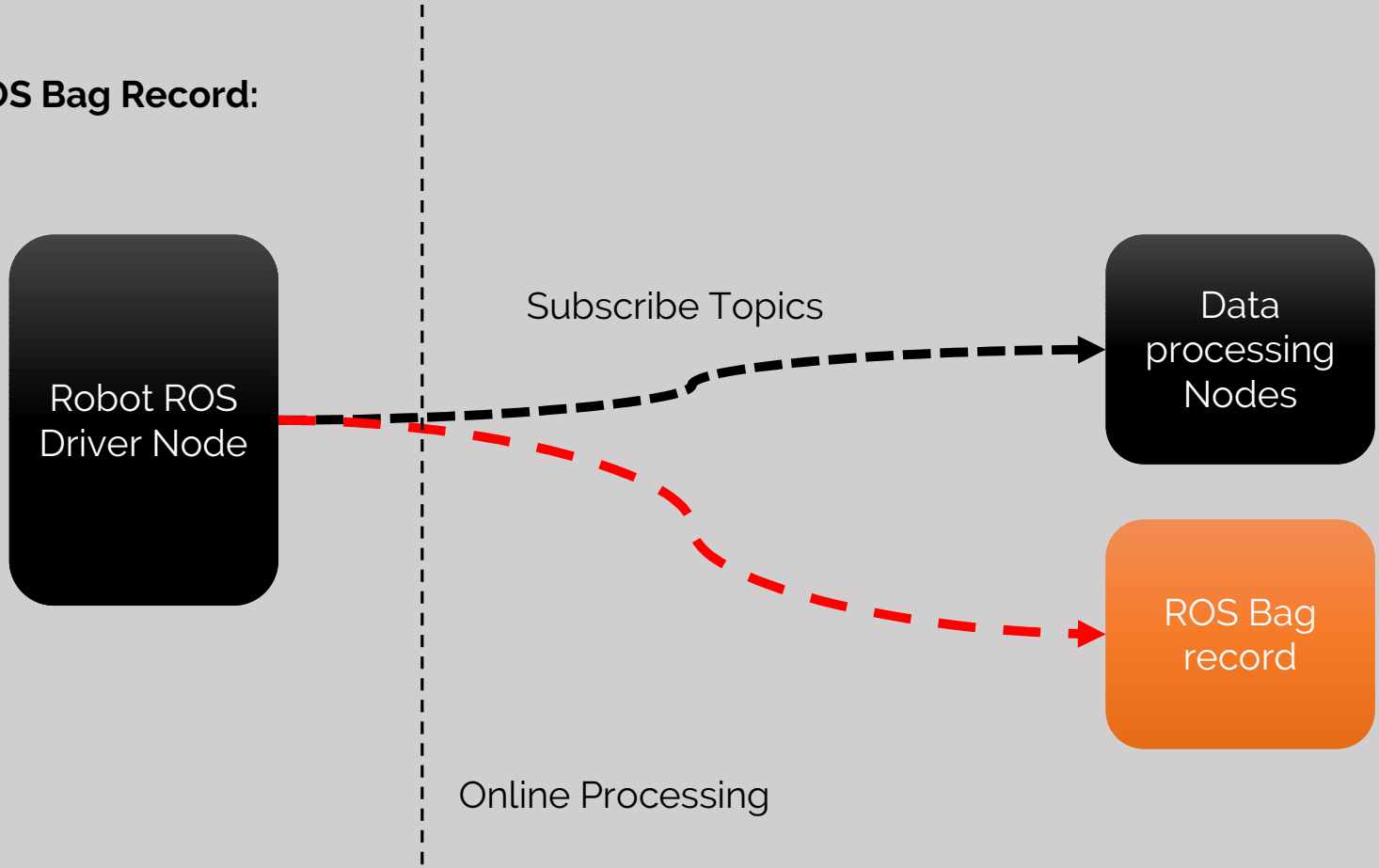




What is a ROS Bag?

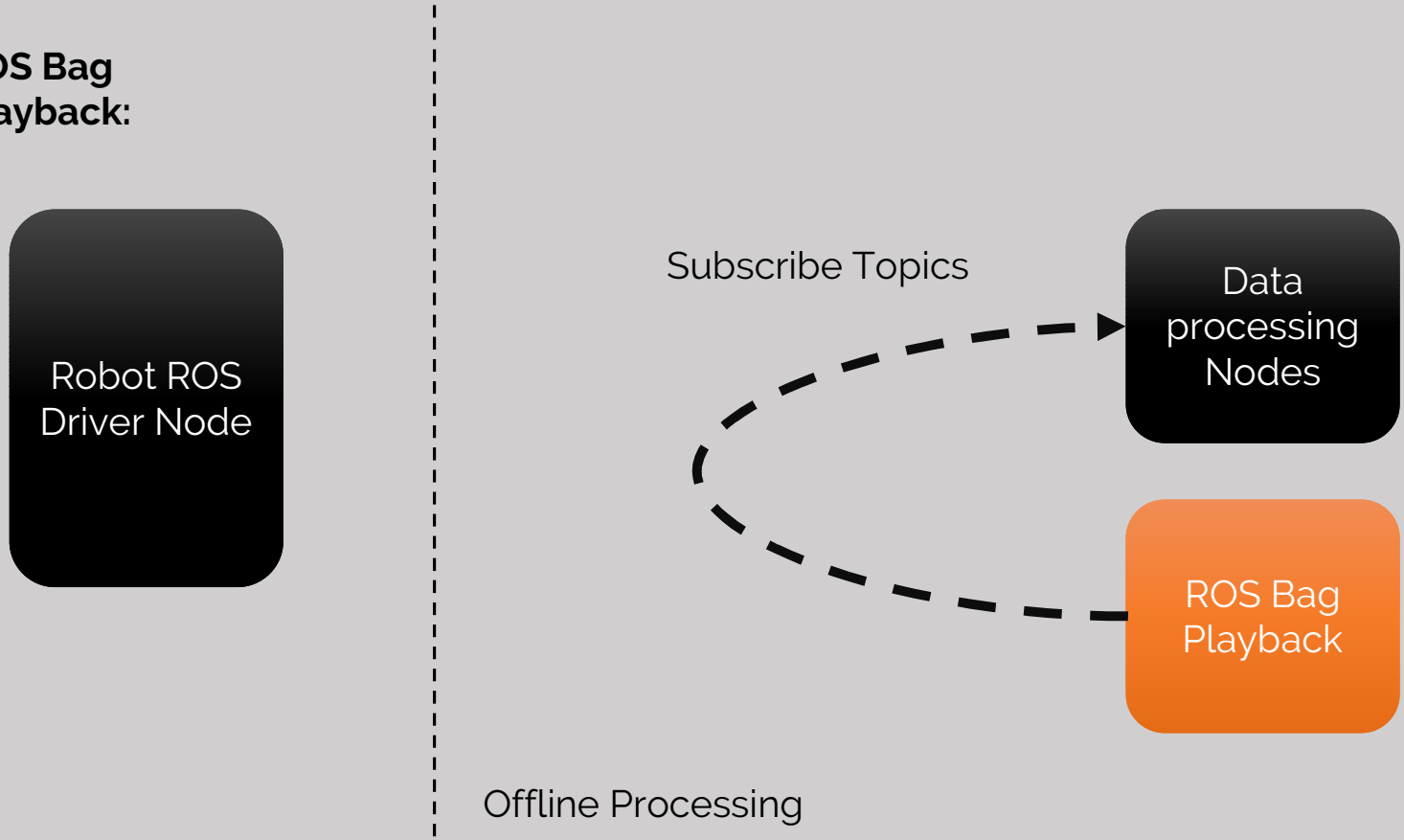
What is a ROS Bags?

ROS Bag Record:



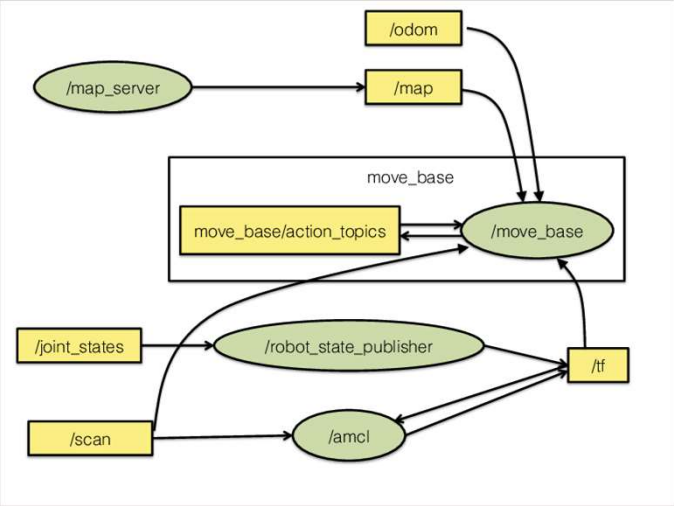
What is a ROS Bags?

