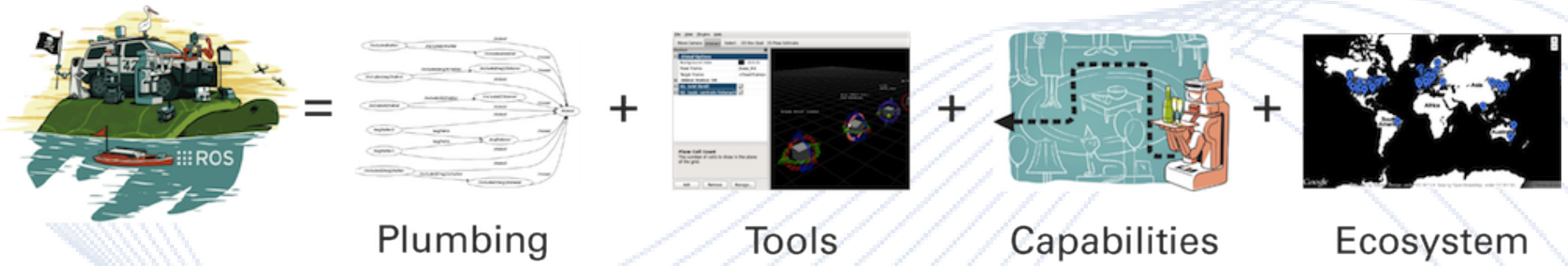
University of Applied Sciences

HOCHSCHULE EMDEN·LEER

Mobile Robotics

Prof. Dr.-Ing. Gavin Kane



University of Applied Sciences

HOCHSCHULE EMDEN·LEER

Robot Operating System (ROS)

Course Content

- Lecture 1 (Online) - ROS Overview
  - Practical 1 - ROS Installation
- Lecture 2 - Publishers, Subscribers, Services, Action Servers
  - Practical 2 - Publishers and Subscribers
- Lecture 3 - URDF (Unified Robot Description Format)
  - Practical 3 - Building our own Robot
- Lecture 4 - ROS Modules (TF und MoveIT)
  - Practical 4 - Getting our Robot to move
- Lecture 5 - Modules, SLAM and Navigating
  - Practical 5 - Getting our Robot to move to where we want it to!

Robot Operating System (ROS)

Lecture Content

- ROS Overview
- Philosophy
- Communications
- Organisation
- Packages
- Nodes / Topics / Messages

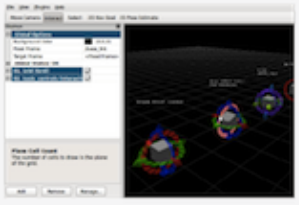
Robot Operating System (ROS)

What is ROS?

ROS = Robot Operating System

=

+



+

+

Plumbing

- Process management
- Interprocess communication
- Device Drivers

Tools

- Simulation
- Visualization
- Graphical user interface
- Data logging

Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

Ecosystem

- Package organization
- Software distribution
- Documentation
- Tutorials

Robot Operating System (ROS)

History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- De facto standard for robot programming
- Additionally since 2012 ROS-I consortium:

www.ros.org

Robot Operating System (ROS)

ROS Philosophy

- **Peer to peer**
  Individual programs communicate over defined API (ROS messages, services, etc.).
- **Distributed**
  Programs can be run on multiple computers and communicate over the network.
- **Multi-lingual**
  ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- **Light-weight**
  Stand-alone libraries are wrapped around with a thin ROS layer.
- **Free and open-source**
  Most ROS software is open-source and free to use.

Robot Operating System (ROS)

Communications Infrastructure

- At the lowest level, ROS offers a message passing interface that provides inter-process communication and is commonly referred to as a middleware.
- The ROS middleware provides these facilities:
  - publish/subscribe anonymous message passing
  - recording and playback of messages
  - request/response remote procedure calls
  - distributed parameter system

Robot Operating System (ROS)

Robot-Specific Features

- In addition to the core middleware components, ROS provides common robot-specific libraries and tools that will get your robot up and running quickly.
  - Standard Message Definitions for Robots
  - Robot Geometry Library
  - Robot Description Language
  - Preemptable Remote Procedure Calls
  - Diagnostics
  - Pose Estimation
  - Localization
  - Mapping
  - Navigation

Robot Operating System (ROS)

ROS Tools

- One of the strongest features of ROS is the powerful development toolset. These tools support:
  - introspecting,
  - debugging,
  - plotting,
  - visualization
  - etc.
- The underlying publish/subscribe mechanism allows you to spontaneously introspect the data flowing through the system, making it easy to comprehend and debug issues as they occur. The ROS tools take advantage of this introspection capability through an extensive collection of graphical and command line utilities that simplify development and debugging.

Robot Operating System (ROS)

ROS as Opensource Project

- Organised through the OSRF, ROS is being continially updated. Older Versions including Jade, Indigo, Hydro, Groovy, Fuerto and more. Current Versions are:
  - Kinetic Kame. Released May 2016 with long term support until April 2021

  - Melodic Morenia. Released May 2018 with long term support until May 2023

Robot Operating System (ROS)

