Robot Operating System (ROS)

Lab 1: Introduction to ROS

Haitham El-Hussieny, PhD

October 10, 2022

Department of Mechatronics and Robotics Engineering
Egypt-Japan University of Science and Technology (E-JUST)
Alexandria, Egypt.

WHY ROS NOW!

Principal Applied Scientist, Robotics & Perception - Amazon Scout
Amazon
South San Francisco, CA

Preferred Qualifications

- 10+ years of experience conducting R&D in computer vision and machine learning
- 5+ years of experience leading R&D projects that include highly talented teams responsible for developing and owning complex algorithms that go beyond the state of the art.
- 5+ years of experience managing R&D teams.
- 5+ years of experience shipping computer vision based products.
- Experience in the design, development, evaluation, deployment and updating of multiple robotic based computer vision and deep learning solutions.
- Experience in creating the right culture and environment to ensure a diverse group of people and skillsets can flourish and deliver true breakthrough science and innovation
- Experience in influencing and collaborating effectively across all levels of the organization.
- Experience with ROS
- Experience with computer vision libraries such as PCL, OpenCV
- Experience in CPU / GPU performance analysis and optimization

Amazon is an Equal Opportunity-Affirmative Action Employer - Female/Minority/Disability/Veteran/Gender Identity/Sexual Orientation.

PS Engineering - Robotics
Procter & Gamble
Oxnard, CA (On-site)
Medical benefit

Qualifications

REQUIRED:

- BS or MS degree in Electrical Engineering, Computer Science, Robotics, Mechanical, or Software Engineering.
- These positions are entry-level with up to 4 years work experience.

PREFERRED:

- Proven experience in the Robotics Engineering field
- Engineering experience in design, coding, testing, deployment and support.
- CAD design and robotics simulation.
- ROS (Robot Operating Systems) open source programming.
- computer programming languages, such as Python and C++, Linux operating systems.
- Experience in computer vision technology and industrial vision systems such as Cognex, National Instrument and Keyence.
- Experience with industrial PLC, especially with Rockwell Control Logix platforms.
- Experience with computer and industrial networks.
- Experience with robotics and autonomous mobile vehicle safety requirements and validation.

THE CHALLENGE!

Suppose you are working on a KUKA R820 iiwa Robot for years.

End-effector moving (Java)

```
CartesianPTP ptpMove = ptp(destFrame);
ptpMove.setJointJerkRel(0.1).setJointVelocityRel(0.5);
toolAttachedToLBR.move(ptpMove);
```

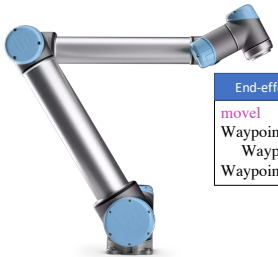
THE CHALLENGE!

Suppose you are working on a KUKA R820 iiwa Robot for years.

End-effector moving (Java)

```
CartesianPTP ptpMove = ptp(destFrame);
ptpMove.setJointJerkRel(0.1).setJointVelocityRel(0.5);
toolAttachedToLBR.move(ptpMove);
```

Now, you need to move to use UR10 robot, or use a ready avaiable code built for UR10.



End-effector moving (CB3)

```
movel
WaypointStart
    Waypoint1
WaypointEnd1
```

THE CHALLENGE!

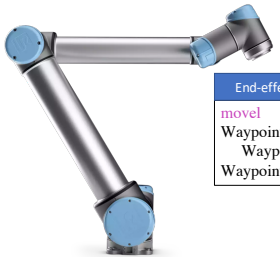
Suppose you are working on a KUKA R820 iiwa Robot for years.

End-effector moving (Java)

```
CartesianPTP ptpMove = ptp(destFrame);
ptpMove.setJointJerkRel(0.1).setJointVelocityRel(0.5);
toolAttachedToLBR.move(ptpMove);
```

Now, you need to move to use UR10 robot, or use a ready avaiable code built for UR10.