ROS Bag Playback:



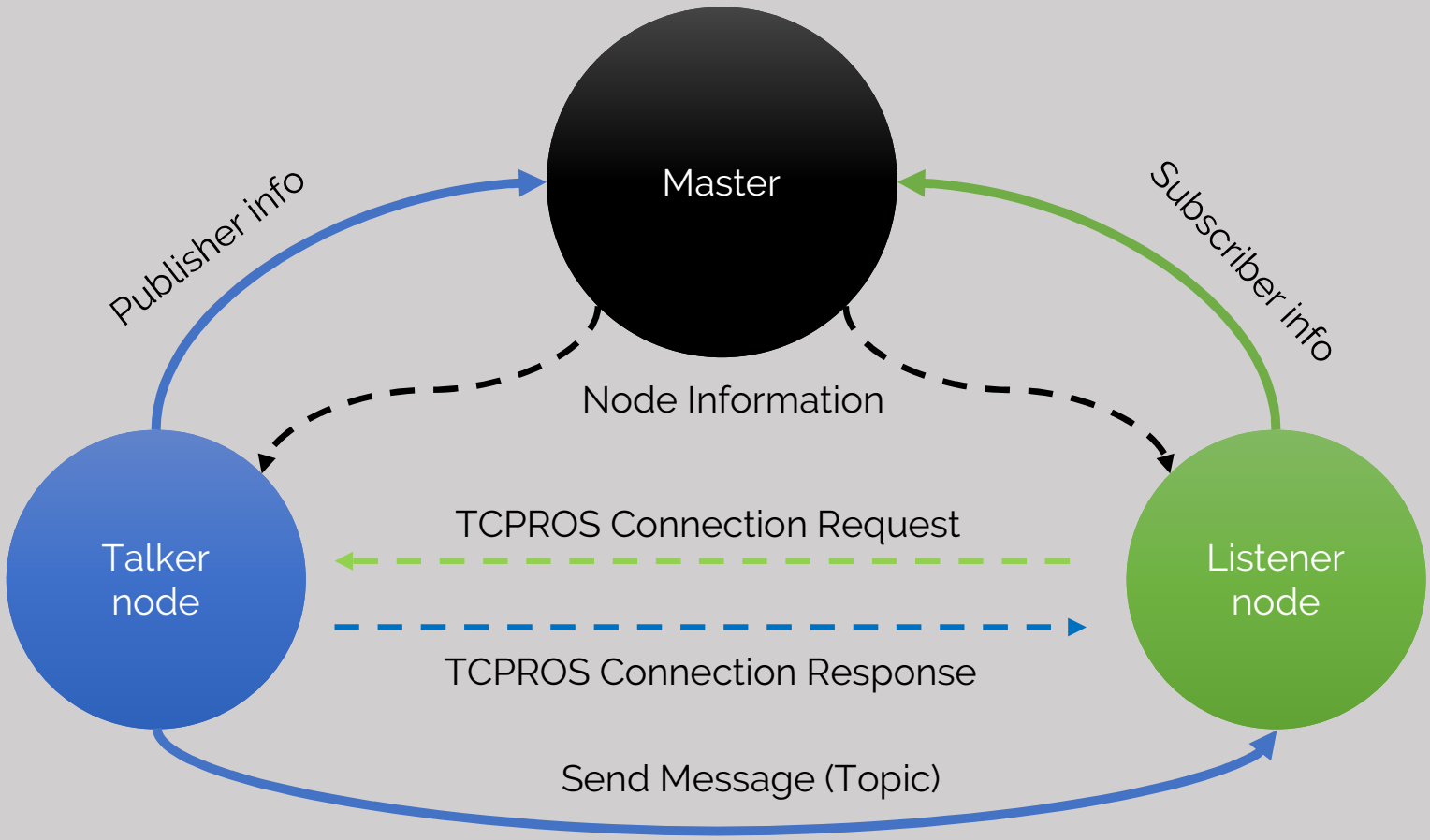
What is a ROS Bag?

- **ROS Bag:**
  - The ROS Bags are used for data logging
  - A ROS bag file can store ROS message data with a file extension (***.bag***)
  - The ROS topics can be record/playback, process, analyze and visualize using bag files.
  - One of useful feature in ROS for offline programming (without the robot hardware)
  - http://wiki.ros.org/Bags