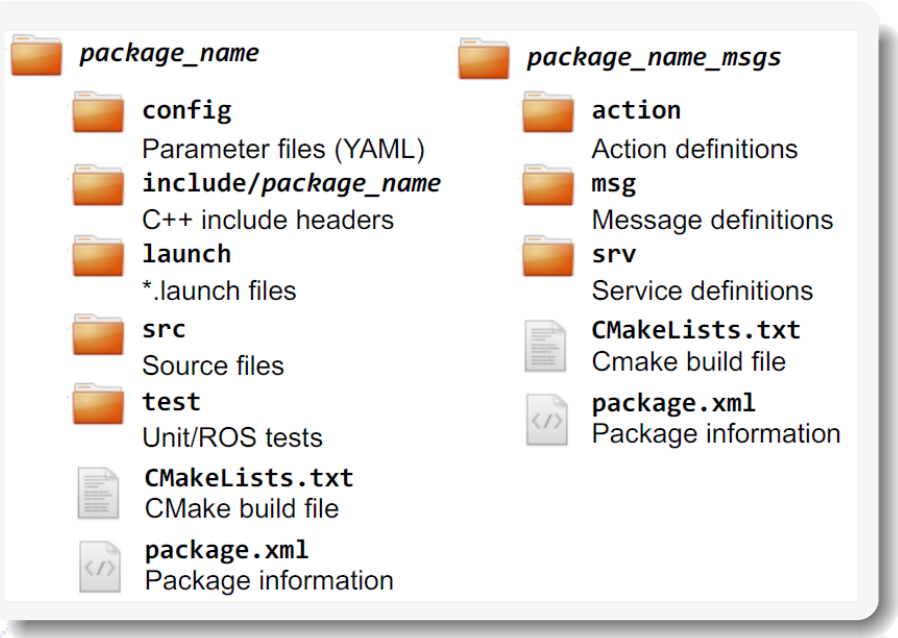
Organisation - Workspace Directory Structure

```
/catkin_ws ..... Workspace Base Directory
├── devel ..... Location of setup.bash File
├── build
└── src ..... Location where Packages are placed
```

Robot Operating System (ROS)

ROS Packages

- ROS software is organized into packages, which can contain source code, launch files, configuration files, message definitions, data, and documentation
- A package that builds up on/requires other packages (e.g. message definitions), declares these as dependencies



Robot Operating System (ROS)

ROS Packages

package.xml

- The package.xml file defines the properties of the package
  - Package name
  - Version number
  - Authors
  - Dependencies on other packages

package.xml

```
<?xml version="1.0"?>
 <package format="2">
 <name>ros\_package\_template</name>
 <version>0.1.0</version>
 <description>A template for ROS
    packages.</description>
 <maintainer email="somebody@e...">
    Somebody</maintainer>
 <license>BSD</license>
 <url type="website">https://github.com/...
    </url>
 <author email="Somebody@somewhere.com">
    Somebody</author>
 <buildtool\_depend>catkin
    </buildtool\_depend>
 <depend>roscpp</depend>
 <depend>sensor\_msgs</depend>
</package>
```

Robot Operating System (ROS) - Communications Infrastructure

Terminology - ROS Master

- Manages the communication between nodes
- Every node registers at startup with the master

Start a master in MATLAB with:

```
>> rosinit
```

Prepare MATLAB to connect with a master at a given IP Address:

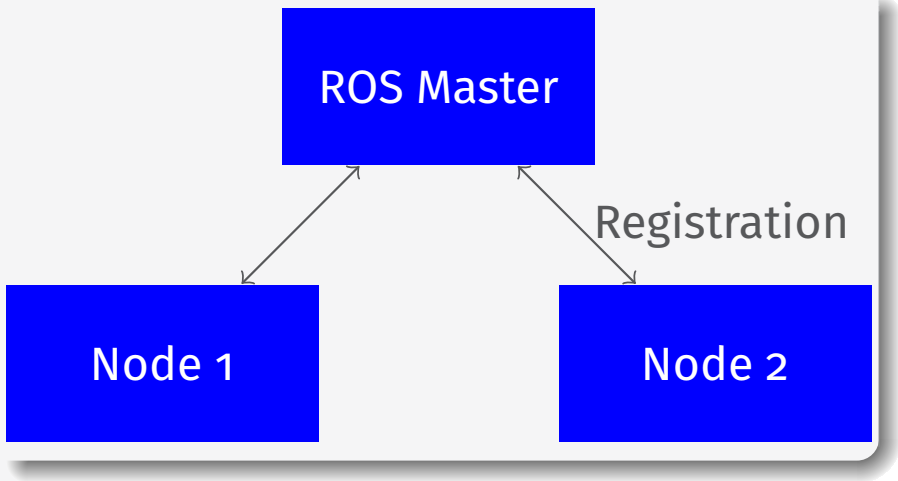
```
>> rosinit('192.168.154.131')
```

Start a master in Linux with:

```
gavin@comp:~$ roscore
```

Prepare Linux to connect with a master on another computer:

```
gavin@comp:~$ export ROS_MASTER_URI=http://min:11311/
```



Robot Operating System (ROS) - Communications Infrastructure

Terminology - ROS Nodes

- Single-purpose, executable program
- Individually compiled, executed, and managed
- Organized in packages

Run a node in MATLAB:

```
>> N = ros.Node(Name)
>>
```

Prepare to run a node in Linux:

```
gavin@comp:~$rosrun <package_name> <node_name>
```

Robot Operating System (ROS) - Communications Infrastructure

Getting information - ROS Nodes

To gain information about what nodes are running, you can use in a Linux Terminal or in Matlab:

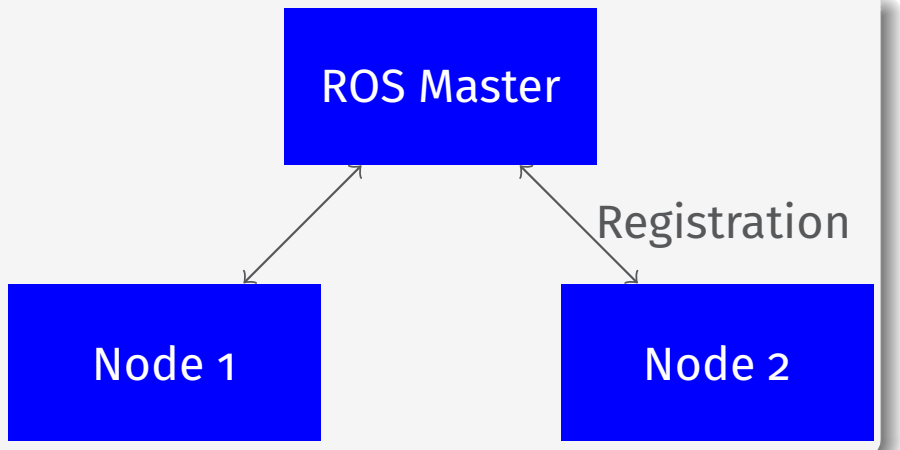
```
gavin@comp:~$ rosnode list
```

or to get a visual view of the nodes and how they are communicating:

```
gavin@comp:~$ rqt_graph
```

To gain information about a specific node:

```
gavin@comp:~$ rosnode info <node_name>
```



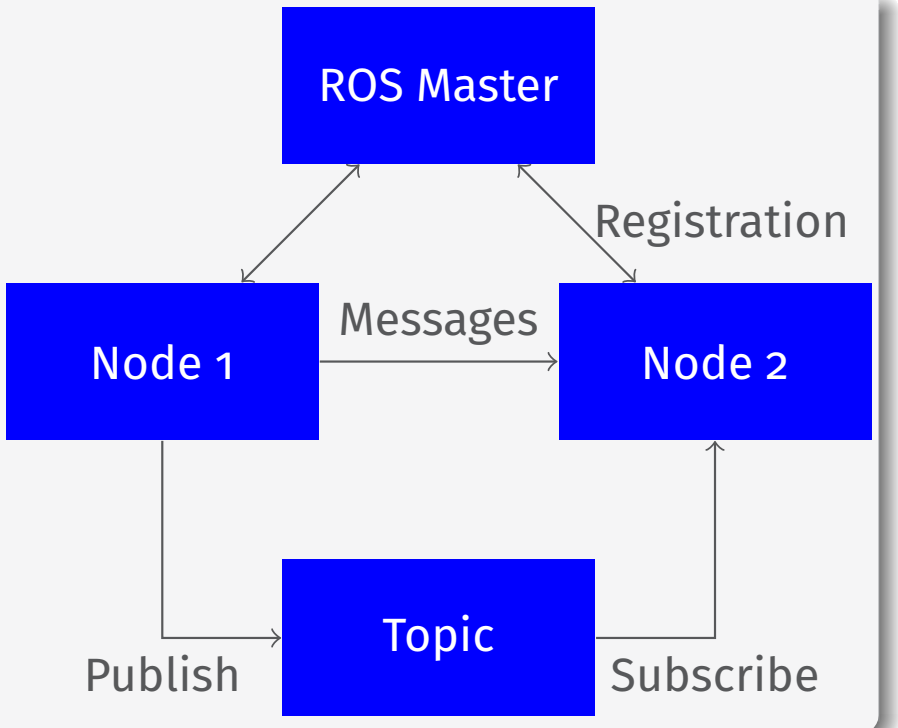
Robot Operating System (ROS) - Communications Infrastructure

Terminology - ROS Topics

- Nodes communicate over topics
- Nodes can publish or subscribe to a topic
- Typically, 1 publisher and n subscribers
- Topic is a name for a stream of messages

List active topics with:

```
>> rostopic list
oder
gavin@comp:~$ rostopic list
```



Robot Operating System (ROS) - Communications Infrastructure

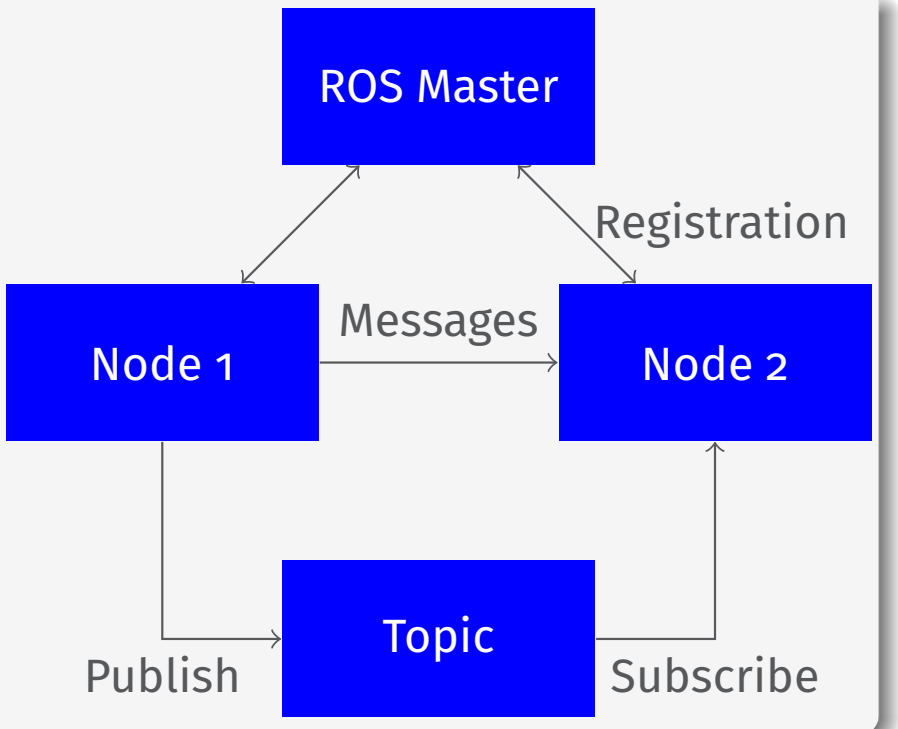
Terminology - ROS Topics

Subscribe and print the contents of a topic with

```
>> rostopic echo /topic
gavin@comp:~$ rostopic echo /topic
```