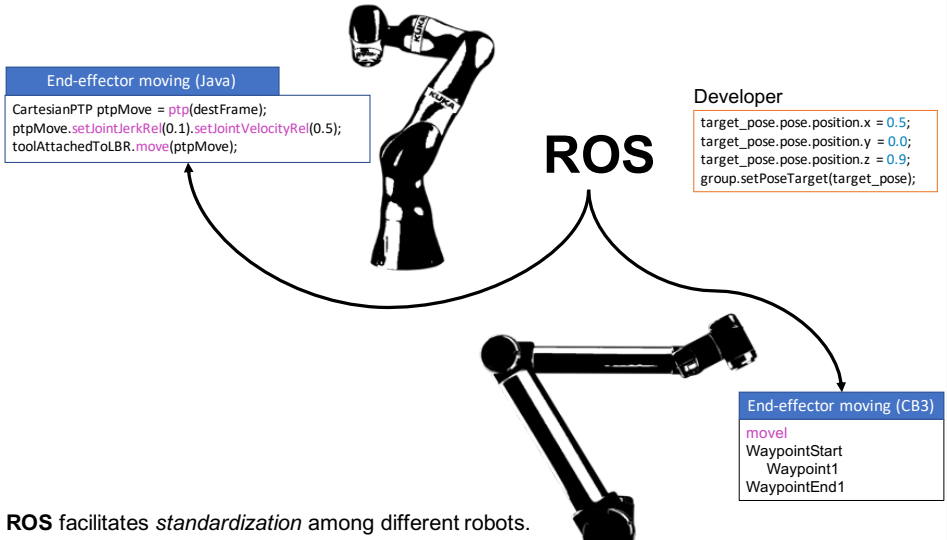
Challenge: No standards. Time consuming for knowledge transfer!



End-effector moving (CB3)

```
movel
WaypointStart
    Waypoint1
WaypointEnd1
```

ROBOT OPERATING SYSTEM (ROS):



ROS facilitates *standardization* among different robots.

OUTLINE

1. What is and why ROS?
2. ROS architecture and philosophy.
3. ROS master, nodes, topics, and messages.

What is and why ROS?

What is and why ROS?

Robot Operating System (ROS):

The Robot Operating System (ROS) is a set of open-source **software libraries and tools** that help you build robot applications.

WHAT IS AND WHY ROS?

ROBOT OPERATING SYSTEM (ROS):

The Robot Operating System (ROS) is a set of open-source **software libraries and tools** that help you build robot applications.

Plumbing

- Process management.
- Inter-process communication.
- Device drivers.

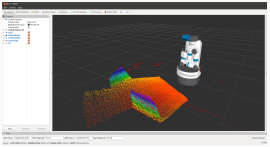
WHAT IS AND WHY ROS?

ROBOT OPERATING SYSTEM (ROS):

The Robot Operating System (ROS) is a set of open-source **software libraries and tools** that help you build robot applications.

Plumbing

- Process management.
- Inter-process communication.
- Device drivers.



Tools

- Simulation
- Visualization
- Data logging

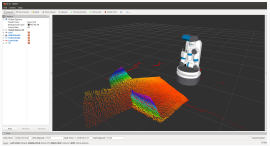
WHAT IS AND WHY ROS?

ROBOT OPERATING SYSTEM (ROS):

The Robot Operating System (ROS) is a set of open-source **software libraries and tools** that help you build robot applications.

Plumbing

- Process management.
- Inter-process communication.
- Device drivers.



Tools

- Simulation
- Visualization
- Data logging

Capabilites

- Control
- Planning
- Perception
- Mapping
- Manipulation

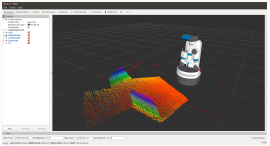
What is and why ROS?

Robot Operating System (ROS):

The Robot Operating System (ROS) is a set of open-source **software libraries and tools** that help you build robot applications.

Plumbing

- Process management.
- Inter-process communication.
- Device drivers.



Tools

- Simulation
- Visualization
- Data logging

Capabilites

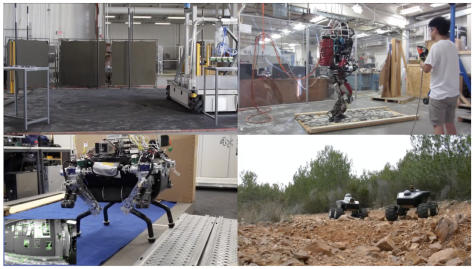
- Control
- Planning
- Perception
- Mapping
- Manipulation

Community

- Packaging
- Software distribution
- Documentation
- Tutorials

History of ROS.