Secret of ROS inter-process communication

ROS Communication: Topic



ROS Communication: Topic

1. Running ROS Master

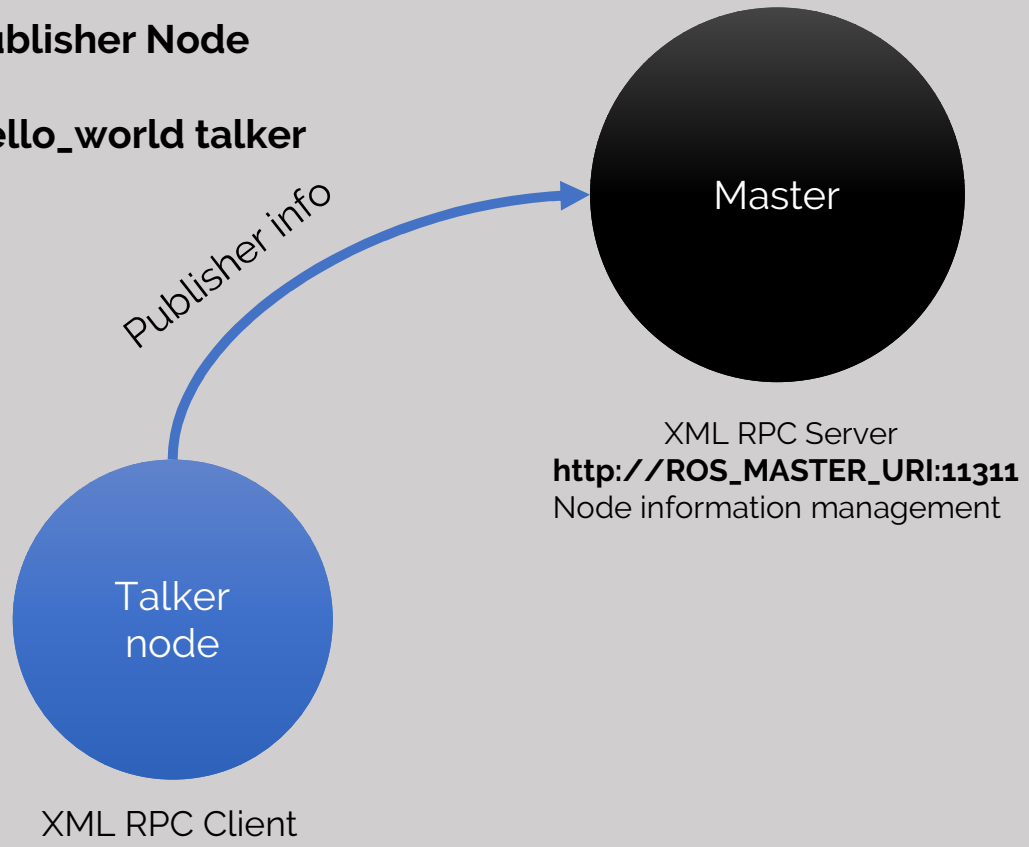
$ roscore

XML RPC Server
http://ROS_MASTER_URI:11311
Node information management

ROS Communication: Topic

2. Running a Publisher Node

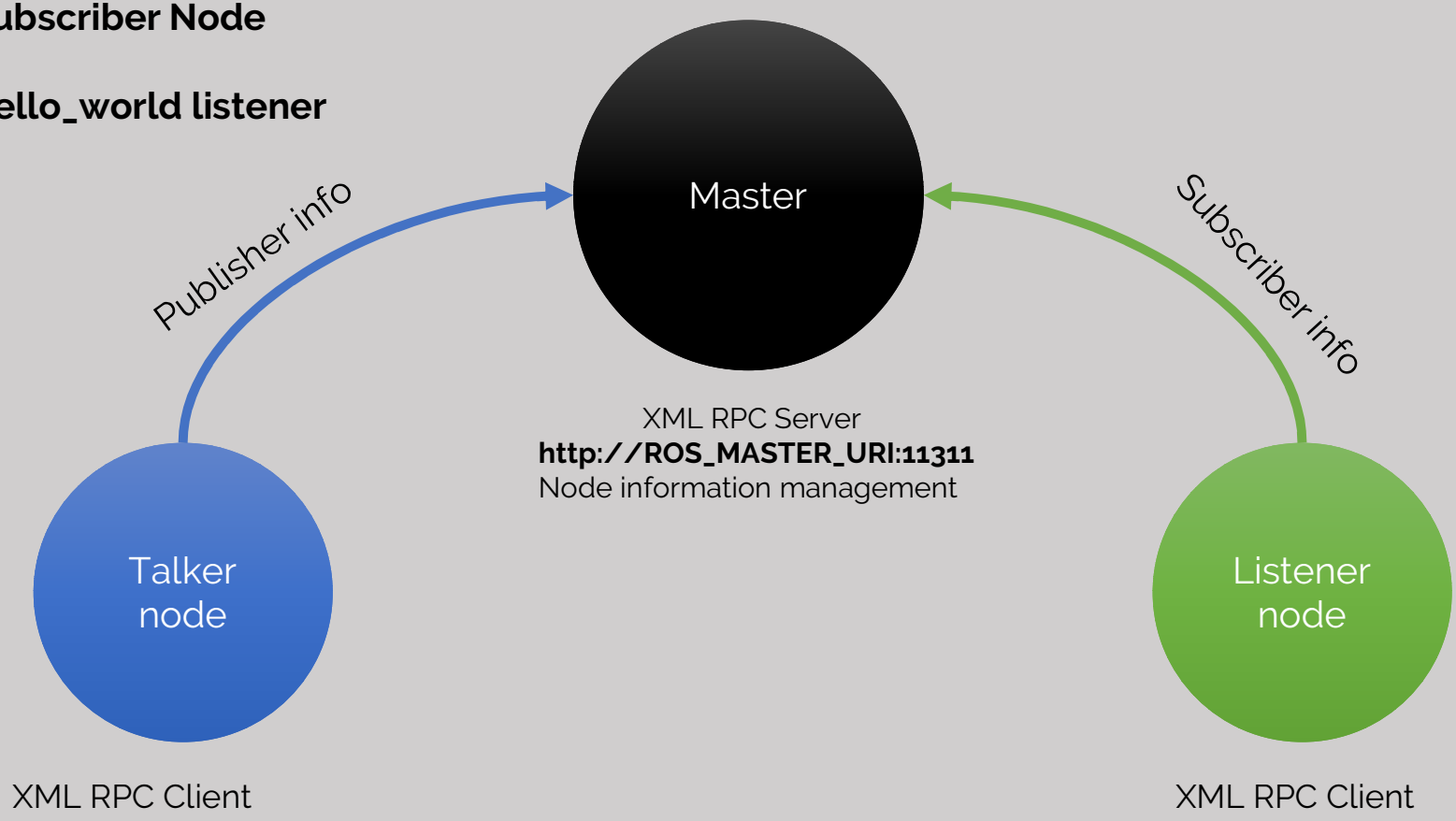