Show information about a topic with

```
>> rostopic info /topic
gavin@comp:~$ rostopic info /topic
```



Robot Operating System (ROS) - Communications Infrastructure

Terminology - ROS Messages

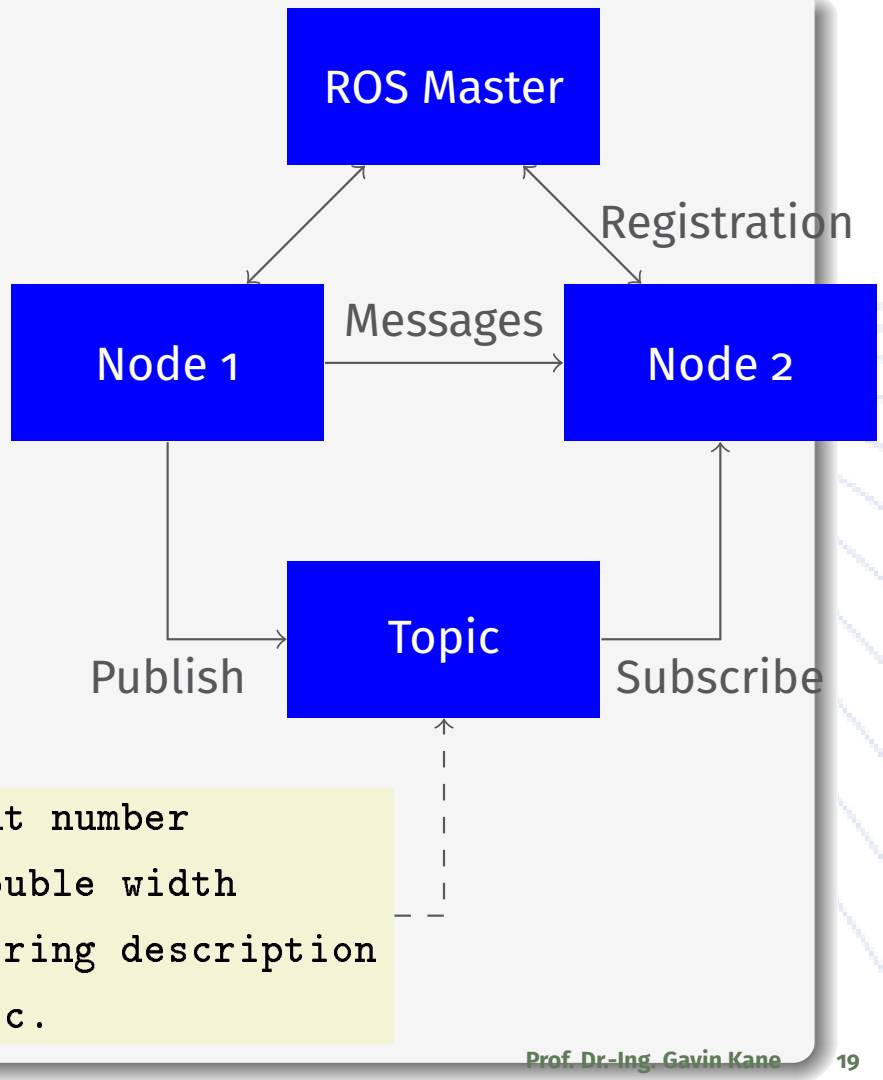
- Data structure defining the type of a topic
- Compromised of a nested structure of integers, floats, booleans, strings etc. and arrays of objects
- Defined in *.msg files

See the type of a topic

```
>> rostopic type /topic
oder
gavin@comp:~$ rostopic type /topic
```

or in matlab

```
>> msg = rosmessage(msgtype);
```



Robot Operating System (ROS)

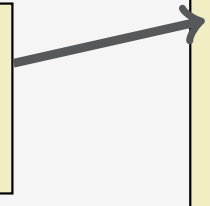
ROS Messages - Pose Stamped Example

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
 uint32 seq
 time stamp
 string frame_id
geometry_msgs/Pose pose
 geometry_msgs/Point position
 float64 x
 float64 y
 float64 z
 geometry_msgs/Quaternion orientation
 float64 x
 float64 y
 float64 z
 float64 w
```

geometry_msgs/Point.msg

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



Robot Operating System (ROS)

Example Publisher Startup and Message Send (Terminal)

Terminal 1:

```
gavin@comp:~$ roscore
... logging to /home/gavin/.ros/log/d450f102-6f7d-11e9-85d9-
                    305a3a0046ba/roslaunch-TheBrain-130.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://TheBrain:57151/

SUMMARY
========
PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.3

NODES
auto-starting new master
process[master]: started with pid [140]
ROS_MASTER_URI=http://TheBrain:11311/
setting /run_id to d450f102-6f7d-11e9-85d9-305a3a0046ba
```

Robot Operating System (ROS)

Example Publisher Startup and Message Send (Terminal)

Terminal 2:

```
gavin@comp:~$ rosnode list
/rosout
gavin@comp:~$ rostopic list
/rosout
gavin@comp:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist
                              -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
publishing and latching message for 3.0 seconds
```

Terminal 3: (during 3 second transmission)