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory.
- Since 2013 managed by OSRF (open robotics).
- Today used by many robots, universities and companies.



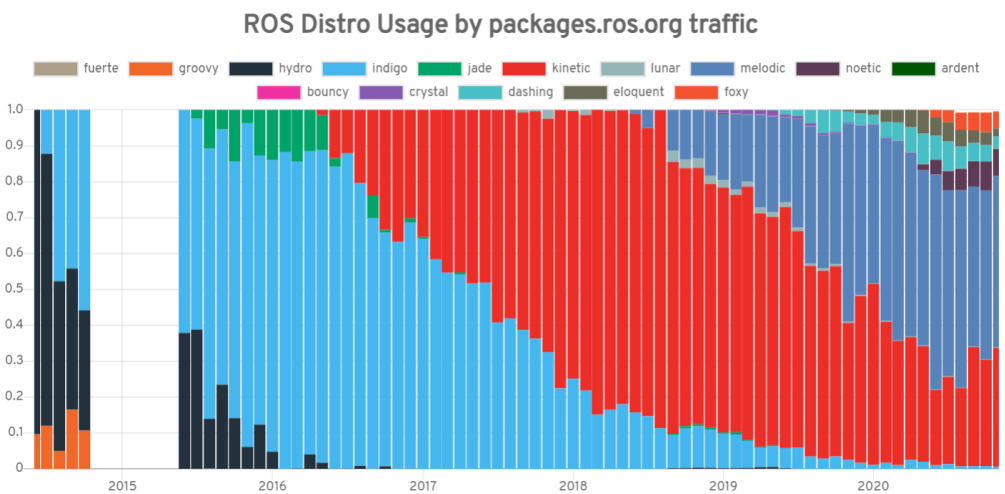
ros.org

ROS Distributions.

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial
ROS Noetic Ninjemys **(Recommended)**	May 23rd, 2020		
ROS Melodic Morenia	May 23rd, 2018		
ROS Lunar Loggerhead	May 23rd, 2017		
ROS Kinetic Kame	May 23rd, 2016		
ROS Jade Turtle	May 23rd, 2015		
ROS Indigo Igloo	July 22nd, 2014		
ROS Hydro Medusa	September 4th, 2013		

ROS Groovy Galapagos	December 31, 2012		
ROS Fuerte Turtle	April 23, 2012		
ROS Electric Emys	August 30, 2011		
ROS Diamondback	March 2, 2011		
ROS C Turtle	August 2, 2010		
ROS Box Turtle	March 2, 2010		

ROS Distributions over Time.



ROS architecture and philosophy.

ROS Philosophy.

- **Peer to peer:**
  Individual programs communicate over defined API (ROS messages, services, etc.).

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.

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.).

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.

ROS Philosophy.

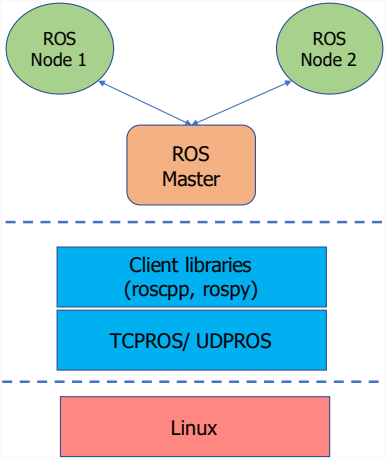
- **Peer to peer:**
  Individual programs communicate over defined API (ROS messages, services, etc.).
- **Distributed:**
  Programs can be run on multiple computers and communicate over the network.
- **Multi-lingual:**
  ROS modules can be written in any language for which a client library exists (C++, 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.

ROS AS A META-OPERATING SYSTEM

ROS is a supporting system for controlling robots and sensors with a **hardware abstraction** and developing robot applications based on existing conventional operating systems.

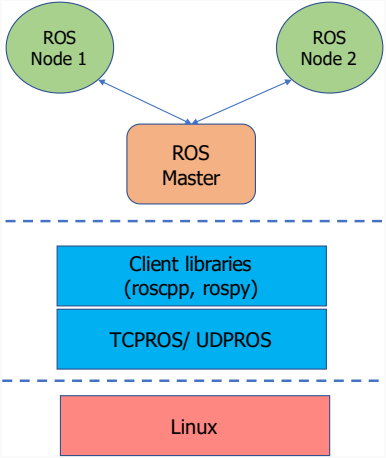
RSO ARCHITECTURE:

- **ROS Node:**
  It is the basic unit of processing.