E.g. $ rosrun hello_world talker



ROS Communication: Topic

3. Running a Subscriber Node

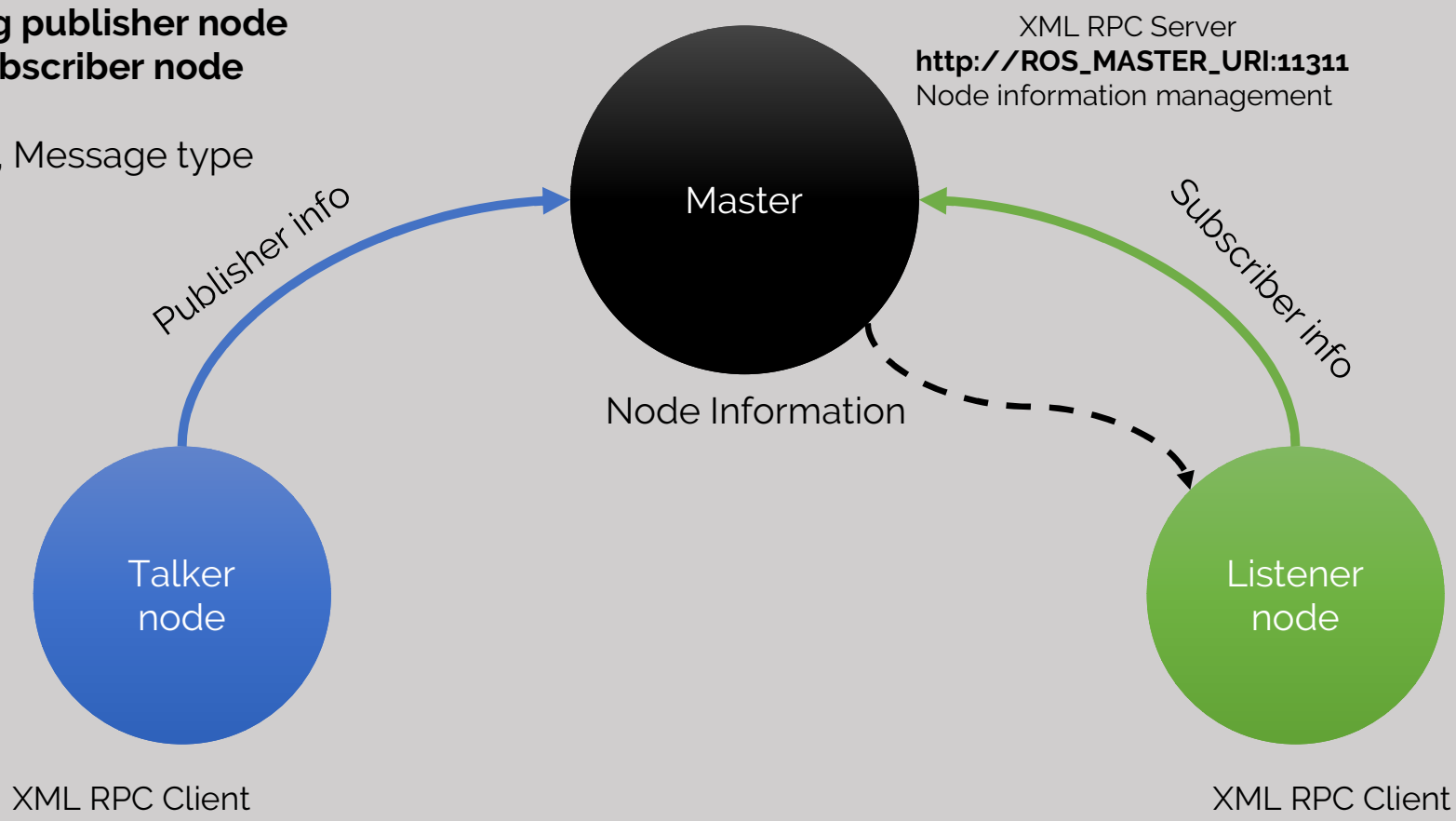
E.g. $ rosrun hello_world listener



ROS Communication: Topic

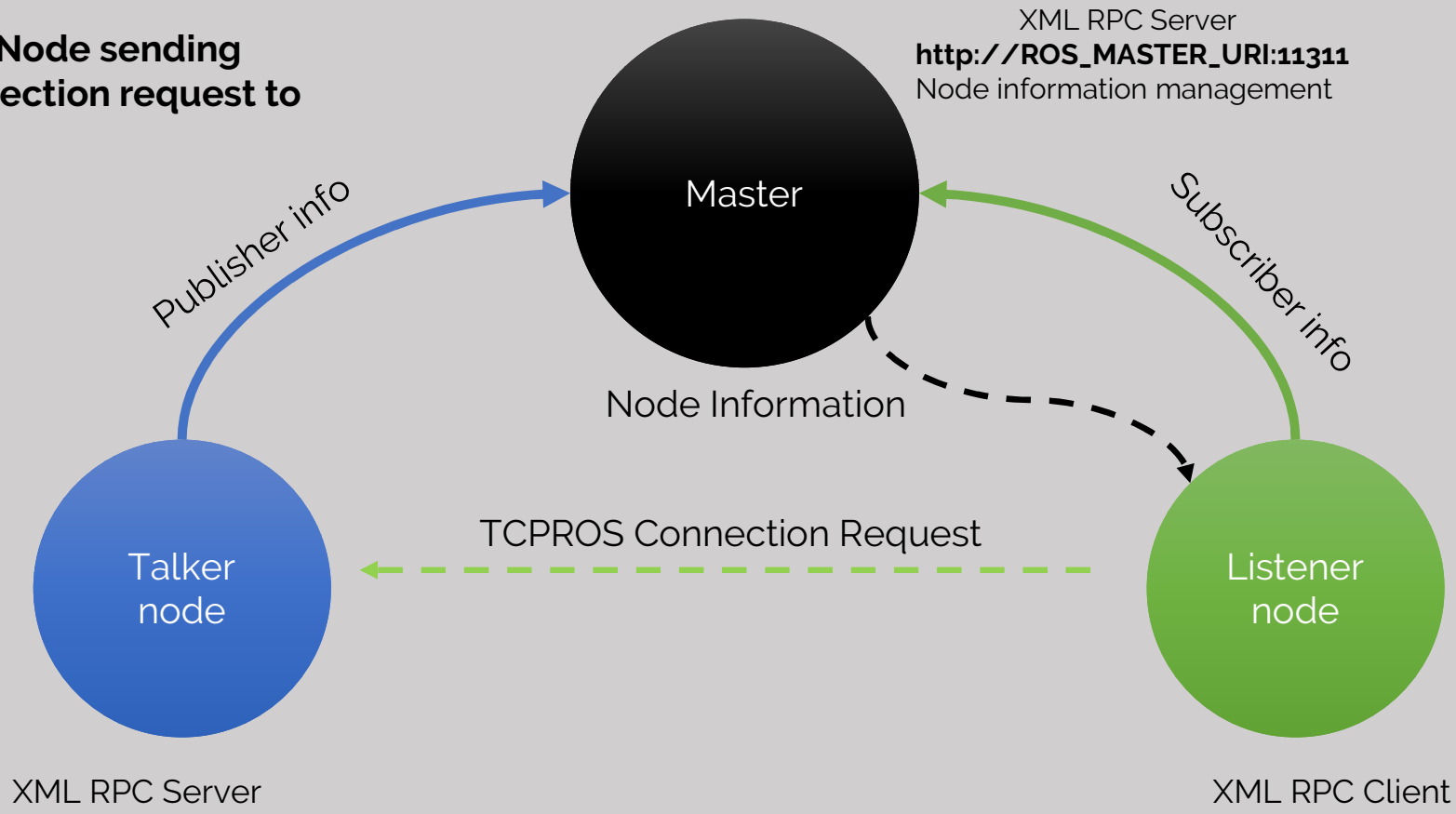
4. Master Sending publisher node information to subscriber node

Node info: Topics, Message type etc.