```
gavin@comp:~$ rosnode list
/rosout
/rostopic_449_1557092476832
gavin@comp:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
```

Robot Operating System (ROS)

Example Publisher Startup and Message Send (Terminal)

To display information about publishing or subscribing node:
Terminal 3:

```
gavin@comp:~$ rosnode info /rostopic_449_1557092476832
--------------------------------------------------------
Node [/rostopic_449_1557092476832]
Publications:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Subscriptions: None

Services:
 * /rostopic_449_1557092476832/get_loggers
 * /rostopic_449_1557092476832/set_logger_level


contacting node http://comp:57529/ ...
Pid: 480
```

Robot Operating System (ROS)

Example Publisher Startup and Message Send (Terminal)

Terminal 4:

```
gavin@comp:~$ rostopic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.8
---
```

New Data will be echoed to terminal as it is received

Robot Operating System (ROS)

Example Publisher Startup and Message Send (MATLAB)

```
>> rosinit
Initializing ROS master on http://ctl:11311/.
Initializing global node /matlab\_global\_node\_98146 with NodeURI
    http://ctl:50407/
>> rostopic list
/rosout
>> chatpub = rospublisher('/chatter','std\_msgs/String');
>> rostopic list
/chatter
/rosout
>> msg = rosmessage(chatpub);
>> msg.Data = 'Hello to Subscriber';
```

Robot Operating System (ROS)

Example Subscriber Startup

```
>> rosinit('192.168.178.32')
Initializing global node /matlab\_global\_node\_98146 with NodeURI
    http://192.168.178.32:53065/
>> rostopic list
/chatter
/rosout
>> sub = rossubscriber('/chatter',@myROSCallbackFcn)

sub =

  Subscriber with Properties:
    TopicName: '/chatter'
    MessageType: 'std\_msgs/String'
    LatestMessage: [0x1 String]
    Buffersize: 1
    NewMessageFcn: @myROSCallbackFcn

>>
```

Robot Operating System (ROS)

Working with Specialized ROS Messages - Laser Scan Messages

- Laser scanners are commonly used sensors in robotics. You can see the standard ROS format for a laser scan message by creating an empty message of the appropriate type.
- An example Scan Message:

```
scan =

  ROS LaserScan message with properties:

        MessageType: 'sensor\_msgs/LaserScan'
             Header: [1x1 Header]
           AngleMin: -0.5467
           AngleMax: 0.5467
     AngleIncrement: 0.0017
      TimeIncrement: 0
           ScanTime: 0.0330
           RangeMin: 0.4500
           RangeMax: 10
             Ranges: [640x1 single]
        Intensities: [0x1 single]
```

Robot Operating System (ROS)

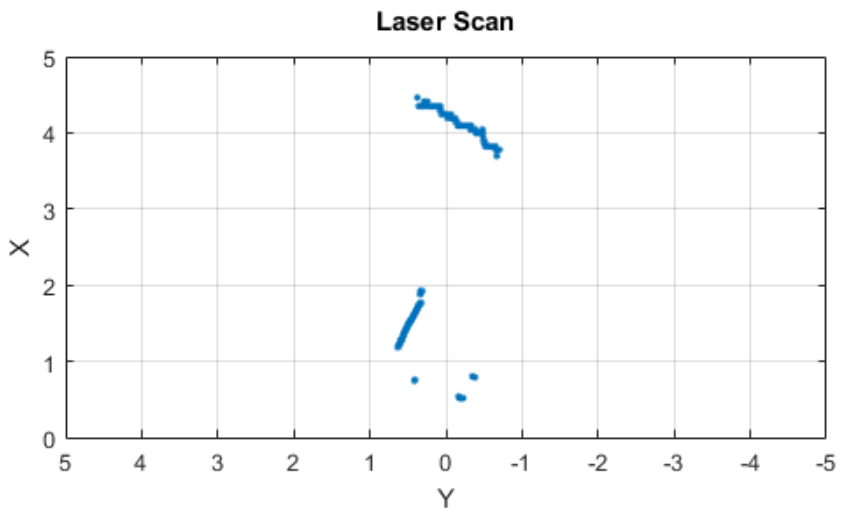
Working with Specialized ROS Messages - Laser Scan Messages

- Special Operators exist for performing polar to cartesian transformations:

```
xy = readCartesian(scan)

xy =

    0.7886   -0.3803
    0.7887   -0.3786
    0.7907   -0.3780
    ...
```

- and visualisations:

```
figure
plot(scan,'MaximumRange',5)
```



Robot Operating System (ROS)

Working with Specialized ROS Messages - Image Messages

- MATLAB also provides support for image messages, which always have the message type sensor_msgs/Image.

```
img =

  ROS Image message with properties:

    MessageType: 'sensor\_msgs/Image'
         Header: [1x1 Header]
         Height: 480
          Width: 640
       Encoding: 'rgb8'
    IsBigendian: 0
           Step: 1920
           Data: [921600x1 uint8]
```

- To view the image:

```
imageFormatted = readImage(img);
figure
imshow(imageFormatted)
```

Robot Operating System (ROS)

Working with Specialized ROS Messages - Point Cloud Messages

- Point clouds can be captured by a variety of sensors used in robotics, including LIDARs, Kinect® and stereo cameras. The common message type is sensor_msgs/PointCloud2.

