Introduction to ROS

COMP3431

Robot Software Architectures

Robot Software Architecture

- A robot’s software has to control a lot of things:
  - 2D/3D Cameras, LIDAR, Microphones, etc
  - Drive motors, Arm motors
  - Vision, Mapping, Navigation
  - Task Planning, Motion Planning
  - Speech and Natural Language Processing
  - ….

Robot Software Architecture

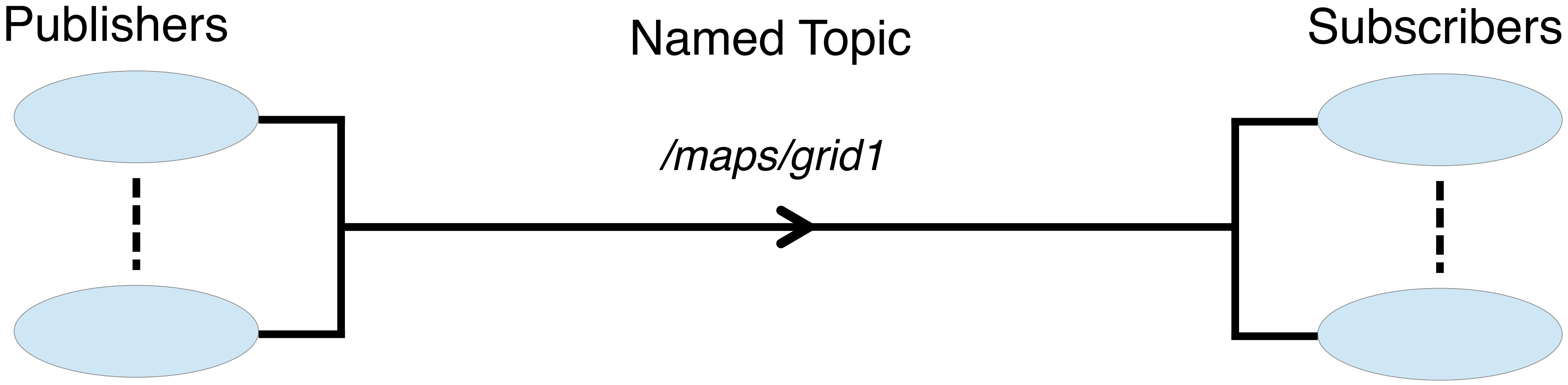
- Component-based software design put each function in its own module
- Need a communication mechanism between components

ROS (Robot Operating System)

- Open-source
- NOT an operating system:
  - Peer-to-peer comms for distributed processes (*nodes*).
  - Library of drivers, filters (e.g., mapping), behaviours (e.g., navigation)
- Not real-time
- OS agnostic (in theory, but only really works on Ubuntu)
- Language agnostic:
  - APIs for Python and C++ and other languages

ROS Basics

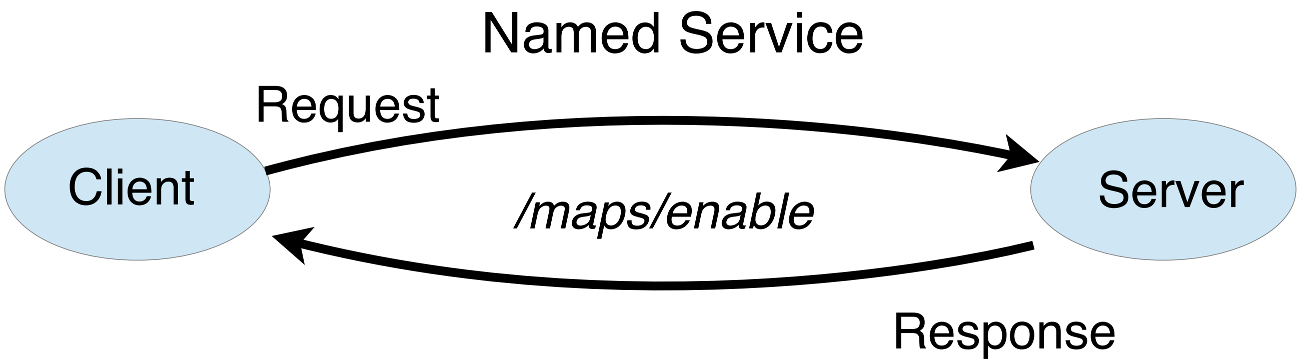
- ROS Nodes - registration at process startup
- Two models of comms between nodes:
  - ROS Topics: Publisher-subscriber (many-to-many).