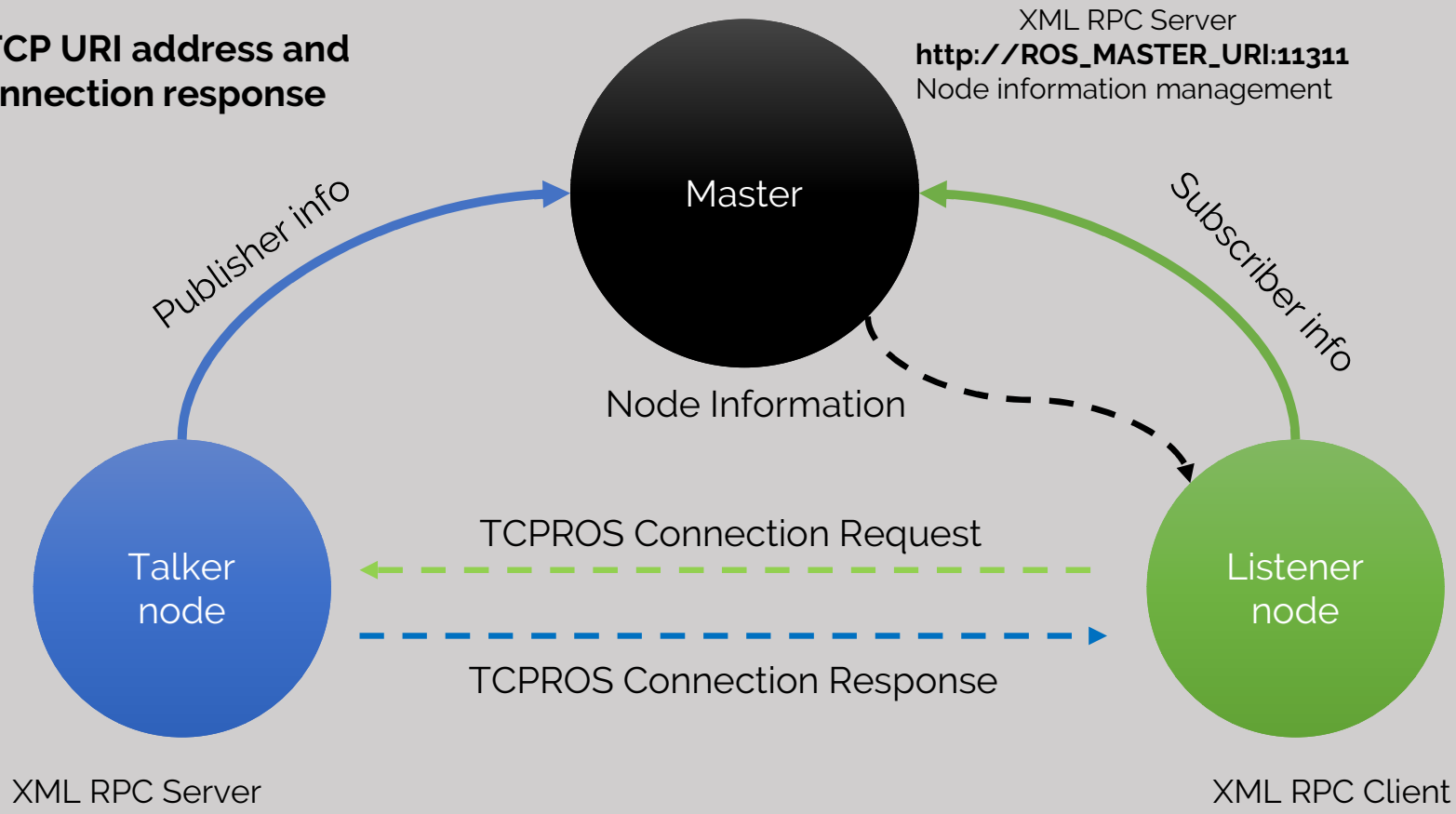
ROS Communication: Topic

5. Subscriber Node sending TCPROS connection request to Publisher



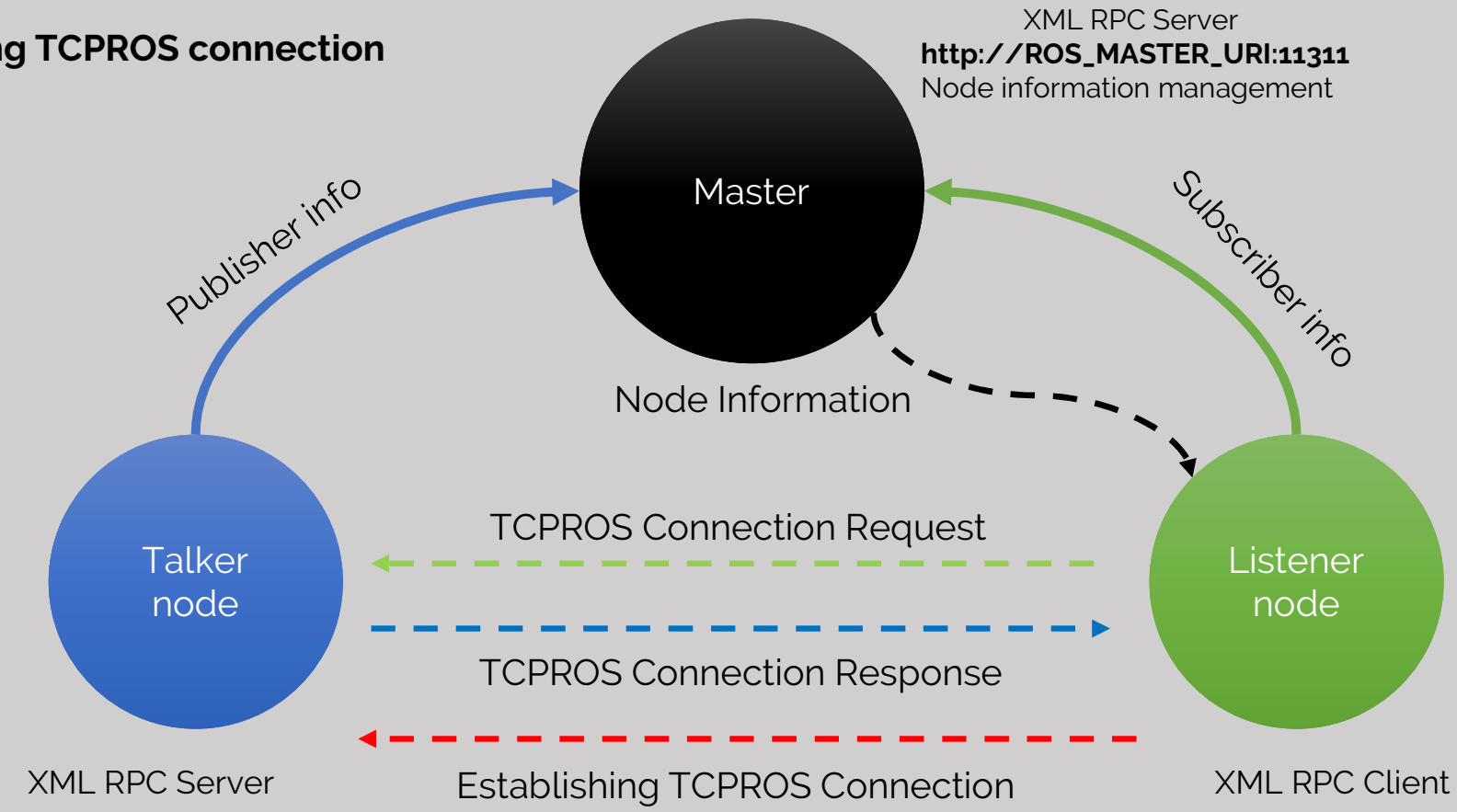
ROS Communication: Topic

6. Returning TCP URI address and port as the connection response



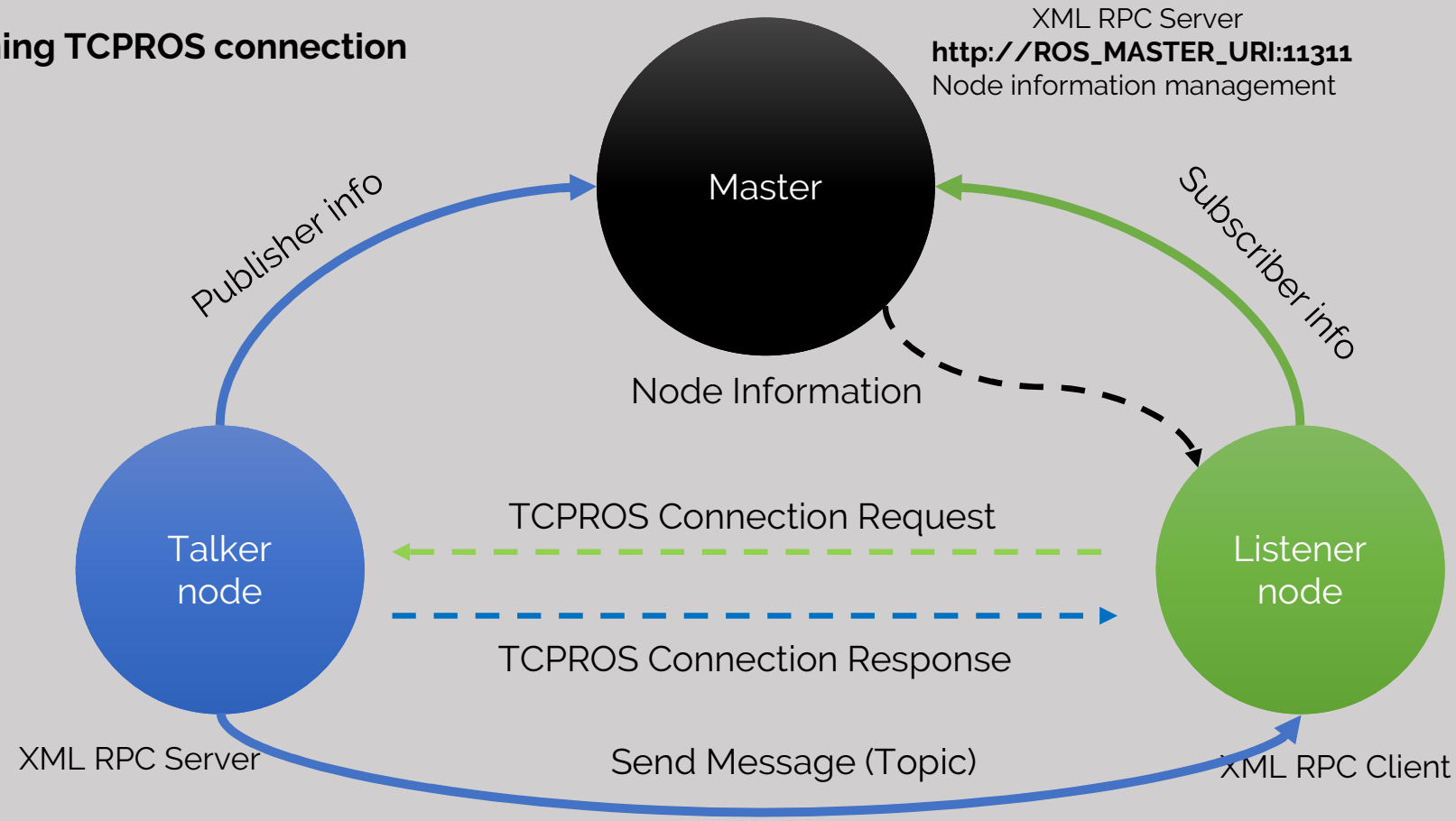
ROS Communication: Topic

7. Establishing TCPROS connection



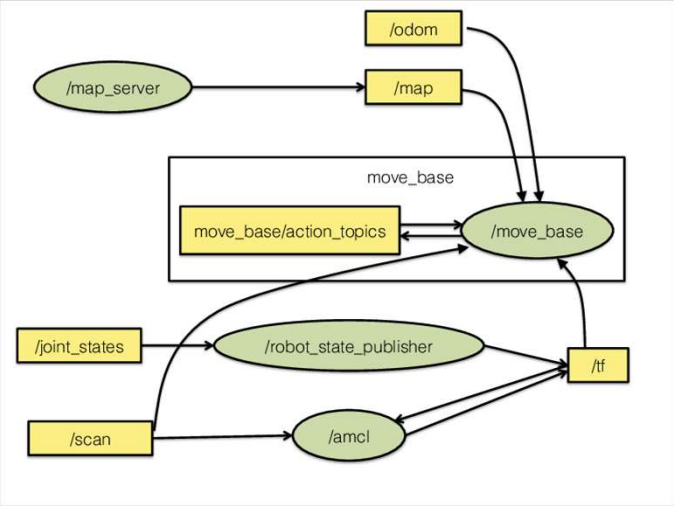
ROS Communication: Topic

8. Establishing TCPROS connection



ROS Communication: Topic

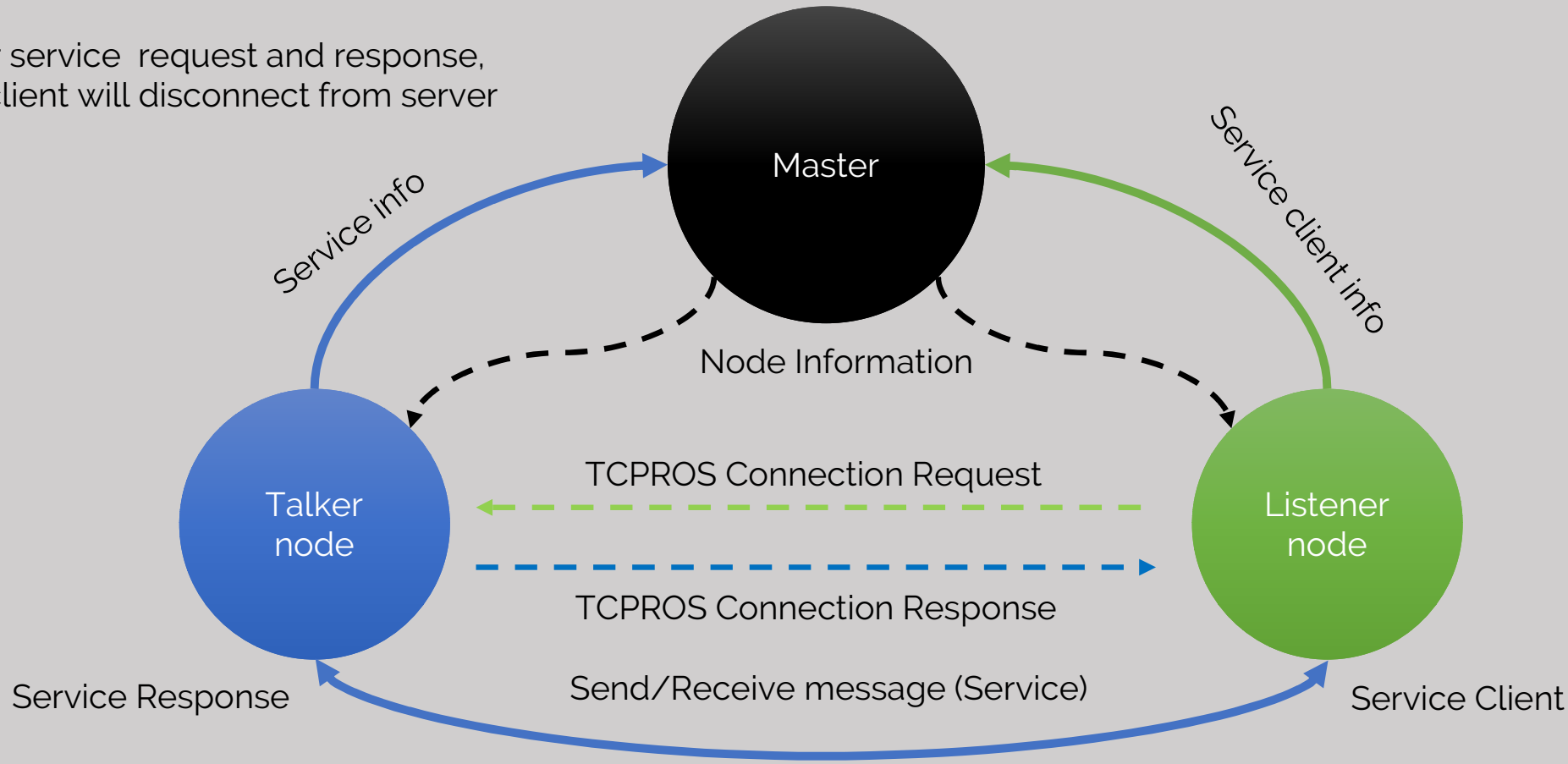
9. Publishing and Subscribing topic



ROS Communication: ROS Services

ROS Communication: Services

After service request and response,
the client will disconnect from server



ROS Communication: Services

After service request and response,
the client will disconnect from server

ROS

What is ROS Build System?

Discussing the basic concepts of ROS build system

ROS Build System

- **ROS Build System:**
  - A build system generate target executable/library from the source code
