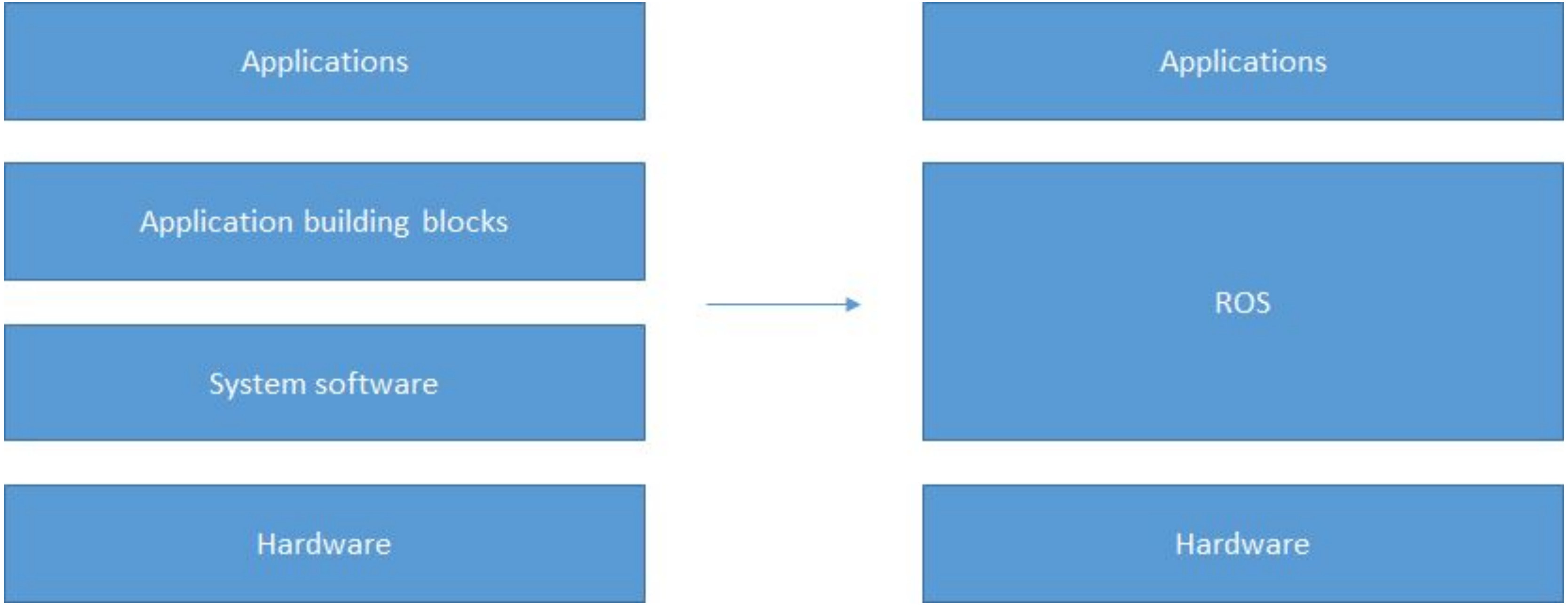
ROS

Intro to
Robot Operating System

PC ecosystem vs. Robotics ecosystem



Applications

Application building blocks

System software

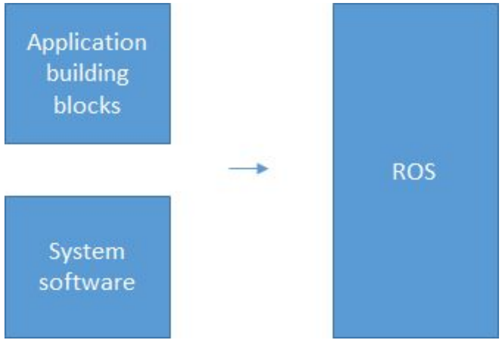
Hardware

Applications

ROS

Hardware

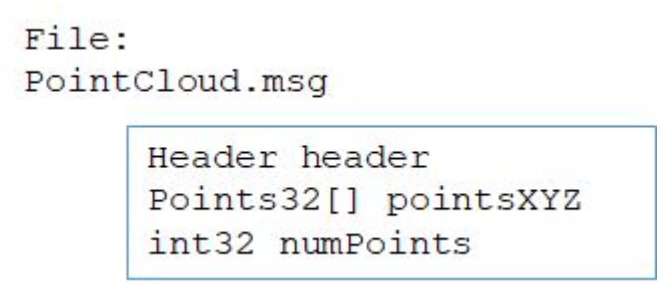
What is ROS?