*Commonly: one publisher and many subscribers

ROS Basics

- ROS Nodes - registration at process startup.
- Two models of comms between nodes:
  - ROS Topics: Publisher-subscriber (many-to-many).
  - ROS Services: remote procedure call (one-to-one).



ROS Basics

- ROS Nodes - registration at process startup
- Two models of comms between nodes:
  - ROS Topics: Publisher-subscriber (many-to-many)
  - ROS Services: remote procedure call (one-to-one)
- ROS *ActionLib*
  - Services with incremental feedback
  - built using ROS topics

Messages

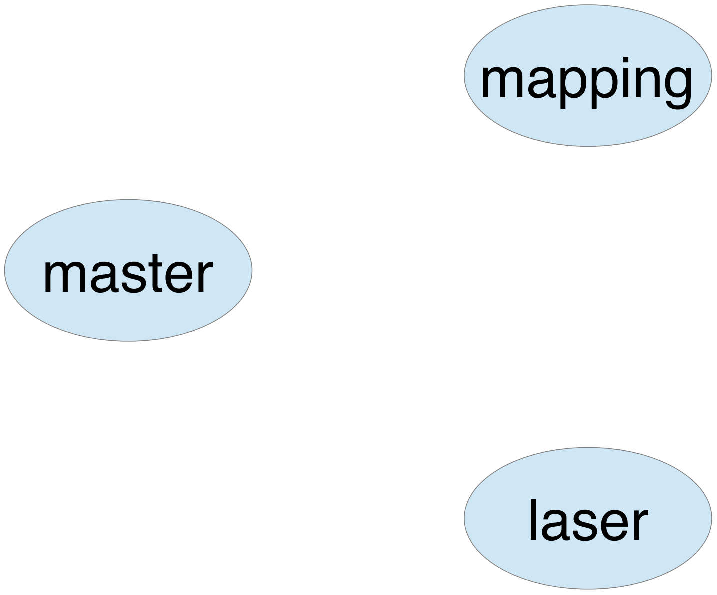
- Topics and services use a well-defined message format:
  - Primitive types (e.g., int8, bool, string, etc).
  - User-defined types (e.g., geometry_msgs/Point, sensor_msgs/LaserScan).
  - ROS takes care of generating language bindings (e.g., C++, Python).

geometry_msgs/Point

```
float64 x
float64 y
float64 z
```

Topic Setup

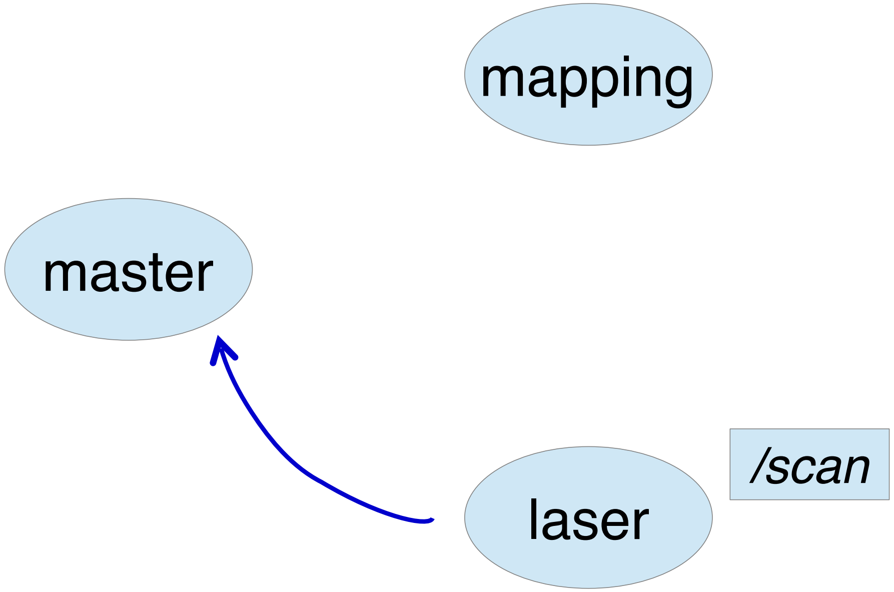
- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.