  - In ROS, we are using ***catkin*** *(http://wiki.ros.org/catkin/conceptual_overview)*
  - Catkin = CMake Macros (http://www.cmake.org) + Python script
  - The source code in ROS is organized as '*ROS Packages*'.
  - The build system in ROS is used build complex code organization and dependencies
  - The catkin commands used to create ROS packages and workspace

ROS

What is ROS Catkin Workspace and ROS Packages?

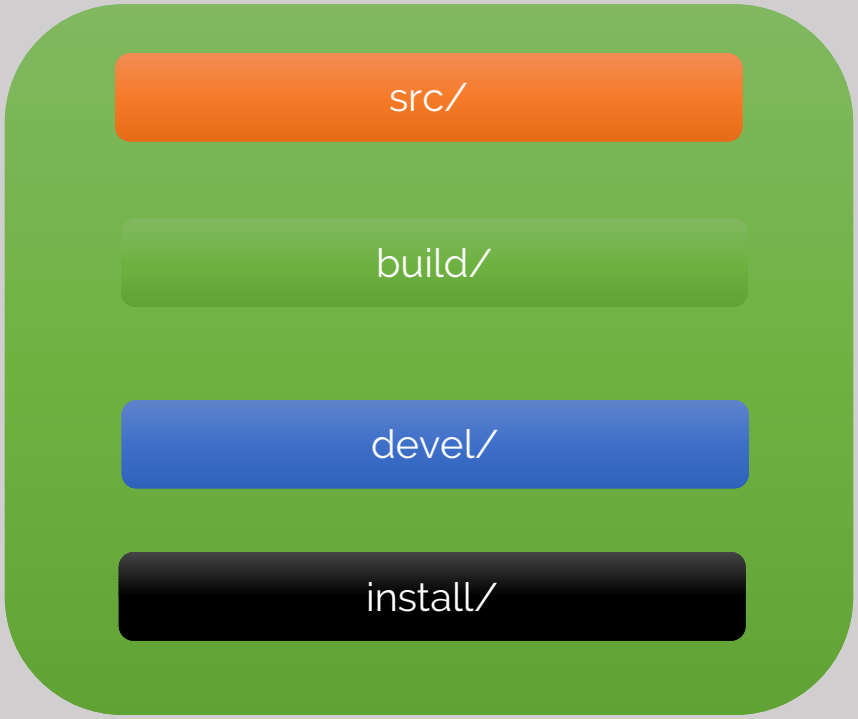
Discussing the basic concepts of ROS catkin workspace and packages

ROS Catkin workspace and packages

- **ROS Catkin Workspace and packages:**
  - It is the folder where we can modify, build and install **catkin ROS packages**
  - The catkin ROS packages are unit of organizing software in ROS.
  - A ROS package can have ROS nodes, ROS library, configuration files etc.
  - We can create any number of catkin workspace in your PC
  - http://wiki.ros.org/catkin/workspaces

ROS Catkin workspace and packages

catkin workspace folder/



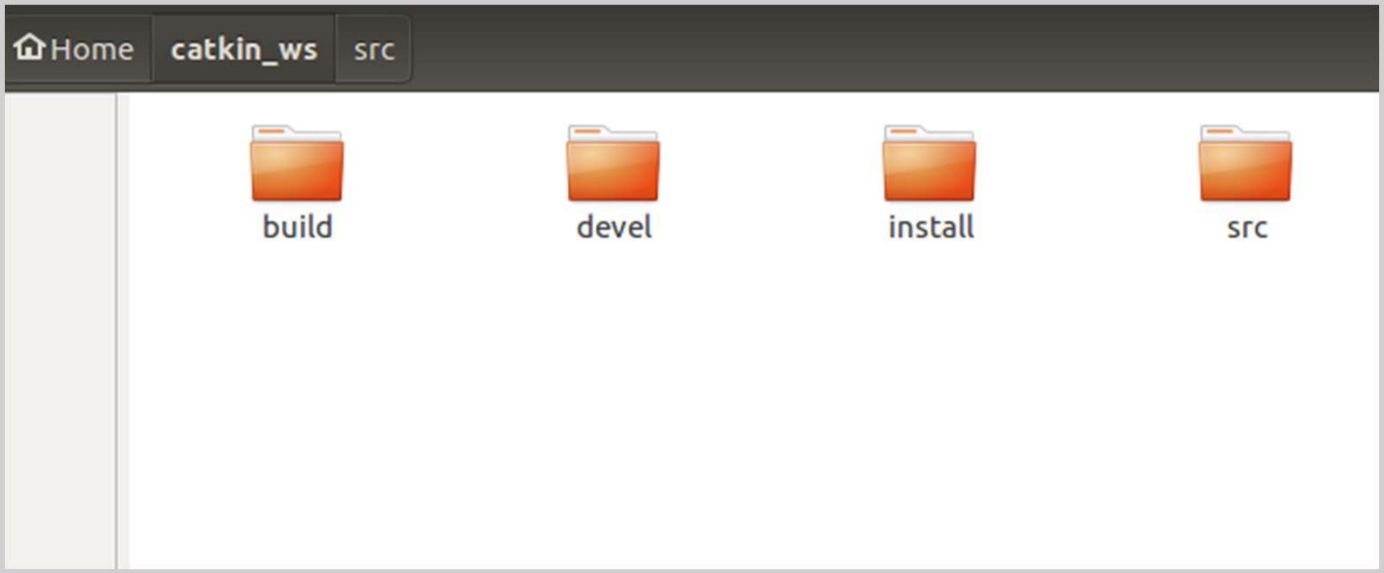
src: The ROS packages are keeping inside src folder.

