ROS: Robot Operating System

Robotic Agents © Allegheny College

Janyl Jumadinova

November 5, 2019
Robot Operating System (ROS)

ROS provides:

- Communication Infrastructure
- Robot-specific features
- Less programming language barriers
- Diagnostic tools
- Simulation capabilities
- Software libraries/tools
Robot Operating System (ROS)

**ROS provides:**
- Communication Infrastructure
- Robot-specific features
- Less programming language barriers
- Diagnostic tools
- Simulation capabilities
- Software libraries/tools

ROS: https://www.ros.org/
Two Sides of ROS

1. OS side provides standard operating system services:
   - hardware abstraction
   - low-level device control
   - implementation of commonly used functionality
   - message-passing between processes
   - package management
Two Sides of ROS

1. OS side provides standard operating system services:
   - hardware abstraction
   - low-level device control
   - implementation of commonly used functionality
   - message-passing between processes
   - package management

2. A suite of user contributed packages that implement common robot functionality such as SLAM (localization), planning, perception, vision, manipulation, etc.
ROS Core Concepts

- Nodes
- Messages and Topics
- Services
- ROS Master
- Parameters
- Stacks and packages
Nodes

http://wiki.ros.org/Nodes

- A **node** is a process that performs some computation.
- Typically we try to divide the entire software functionality into different modules - each one is run over a single or multiple nodes.
- Nodes are combined together into a graph and communicate with one another using streaming topics, RPC services, and the Parameter Server.
http://wiki.ros.org/Topics

- **Topics** are named buses over which nodes exchange messages.
- Topics have anonymous publish/subscribe semantics - a node does not care which node published the data it receives or which one subscribes to the data it publishes.
- There can be multiple publishers and subscribers to a topic.
Messages

http://wiki.ros.org/Messages

- Nodes communicate with each other by publishing messages to topics.
- A message is a simple data structure, comprising typed fields (http://wiki.ros.org/std_msgs).
- Messages may also contain a special field called header which gives a timestamp and frame of reference.
Services

http://wiki.ros.org/Services
- Synchronous inter-node transactions / RPC.

Caller (client process)
- Call procedure
- waiting for reply

Callee (Server process)
- waiting for request
- Receive request and start procedure execution
- Procedure executes
- Send reply
- waiting for next request

Request message (contains remote procedure's parameter)

Resume execution

Reply message (contains result of procedure execution)

Remote procedure call model
http://wiki.ros.org/Services

- Synchronous inter-node transactions / RPC.
- **Service/Client model**: 1-to-1 request-response.
- Service roles:
  - carry out remote computation
  - trigger functionality / behavior
http://wiki.ros.org/Services

- Synchronous inter-node transactions / RPC.
- **Service/Client model**: 1-to-1 request-response.
- Service roles:
  - carry out remote computation
  - trigger functionality / behavior
- Example:
  - `map_server/static_map` - retrieves the current grid map used by the robot for navigation
http://wiki.ros.org/Master

Provides connection information to nodes so that they can transmit messages to each other

- Every node connects to a master at startup to register details of the message streams they publish, and the streams to which they subscribe.
- When a new node appears, the master provides it with the information that it needs to form a direct peer-to-peer connection with other nodes publishing and subscribing to the same message topics.
Parameter Server

http://wiki.ros.org/ParameterServer

- A shared, multi-variate dictionary that is accessible via network APIs.
- Best used for static, non-binary data such as configuration parameters.
- Runs inside the ROS master.
ROS Package System
ROS Distribution Releases

ROS Version / Poster / Icon

http://wiki.ros.org/Distributions
ROS Communication

- cmvision node: I will receive images on topic "image" and publish blobs on topic "blobs"
- control node: I will receive blobs on topic "blobs" and publish velocities on topic "cmd_vel"
- ROS Master
- camera node: I will publish images on topic "image"
- create node: I will receive velocities on topic "cmd_vel"

Connections:
- cmvision node to camera node via USB
- control node to ROS Master
- create node to ROS Master via Serial
ROS Communication

- cmvision node
  - blobs on "blcbs"
- control node
- ROS Master
  - velocities on "cmd_vel"
- camera node
  - images on "image"
- create node
  - SETS UP COMMUNICATION

USB

USB-Serial
ROS Communication

Computation Graph:
the peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph concepts of ROS are nodes, Master, Parameter Server, messages, services, topics, and bags, all of which provide data to the Graph in different ways.

http://wiki.ros.org/ros_comm
http://wiki.ros.org/ROS/Tutorials
Turtlebot 2

- Max speed: 65 cm/s
- High resolution wheel encoders
- Cliff sensors
- Bump sensors
- etc.
Kobuki Mobile Base

Specifications:

- 3 x Bumpers: left, centre, right
- 1 x Cliff sensors
- 1 x Wheel drop sensor (one per wheel)
- Programmable audio, LEDs and touch buttons
- Battery: Lithium Ion 2200 mAh
- Payload: 5 kg (hard floor), 4 kg (carpet)
Yujin Robot’s Kobuki

https://www.youtube.com/watch?v=t-KTHkbUwrU
Kinect Sensor

- Depth image
- RGB image
- Tilt

Kinect depth sensor range is: minimum 800mm and maximum 4000mm.

Kinect uses Infrared so it can see through glass (not reliable for obstacle avoidance if have glass doors).

Kinect uses IR, so it will not work in direct sunlight (outdoors).

https://www.youtube.com/watch?v=uq9SEJxZiUg
Kinect Sensor

- Depth image
- RGB image
- Tilt
- Kinect depth sensor range is: minimum 800mm and maximum 4000mm
Kinect Sensor

- Depth image
- RGB image
- Tilt
- Kinect *depth sensor range* is: minimum 800mm and maximum 4000mm
- Kinect uses Infrared so it can see through glass (not reliable for obstacle avoidance if have glass doors)
- Kinect uses IR, so it will not work in direct sunlight (outdoors)
Kinect Sensor

- Depth image
- RGB image
- Tilt
- Kinect *depth sensor range* is: minimum 800mm and maximum 4000mm
- Kinect uses Infrared so it can see through glass (not reliable for obstacle avoidance if have glass doors)
- Kinect uses IR, so it will not work in direct sunlight (outdoors)

https://www.youtube.com/watch?v=uq9SEJxZiUg
Turtlebot ROS Resources

http://wiki.ros.org/Robots/TurtleBot

http://edu.gaitech.hk/turtlebot/turtlebot-tutorials.html
ROS driver for BEBOP

bebop_autonomy

http://autonomylab.org/bebop_autonomy/
ROS driver for BEBOP

bebop_autonomy

http://autonomylab.org/bebop_autonomy/

Example: https://www.youtube.com/watch?v=8oMZeMhczl8