Topic Setup

- TCP/IP model - nodes can run on same or different computers
- ROS *master* provides directory services
- Scenario: *laser* node publishes and *mapping* node subscribes

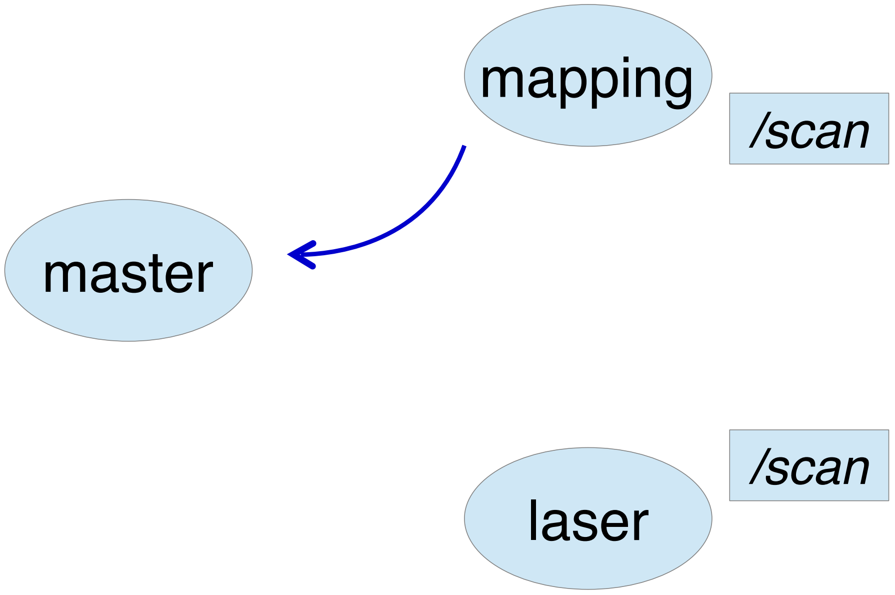
Laser node registers with master that it is publishing laser scans on a topic (with some name).



Topic Setup

- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.

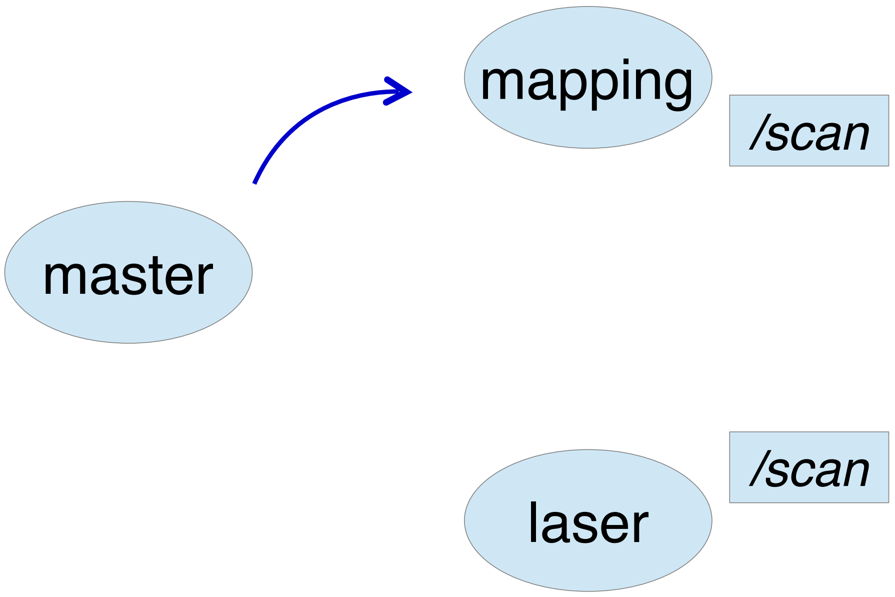
Mapping node registers with master that it is subscribing to the topic name.