- Open source, Free
- Runs in Linux/Ubuntu
- Agent (node) based
- Peer to Peer Message passing
  - Publish/Subscribe Topics
  - Services
- Support multiple languages (C++, Python, Java, Lisp...)
- Low level device abstraction - GPS, Camera, LIDAR ... (Hardware agnosticism)
- Application building blocks such as SLAM, Object recognition, Arm path planning, ...
- Software management (compiling, packaging)
- Remote communication and control

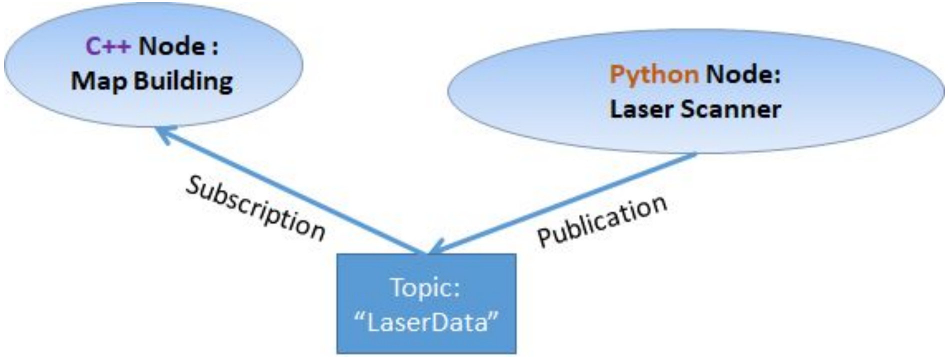
Multiple Language Support

- ROS is implemented natively in each target language.
- Language-independent Interface Definition Language (IDL) for defining messages

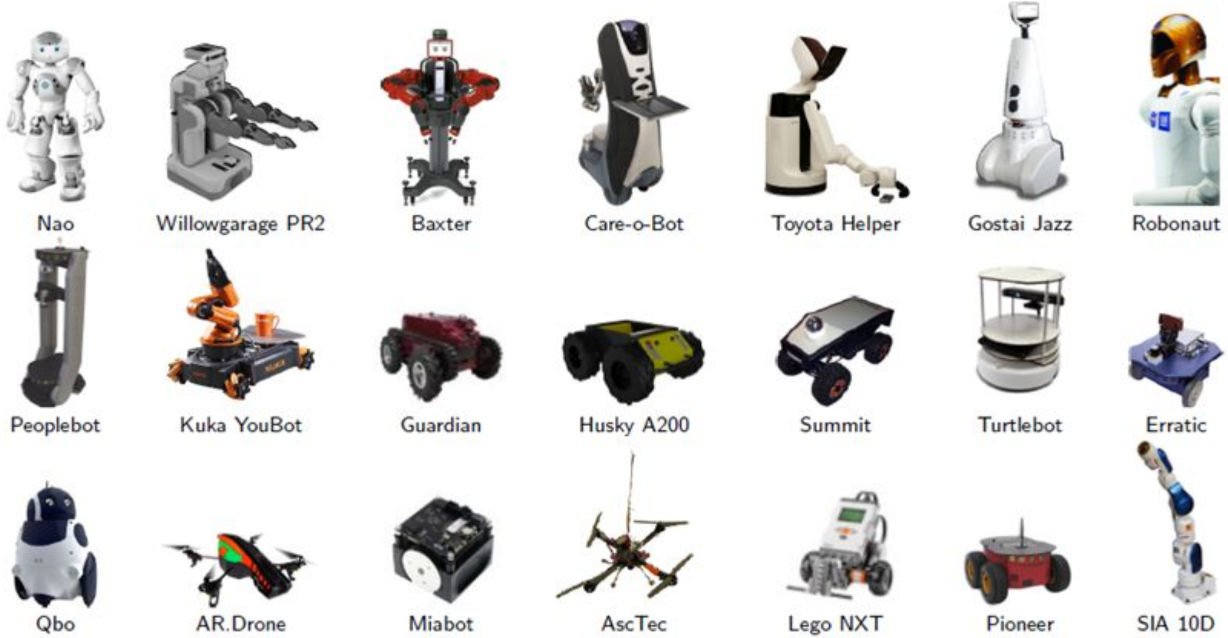


File:
PointCloud.msg

```
Header header
Points32[] pointsXYZ
int32 numPoints
```



Supported Robots



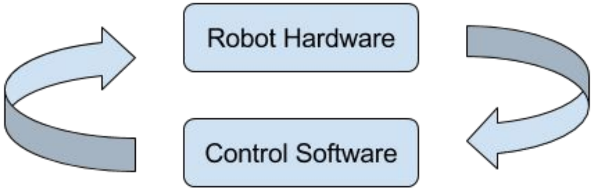
A lot more on http://www.ros.org/wiki/Robots

Sensors

- 1D/2D/3D rangefinders
  - Sharp IR range finder
  - Hokuyo, SICK laser scanners
  - Microsoft Kinect
  - Velodyne
- Cameras
  - monocular and stereo
  - USB and Ethernet
- Force/torque/touch sensors
- Motion capture systems
- Pose estimation (IMU/GPS)
- Audio/Speech recognition
- RFID
- Sensor/actuator interfaces: Arduino, ...