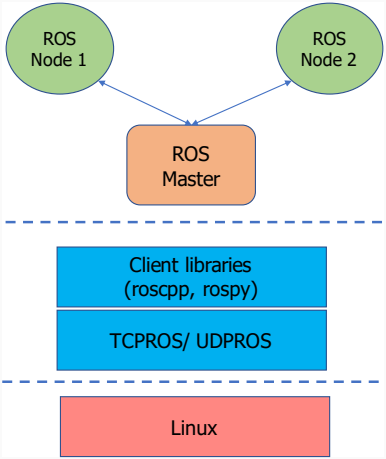
RSO ARCHITECTURE:

- **ROS Node:**
  It is the basic unit of processing.
- **ROS Master:**
  The core of ROS program. Nodes need to register theselves and communicate through it. (**Problematic!**)



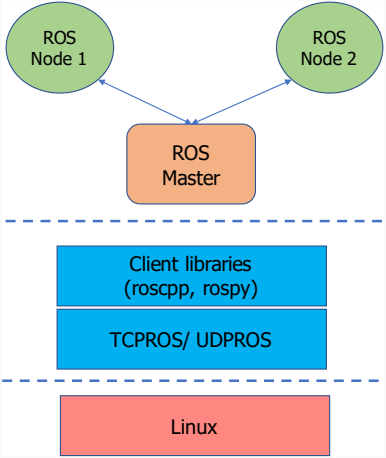
RSO ARCHITECTURE:

- **ROS Node:**
  It is the basic unit of processing.
- **ROS Master:**
  The core of ROS program. Nodes need to register theselves and communicate through it. (**Problematic!**)
- **Client libraries:**
  C++ and Python programming language are supported.



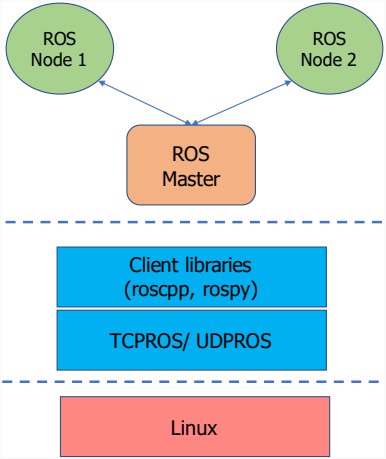
RSO ARCHITECTURE:

- **ROS Node:**
  It is the basic unit of processing.
- **ROS Master:**
  The core of ROS program. Nodes need to register theselves and communicate through it. (**Problematic!**)
- **Client libraries:**
  C++ and Python programming language are supported.
- **TCP/ UDP:**
  For peer-to-peer communications.



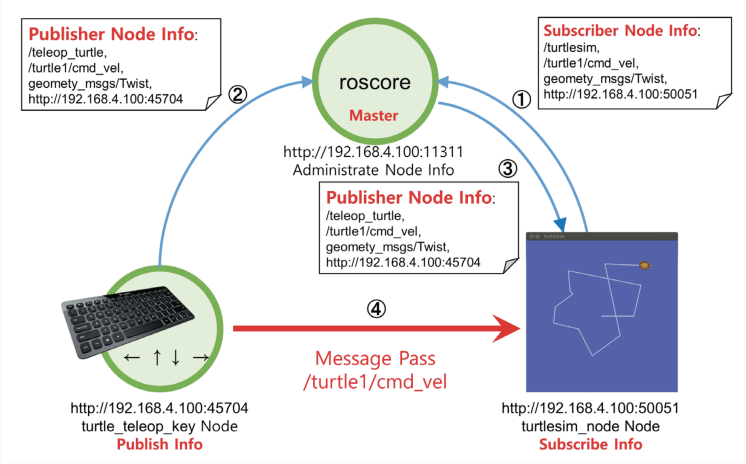
RSO ARCHITECTURE:

- **ROS Node:**
  It is the basic unit of processing.
- **ROS Master:**
  The core of ROS program. Nodes need to register theselves and communicate through it. (**Problematic!**)
- **Client libraries:**
  C++ and Python programming language are supported.
- **TCP/ UDP:**
  For peer-to-peer communications.
- **Linux OS:**
  ROS 1 is supported by Linux OS.