Topic Setup

- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.

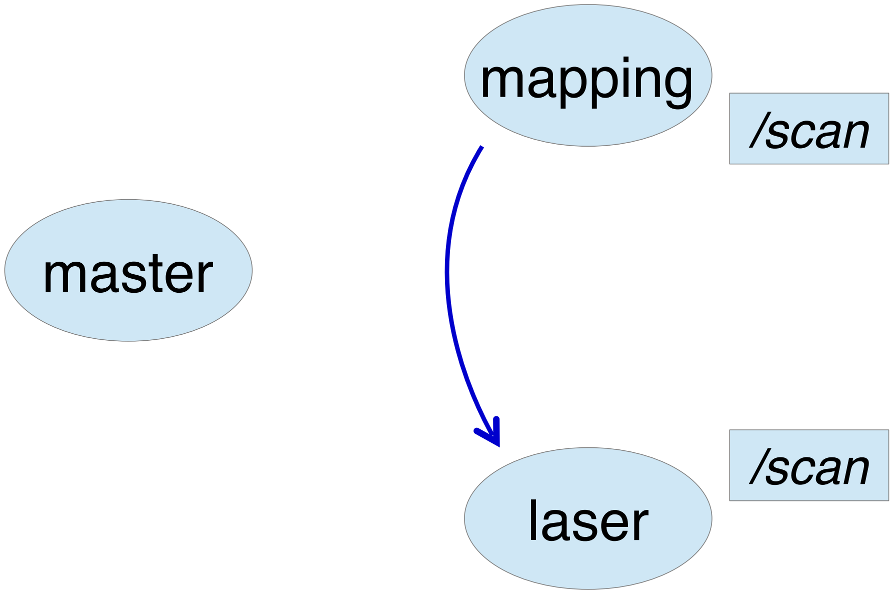
Master tells mapping node that the laser node is publishing the topic.



Topic Setup

- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.

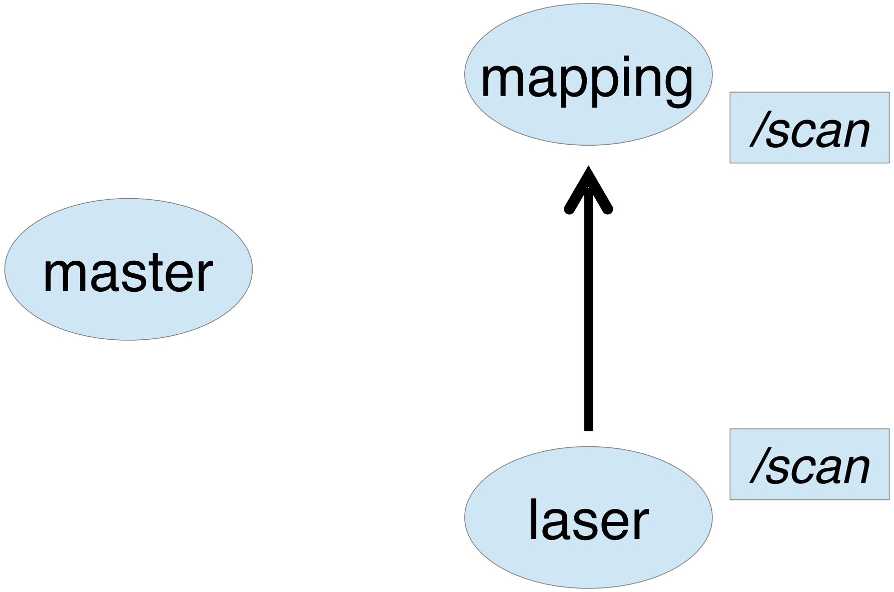
Mapping node initiates direct connection with laser node.