build: CMake and catkin keep their cache information and other intermediate files here.

devel: Build targets (executable) are stored here

install: The build targets can be installed into this space

ROS Catkin workspace and packages

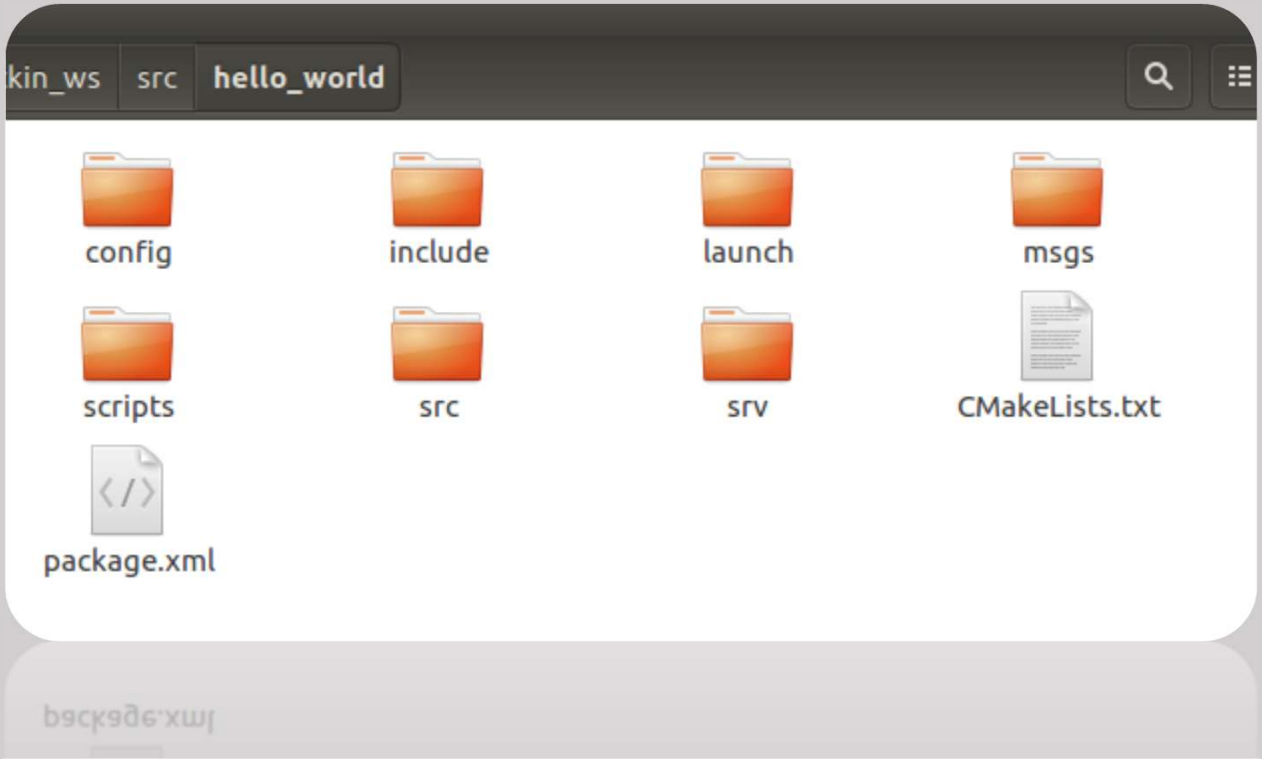


ROS Catkin workspace and packages

- **ROS Package:**
  - The Software in ROS is organized in *packages.*
  - A ROS package may have ROS nodes, library, configuration file etc.
  - The package provides modularity & reusability
  - All ROS packages are keeping inside ROS catkin workspace
  - We can build and customize ROS packages inside a workspace
  - http://wiki.ros.org/Packages

ROS Catkin workspace and packages

- **A typical ROS Package:**



ROS Catkin workspace and packages

- **A typical ROS Package:**
  - **include/package_name**: C++ include headers
  - **msg/**: Folder containing Message (msg) types for ROS topic communication
  - **src/package_name/**: Mainly C++ Source files
  - **srv/**: Folder containing ROS Service (srv) types

ROS Catkin workspace and packages

- **A typical ROS Package:**
  - **scripts/**: Mainly executable python scripts
  - **CMakeLists.txt**: CMake build file.
  - **package.xml**: Package catkin/package.xml. This file contain complete information of the package and its dependencies.

ROS

ROS Command tools

Introduction to ROS Command line tools

Command : roscore

- roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system.

- You **must** have a **roscore** running in order for ROS nodes to communicate. It is launched using the **roscore** command.

- **roscore = rosmaster + parameter server + rosout logging node**

Try roscore in your terminal

```
roscore http://robot-VirtualBox:11311/
robot@robot-VirtualBox:~$ roscore
... logging to /home/robot/.ros/log/e9b72596-6132-11e5-830f-0800273c354c/
ch-robot-VirtualBox-8750.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://robot-VirtualBox:44954/
ros_comm version 1.11.10


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.10

NODES

auto-starting new master
process[master]: started with pid [8762]
ROS_MASTER_URI=http://robot-VirtualBox:11311/
```

Command : rostopic

```
robot@robot-VirtualBox: ~
roscore http://robot-VirtualBox:11311/        robot@robot-VirtualBox: ~
robot@robot-VirtualBox:~$ rostopic
rostopic is a command-line tool for printing information about

Commands:
        rostopic bw     display bandwidth used by topic
        rostopic echo   print messages to screen
        rostopic find   find topics by type
        rostopic hz     display publishing rate of topic
        rostopic info   print information about active topic
        rostopic list   list active topics
        rostopic pub    publish data to topic
        rostopic type   print topic type

Type rostopic <command> -h for more detailed usage, e.g. 'rostc

robot@robot-VirtualBox:~$ rostopic list
/rosout
/rosout_agg
robot@robot-VirtualBox:~$
```

Command : rosnode

```
robot@robot-VirtualBox: ~
roscore http://robot-VirtualBox:11311/        ×   robot@robot-VirtualBox: ~
robot@robot-VirtualBox:~$ rosnode
rosnode is a command-line tool for printing information about ROS Nodes.

Commands:
        rosnode ping    test connectivity to node
        rosnode list    list active nodes
        rosnode info    print information about node
        rosnode machine list nodes running on a particular machine or list m
nes
        rosnode kill    kill a running node
        rosnode cleanup purge registration information of unreachable nodes

Type rosnode <command> -h for more detailed usage, e.g. 'rosnode ping -h'

robot@robot-VirtualBox:~$ rosnode list
/rosout
robot@robot-VirtualBox:~$
```

Command : rosparam

```
robot@robot-VirtualBox: ~
roscore http://robot-VirtualBox:11311/        robot@robot-VirtualBox: ~
robot@robot-VirtualBox:~$ rosparam
rosparam is a command-line tool for getting, setting, and deleting p
om the ROS Parameter Server.

Commands:
        rosparam set    set parameter
        rosparam get    get parameter
        rosparam load   load parameters from file
        rosparam dump   dump parameters to file
        rosparam delete delete parameter
        rosparam list   list parameter names

robot@robot-VirtualBox:~$ rosparam list
/rosdistro
/roslaunch/uris/host_robot_virtualbox__44954
/rosversion
/run_id
robot@robot-VirtualBox:~$
```