ROS master, nodes, topics, and messages.

ROS Basic Concept:



ROS DEMO: TURTLESIM

1. Run ROS master:

```
$ roscore
```

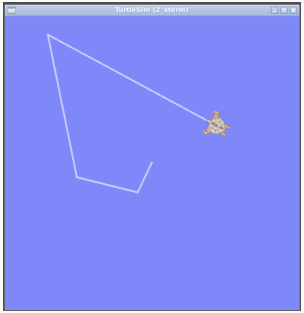
2. Run **turtlesim** node in turtlesim package:

rosrun [package_name] [node_name]

```
$ rosrun turtlesim turtlesim_node
```

3. Run **turtle_teleop_key** node:

```
$ rosrun turtlesim turtle_teleop_key
```




```
user:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```

ROS DEMO: TURTLESIM

1. List all the running nodes

```
$ rosnode list
```

/rosout
/teleop_turtle
/turtlesim

2. Get info about a certain node:

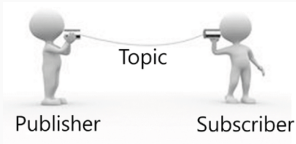
```
$ rosnode info turtlesim_node
```

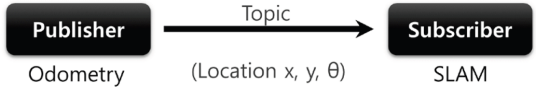
```
Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
 * /turtle1/teleport_relative
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
```

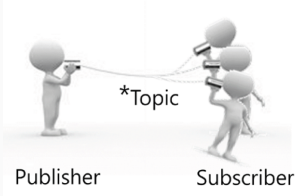
3. Visualize nodes communication:

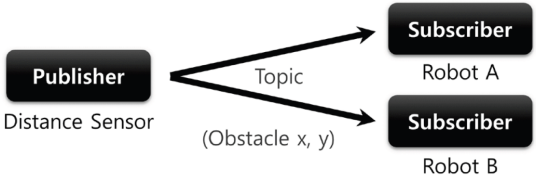
```
$ rqt_graph
```

PUBLISHER/SUBSCRIBER:









ROS topics are unidirectional and allow communication of 1:N, N:1 or N:N.

ROS DEMO: TURTLESIM

1. List all running ROS topics:

```
$ rostopic list
```

/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor

/turtle1/pose

2. Get information about a certain topic:

```
$ rostopic info /turtle1/cmd_vel
```

```
Type: geometry_msgs/Twist

Publishers:
 * /teleop_turtle (http://2_xterm:36487/)

Subscribers:
 * /turtlesim (http://2_xterm:38129/)
```

3. Get information about a certain message:

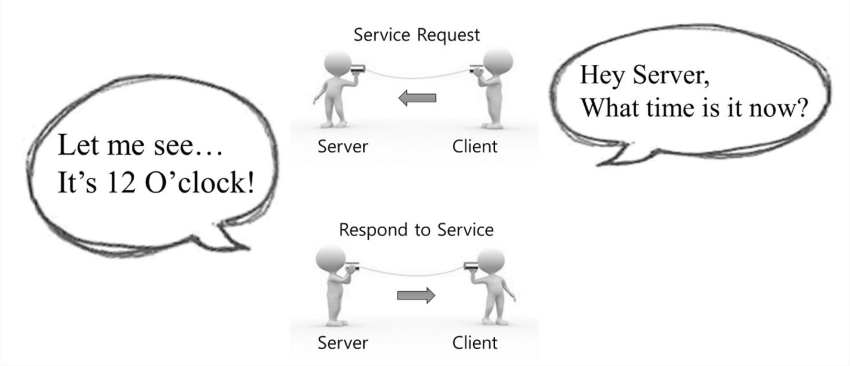
```
$ rosmsg show geometry_msgs/Twist
```