Topic Setup

- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.

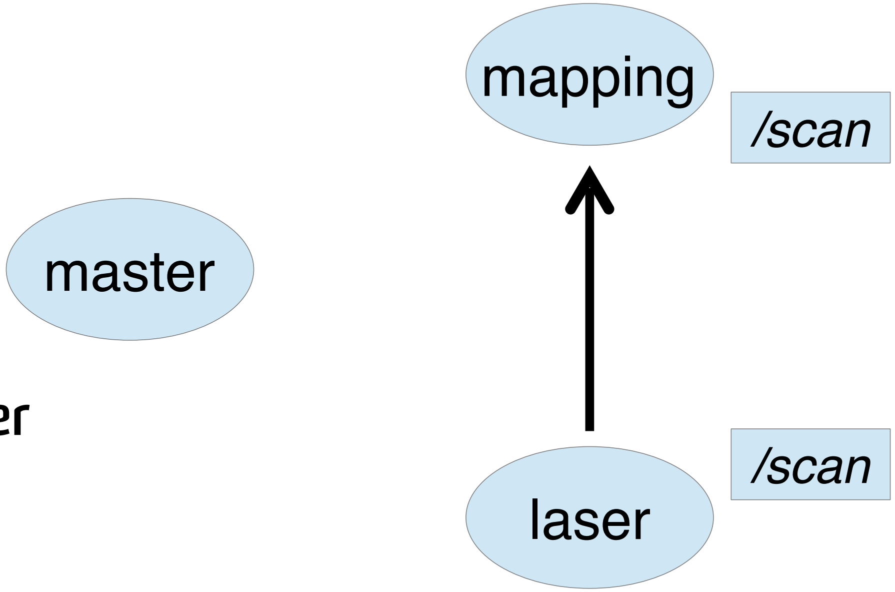
Laser node publishes and mapping node receives laser scan messages.