Command : rosrun

```
robot@robot-pc: ~
roscore http://robot-pc:11311/ | robot@robot-pc: ~ | robot@robot-pc: ~ | robot@robot-pc: ~
robot@robot-pc:~$ rosrun
Usage: rosrun [--prefix cmd] [--debug] PACKAGE EXECUTABLE [ARGS]
  rosrun will locate PACKAGE and try to find
  an executable named EXECUTABLE in the PACKAGE tree.
  If it finds it, it will run it with ARGS.
robot@robot-pc:~$
```

roscpp tutorials: talker & listener

```
robot@robot-pc: ~
roscore http://robot-pc:11311/ 53x9
auto-starting new master
process[master]: started with pid [21816]
ROS_MASTER_URI=http://robot-pc:11311/

setting /run_id to e45a68de-82ef-11e7-be47-080027c6ff
27
process[rosout-1]: started with pid [21829]
started core service [/rosout]
```

```
robot@robot-pc: ~ 53x9
[ INFO] [1502935302.643387814]: hello world 359
[ INFO] [1502935302.743370823]: hello world 360
[ INFO] [1502935302.843917325]: hello world 361
[ INFO] [1502935302.943391274]: hello world 362
[ INFO] [1502935303.043465820]: hello world 363
[ INFO] [1502935303.143660817]: hello world 364
[ INFO] [1502935303.245170718]: hello world 365
[ INFO] [1502935303.346762969]: hello world 366
```

```
robot@robot-pc: ~ 53x9
[ INFO] [1502935303.044257480]: I heard: [hello world
 363]
[ INFO] [1502935303.144208897]: I heard: [hello world
 364]
[ INFO] [1502935303.246165810]: I heard: [hello world
 365]
[ INFO] [1502935303.347256646]: I heard: [hello world
 366]
```

```
robot@robot-pc: ~ 53x9
robot@robot-pc:~$
```

roscpp tutorials: talker & listener [Code explanation]

- $ **roscore**

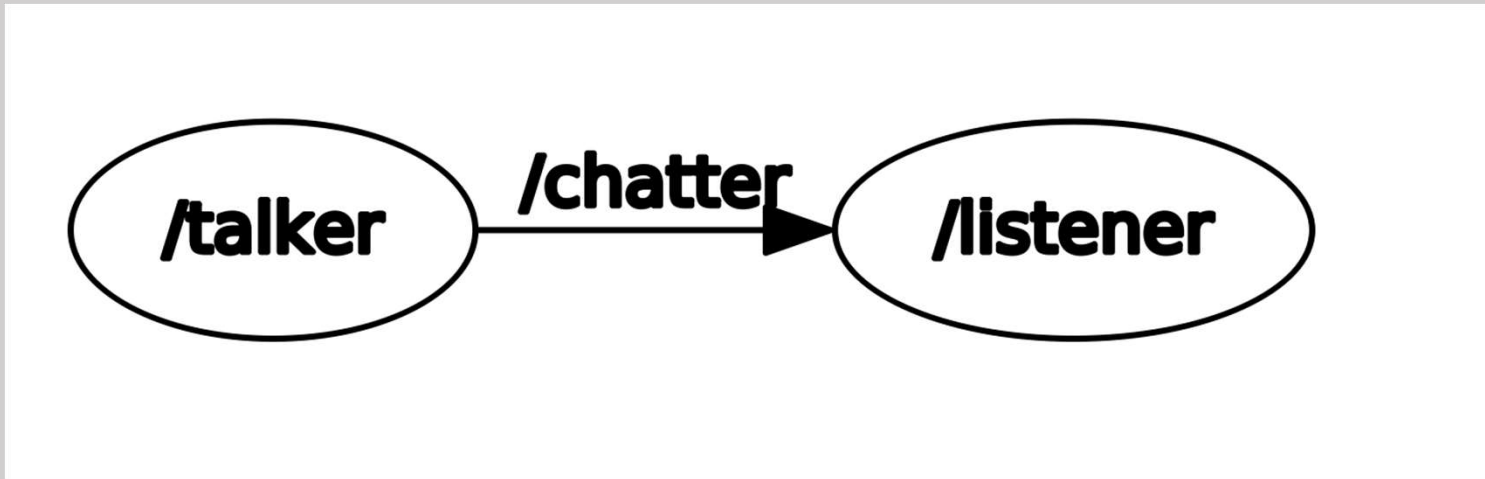
[Publisher]

- $ **rosrun roscpp_tutorials talker**

[Subscriber]

- $ **rosrun roscpp_tutorials listener**

roscpp tutorials: talker & listener



Playing with ROS command tools

robot@robot-pc: ~

roscore http://robot-pc:11311/ 53x9

```
auto-starting new master
process[master]: started with pid [21816]
ROS_MASTER_URI=http://robot-pc:11311/

setting /run_id to e45a68de-82ef-11e7-be47-080027c6ff
27
process[rosout-1]: started with pid [21829]
started core service [/rosout]
```

robot@robot-pc: ~ 53x9

```
[ INFO] [1502935302.643387814]: hello world 359
[ INFO] [1502935302.743370823]: hello world 360
[ INFO] [1502935302.843917325]: hello world 361
[ INFO] [1502935302.943391274]: hello world 362
[ INFO] [1502935303.043465820]: hello world 363
[ INFO] [1502935303.143660817]: hello world 364
[ INFO] [1502935303.245170718]: hello world 365
[ INFO] [1502935303.346762969]: hello world 366
```

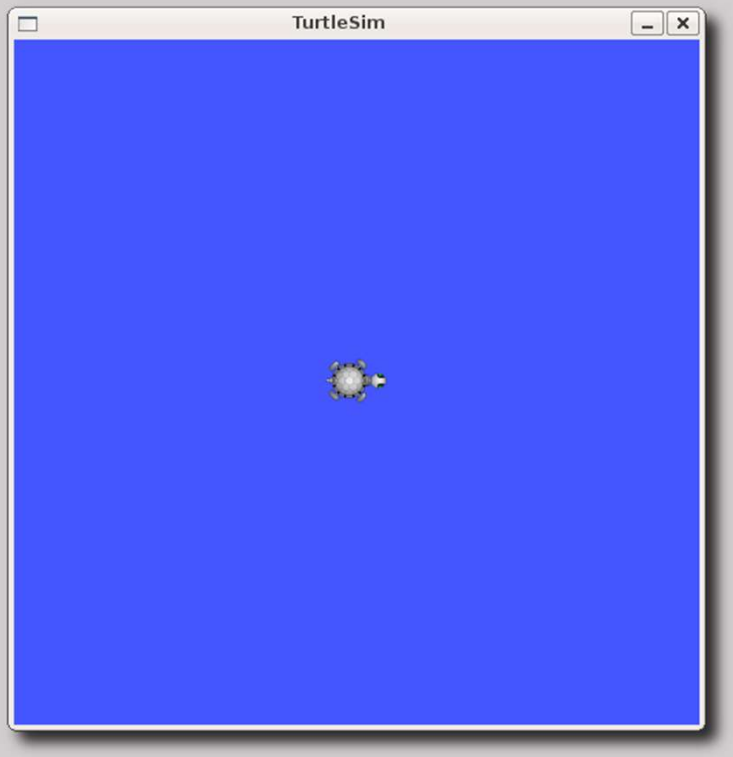
robot@robot-pc: ~ 53x9

```
[ INFO] [1502935303.044257480]: I heard: [hello world
 363]
[ INFO] [1502935303.144208897]: I heard: [hello world
 364]
[ INFO] [1502935303.246165810]: I heard: [hello world
 365]
[ INFO] [1502935303.347256646]: I heard: [hello world
 366]
```

robot@robot-pc: ~ 53x9

```
robot@robot-pc:~$
```

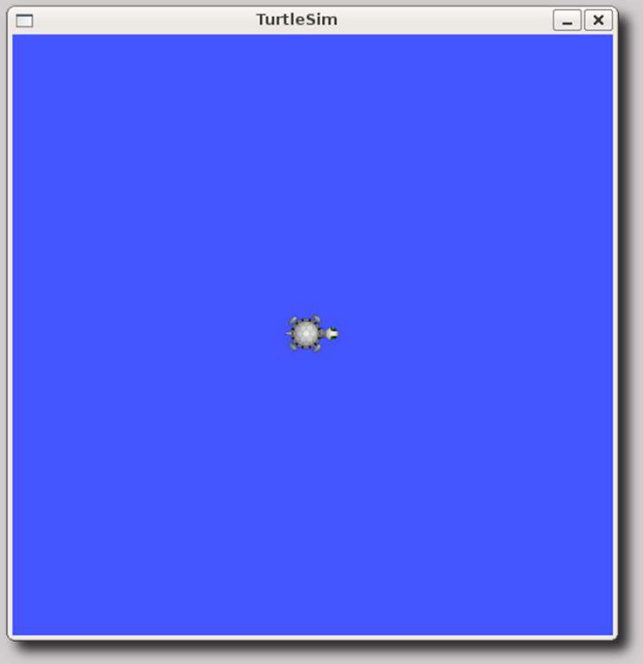
Turtlesim Demo



Do you remember Logo Programming?

Learn ROS using Turtlesim

- http://wiki.ros.org/turtlesim



Launching turtlesim

- $ **roscore** [In first terminal]
- **$ rosrun turtlesim turtlesim_node** [In second terminal]