Conventional Robotics Software

1. Input from robot
2. Make decision based on goal
3. Send command to robot
4. Repeat



Robot Hardware

Control Software

```
loop()
{
   ...
   Image image = camera.getImage();
   Vec2D vec =
getLaneDirection(image);
   if (!lidar.obstacleExists())
      moveToward(vec);
   ...
}
```

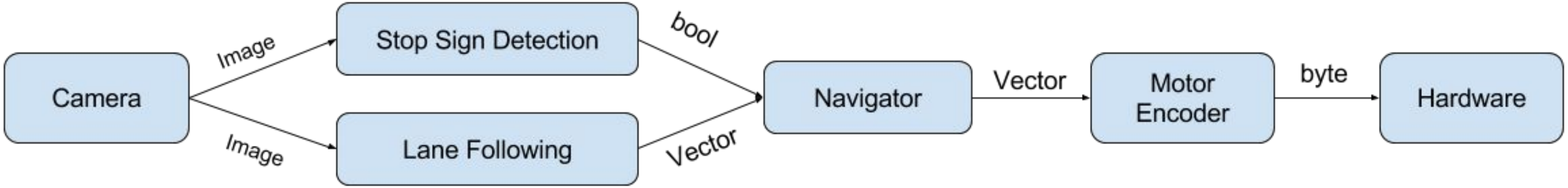
Problems

- The world is asynchronous
- Decisions are complex

Robot Operating System ROS

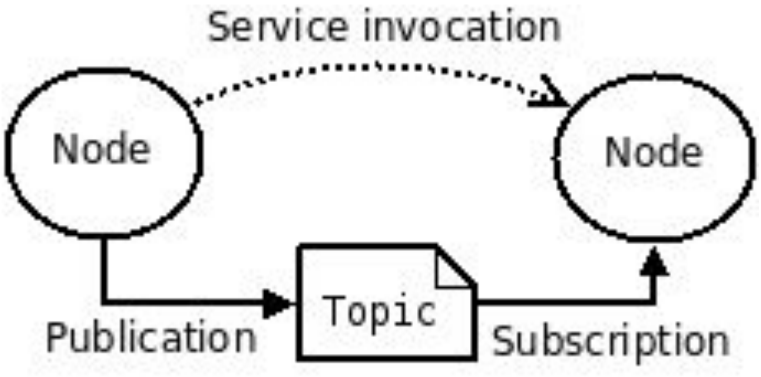
The Solution

- Event (Message) based
  - Publisher / Subscriber
- Modular
  - Nodes are concurrent processes
  - Hardware abstraction



ROS Node & Message Communication

- A Process
- Combined using message passing via publishers and subscribers

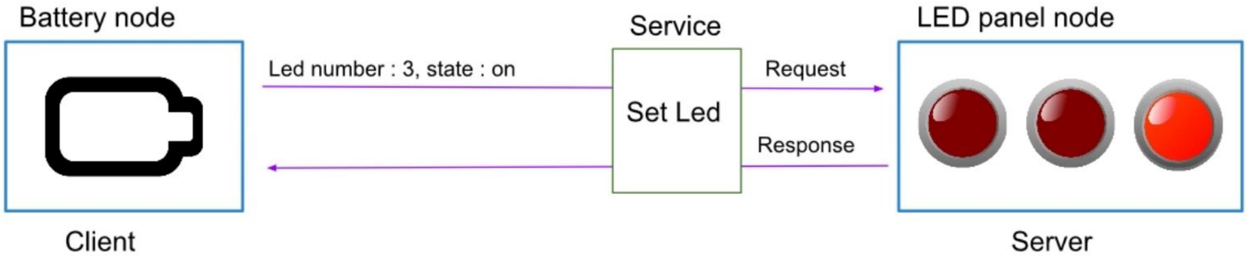


Publish/Subscribe

- Asynchronous
- A subscriber may however be set to synchronous mode and the publisher will be blocked until the subscriber catches up.

ROS Services

- A ROS service is a client/server system
- Synchronous - await for response
- For requiring quick actions
- One message type for Request, another message type for Response
- Can have multiple clients for a service server



Why ROS? (1)

- The robots (controllers including RPI & LEGO EV3, sensors, and actuators) around the world love Linux/ROS
- Doesn't matter if everything is different. language-agnostic
- Something may fail, but not everything
- Concurrent resource handling
- ROS is light
- Everything is modular
- Great simulation tools such as Gazebo, Rviz
- Open source project maintained by many people; The ROS Community is growing
- Sell ROS robots without any licence fee

https://www.intorobotics.com/15-reasons-to-use-the-robot-operating-system-ros/
https://niryo.com/2018/01/8-reasons-use-ros-robotics-projects/

Why ROS? (2)

- Good architecture for CPS and IoT
- Learning a different software architecture
- Max. P & Q
- LTU ACTor team is looking for new members for 2019-20 year