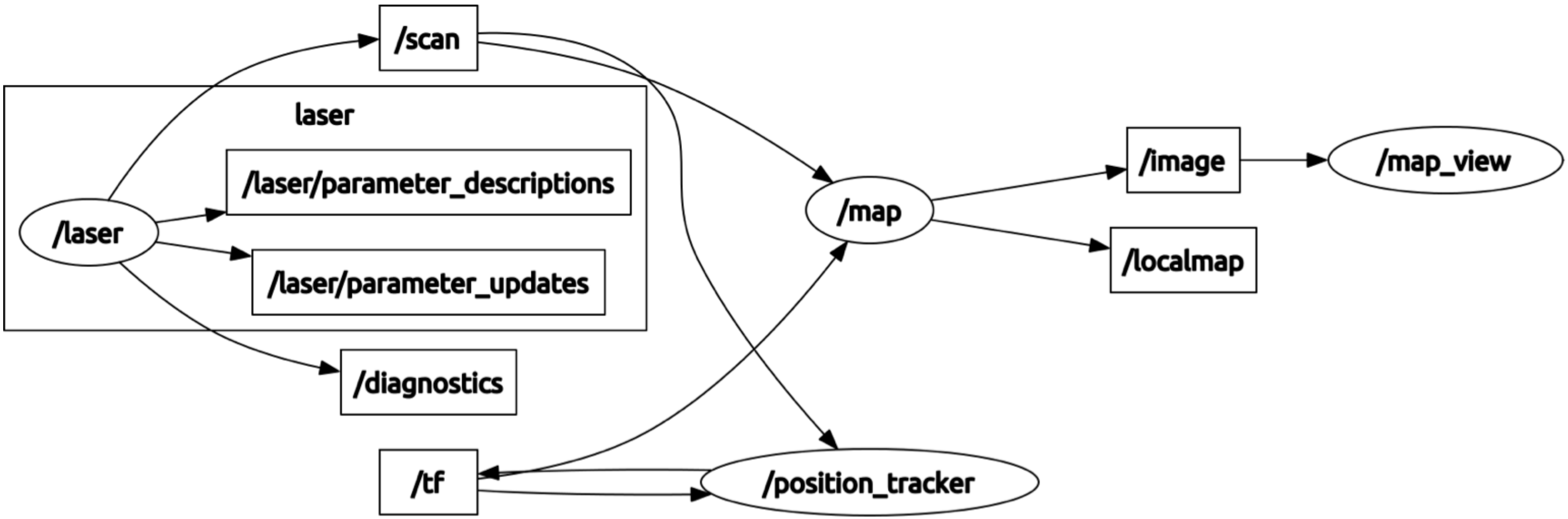
Topic Setup

- TCP/IP model - nodes can run on same or different computers.
- ROS *master* provides directory services.
- Scenario: *laser* node publishes and *mapping* node subscribes.

- NOTE: In reality a bit more complicated:
  - Laser node does not have to register first
  - Multiple publishers and multiple subscribers
  - But same outcome - **peer-to-peer data transfer**



Node/Topic Example



Nodes in a Distributed System

- Nodes can be on different computers.
- Requires some care:
  - Turn off local firewalls
  - Environment variables to specify addresses of nodes and master:
    - ROS_MASTER_URI - location of the master.
    - ROS_HOSTNAME - node will register with master using this value.
  - Safest to use IP addresses (not hostnames).

```
export ROS_MASTER_URI=http://192.168.1.2:11311
export ROS_HOSTNAME=192.168.1.5
```

IP Address of robot

Packages – Flexible Structure

- Dependencies to other packages.
- Custom *messages* and *service* definitions.
- Specify nodes - 0 or more.
- Libraries – export for use by other packages.

Catkin Workspaces

- Used for compiling and running a catkin system.
- Workspace layout:

```
catkin_ws/
        src/my_package/   - individual packages placed here
        build/
        devel/            - install location for development files
```

- Catkin tools are run within workspace directory.
- To compile your workspace:

```
$ cd catkin_ws
$ catkin_make
```

Catkin Packages

- *Catkin* – the ROS build system:
  - Combines *CMake* (popular C++ build tool) and some Python components.
- User-built components are organised in *packages*.
- A typical package:

```
mypackage/
        CMakeLists.txt   - CMake building
        package.xml      - dependencies between packages
        src/             - source directory: C++/Python/Java/etc
        include/         - typical for C++ headers
        scripts/         - typical for Python
        setup.py         - python installation file
```

- Use the Catkin tools: `catkin_create_pkg my_package depend1 ...`

Names and Namespaces - Warning

- ROS uses namespaces in different contexts.
- Positive: easy to avoid name clashes.
- Negative: can create confusion.
- Do not confuse namespace usage in:
    - Node names.
    - Topic names.
    - Frames of reference – to be discussed later.
- Node name “/mynode/laser” is different from frame “/mynode/laser”.

Laboratories

- Work through the ROS tutorials.
  - http://wiki.ros.org/ROS/Tutorials.
  - http://emanual.robotis.com/docs/en/platform/turtlebot3/overview
- First assignment:
  - due week 5.
  - Turtlebot3 navigation and recognition task.
  - Get started now!