```
ptcloud =

  ROS PointCloud2 message with properties:

    PreserveStructureOnRead: 0
                MessageType: 'sensor_msgs/PointCloud2'
                     Header: [1×1 Header]
                     Height: 480
                      Width: 640
                IsBigendian: 0
                  PointStep: 32
                    RowStep: 20480
                    IsDense: 0
                     Fields: [4×1 PointField]
                       Data: [9830400×1 uint8]
```

Robot Operating System (ROS)

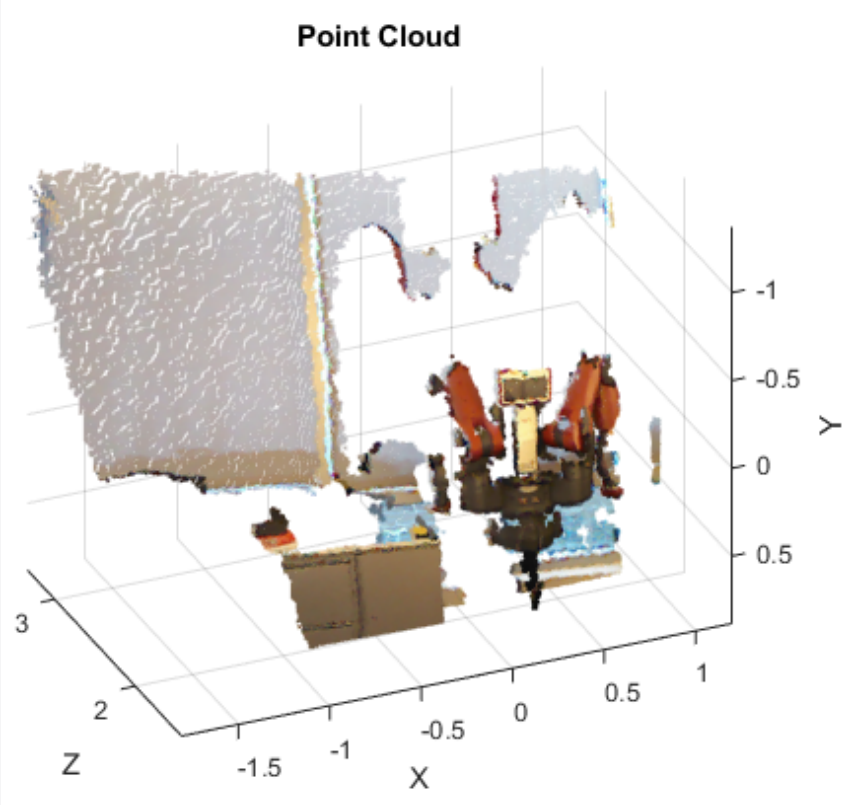
Working with Specialized ROS Messages - Point Cloud Messages

- The point cloud information is encoded in the Data field of the message. You can extract the [x,y,z] coordinates as an N-by-3 matrix by calling the readXYZ function.

```
xyz = readXYZ(ptcloud)

xyz =

  193359×3 single matrix

    0.1378   -0.6705    1.6260
    0.1409   -0.6705    1.6260
    0.1433   -0.6672    1.6180
    ...
```

- To visualise the point cloud:

```
figure
scatter3(ptcloud)
```



Robot Operating System (ROS)

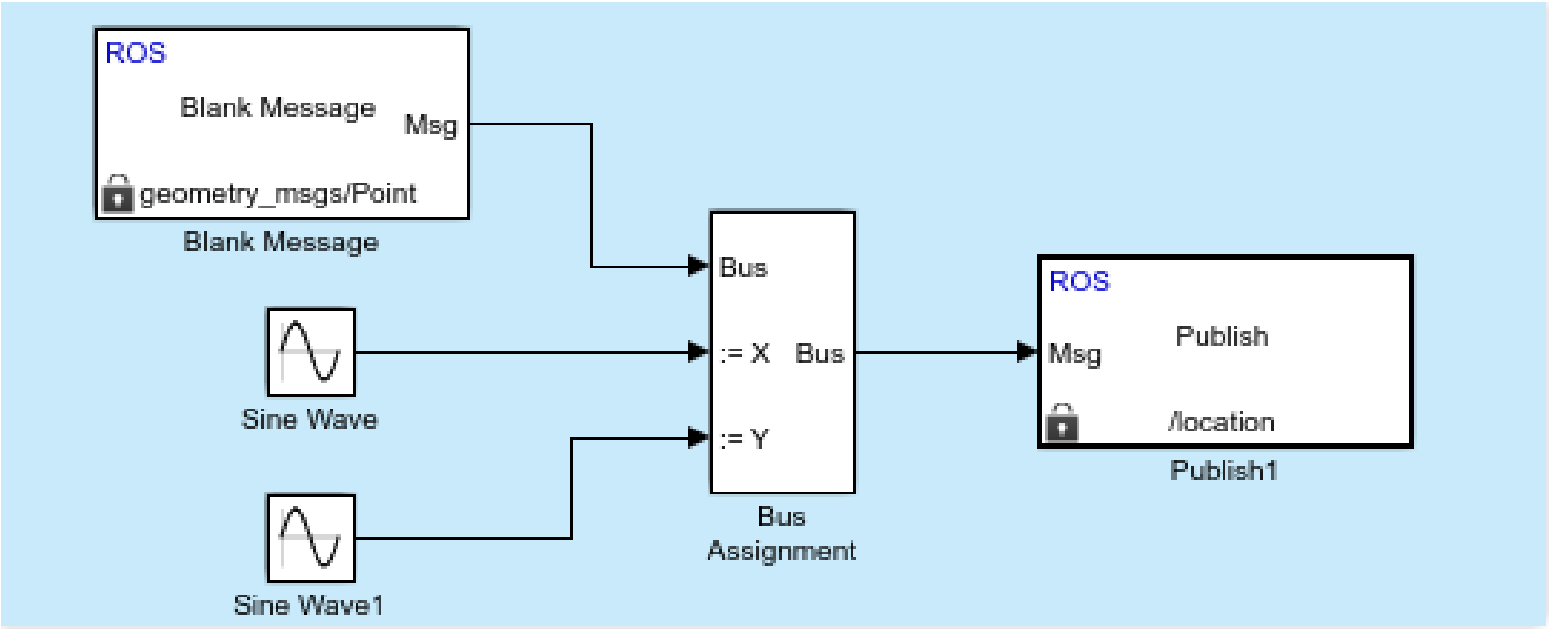
Custom ROS Messages in Matlab / Simulink

- Custom ROS Messages can be created with the Command:
  `» rosgenmsg(folderpath)`
- In the given folder, custom message definitions and service definitions can be found as .msg and .srv files.
- The directory requires a package.xml file to define its contents

Robot Operating System (ROS)

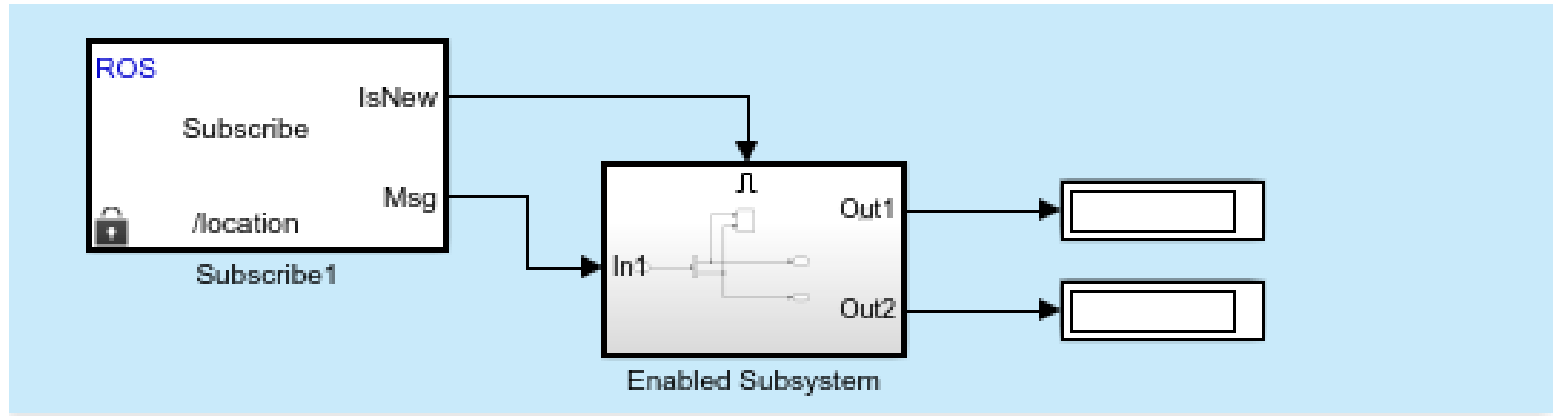
Publish Messages in Simulink

After connecting to the ROS Network in the Command Line, it is possible to use the ROS Blocks in Simulink.



Robot Operating System (ROS)

Subscribe to Messages in Simulink

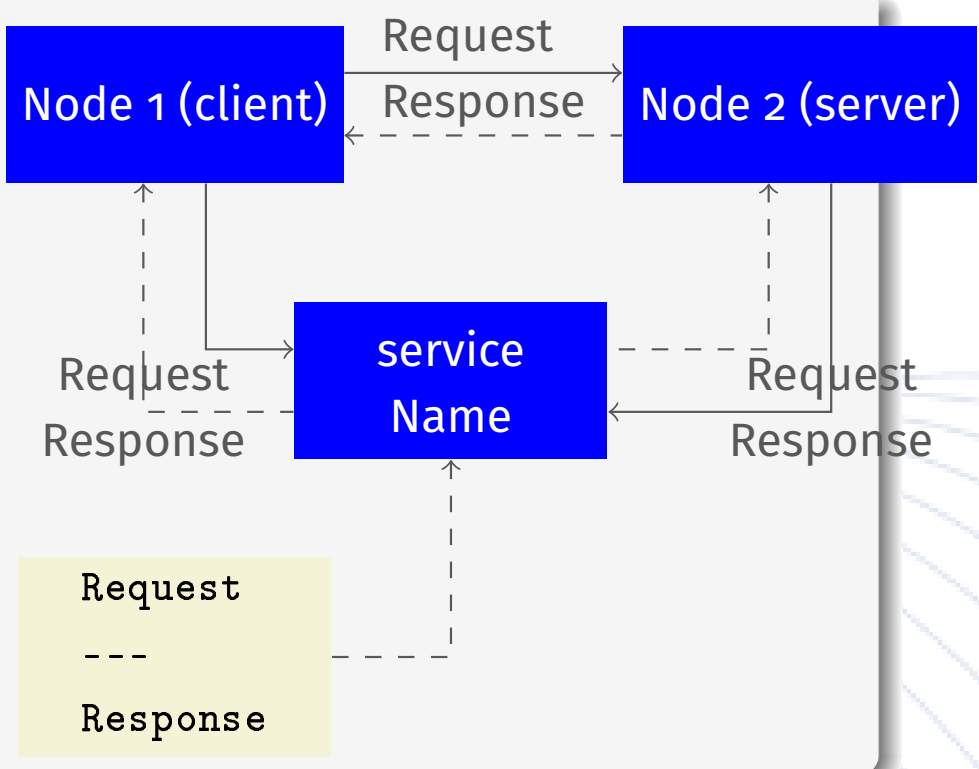


Robot Operating System (ROS)

ROS Services

- Request/response communication between nodes is realized with services
- The service server advertises the service
- The service client accesses this service
- Similar in structure to messages, services are defined in *.srv files
- List available services with