```
user:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

4. Display data sent over a topic:

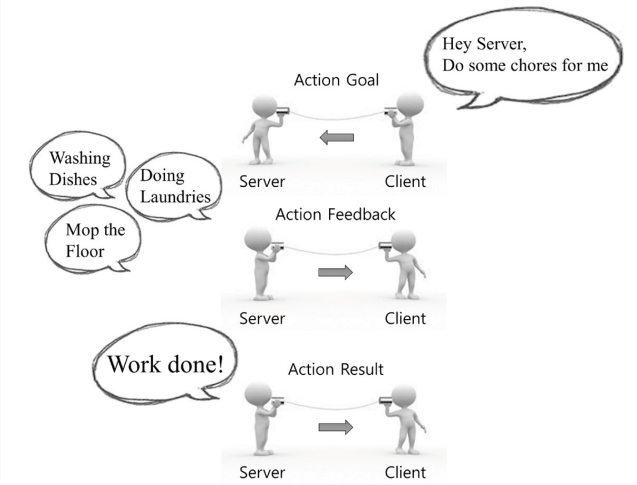
```
$ rostopic echo /turtle1/cmd_vel
```

ROS SERVICES:



- ROS services allow for bidirectional communications between a server and client.
- **Blocked client:** The client is waiting for the response from server.

ROS ACTIONLIB:



- Non-blocking bidirectional communication.
- Used if a server will take time.

ROS LIMITATIONS:

- **Built for a single robot:**
  No support for warm robots or multi-robots.

ROS LIMITATIONS:

- **Built for a single robot:**
  No support for warm robots or multi-robots.
- **Not real-time:**
  Communication throgh the ROS Master could slow processes.

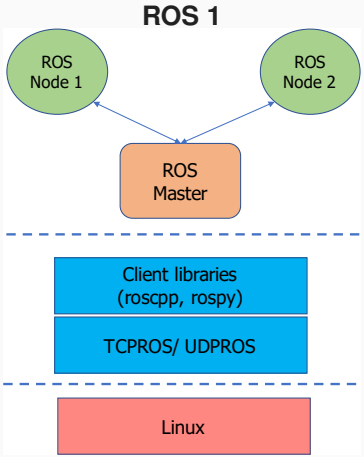
ROS LIMITATIONS:

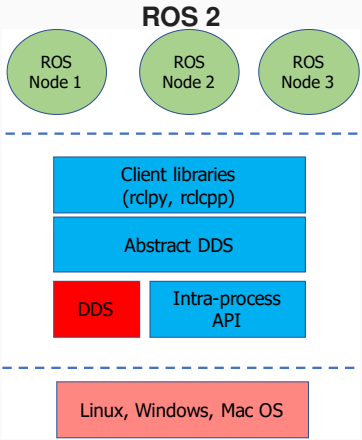
- **Built for a single robot:**
  No support for warm robots or multi-robots.
- **Not real-time:**
  Communication throgh the ROS Master could slow processes.
- **Needs reliable network:**
  Due to communications over TCP/UDP.

ROS LIMITATIONS:

- **Built for a single robot:**
  No support for warm robots or multi-robots.
- **Not real-time:**
  Communication throgh the ROS Master could slow processes.
- **Needs reliable network:**
  Due to communications over TCP/UDP.
- **Single point of filiure:**
  If ROS master crashes, the whole system will fail.

ROS 2 vs. ROS 1:

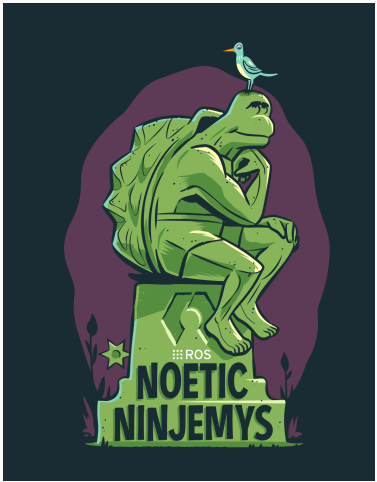

ROS 1
ROS Node 1
ROS Node 2
ROS Master
Client libraries (roscpp, rospy)
TCPROS/ UDPROS
Linux


ROS 2
ROS Node 1
ROS Node 2
ROS Node 3
Client libraries (rclpy, rclcpp)
Abstract DDS
DDS
Intra-process API
Linux, Windows, Mac OS

ROS 1 Installation.

We will use ROS Noetic on Ubuntu 20.04

Follow this link for installation of Ubuntu, ROS, and VS Code for code edit.



ROS Noetic

End of Lecture

haitham.elhussieny@ejust.edu.eg