```
>> rosservice list
```



Robot Operating System (ROS)

ROS Services

To Create a Service:

```
testserver = rossvcserver('/test', 'std_srvs/Empty', @myROSCallback)
```

The required structure of the callback:

```
function resp = exampleHelperROSEmptyCallback(~,~,resp)
% Perform Task
end
```

The client then needs to be created, directed at the server

```
 testclient = rossvcclient('/test2')
```

In order to call the service, the client uses prepares its request message, and calls the server:

```
testreq = rosmessage(testclient)
testresp = call(testclient,testreq,'Timeout',3);
```

Robot Operating System (ROS)

ROS Actions

- Similar to service calls, but provide possibility to
  - Cancel the task (preempt)
  - Receive feedback on the progress
- Best way to implement interfaces to timeextended, goal-oriented behaviors
- Internally, actions are implemented with a set of topics
- Actions are not implemented in MATLAB

Robot Operating System (ROS)

Building ROS Nodes

- Tool to be used is either catkin_make or catkin build
- The workspace will be prepared with catkin init, but first the ROS installation must be sourced.
- This could also be inserted into your /.bashrc file

```
gavin@comp:~$ source /opt/ros/melodic/setup.bash
gavin@comp:~$ mkdir catkin_ws
gavin@comp:~$ cd catkin_ws
gavin@comp:~$ catkin init
Initializing catkin workspace in ‘/home/gavin/catkin_ws‘.
------------------------------------------------------------
Profile:                     default
Extending:             [env] /opt/ros/melodic
Workspace:                   /home/gavin/catkin_ws
------------------------------------------------------------
Source Space:       [missing] /home/gavin/catkin_ws/src
Log Space:          [missing] /home/gavin/catkin_ws/logs
Build Space:        [missing] /home/gavin/catkin_ws/build
Devel Space:        [missing] /home/gavin/catkin_ws/devel
Install Space:       [unused] /home/gavin/catkin_ws/install
DESTDIR:             [unused] None
------------------------------------------------------------
```

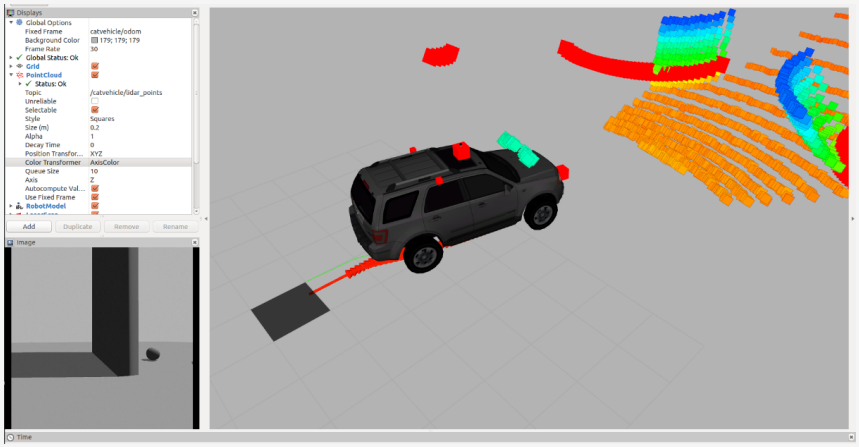
Robot Operating System (ROS)

Demo

Robot Operating System (ROS) - Tools

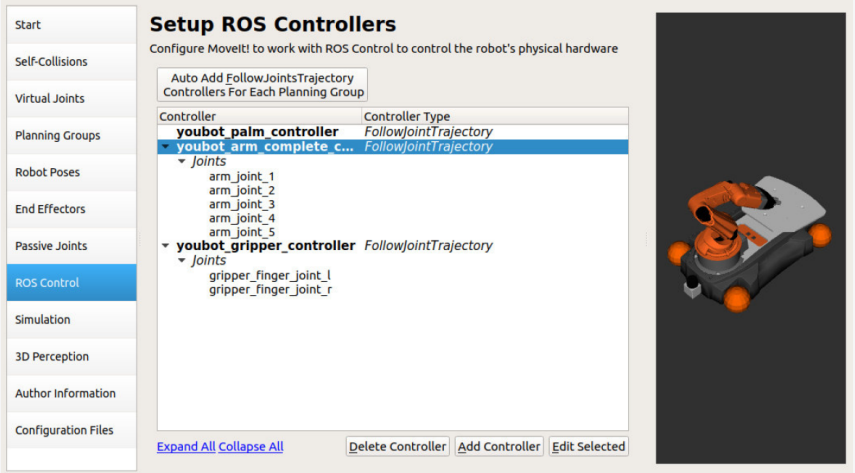
RVIZ

- (ROS visualization) is a 3D visualizer for displaying sensor data and state information from ROS. Using rviz, you can visualize Baxter's current configuration on a virtual model of the robot.



MoveIT

- Motion Planning Framework. Includes Tools such as the "Rviz Motion Planning Plugin" and "MoveIt Setup Assistant".



Robot Operating System (ROS)

Summary

- Robot Operating System, is not an operating system, it is a collection of tools and capabilities that are built around a very powerful commuinications system and supported by a large opensource community
- This lecture has only touched on one small slice
- Publishers, Subscribers, Servers and Clients can be implemented in C, Python, Java MATLAB and Simulink
- Cross platform and Cross Language Development also very easy to implement
- Reference: ROS Wiki http://wiki.ros.org/