
KINETIC KAME
ROS

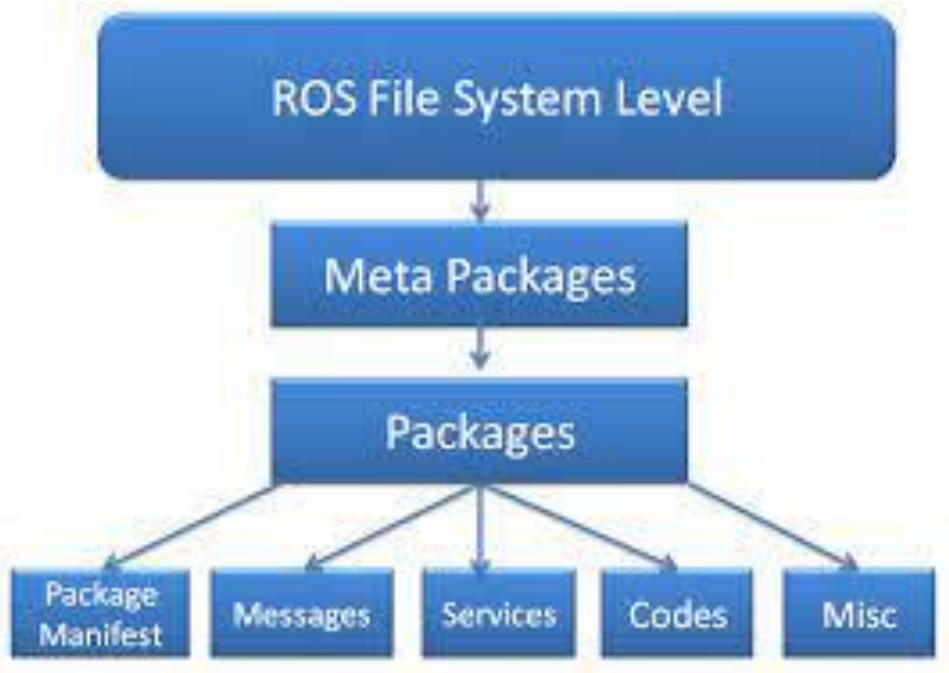
ROS Kinetic Kame Released

By Tully Foote on May 23, 2016 3:43 PM

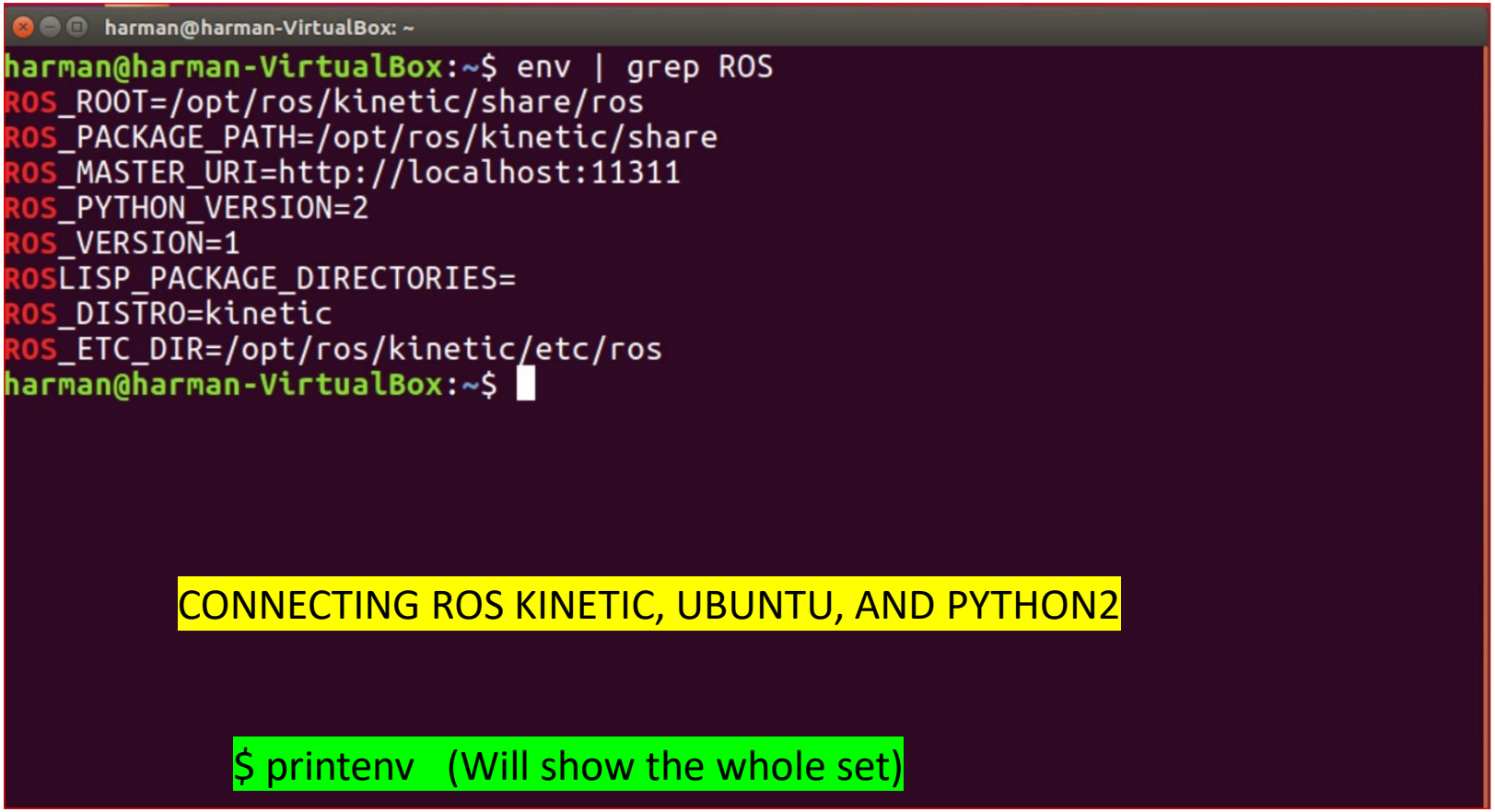
Happy World Turtle Day!

I am pleased to announce that the 10th ROS distribution, Kinetic Kame,
is now available on Ubuntu Xenial 16.04, Ubuntu Wily 15.10, and Debian Jessie.
Packages for 32-bit ARM (armhf) are available on Xenial, and 64-bit ARM (aarch64)
is supported on Debian Jessie.

ROS Kinetic **ONLY** supports Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) and Jessie (Debian 8) for debian packages.



https://subscription.packtpub.com/book/hardware_and_creative/9781788478953/1/ch01lvl1sec13/understanding-the-ros-filesystem-level


harman@harman-VirtualBox: ~
harman@harman-VirtualBox:~$ env | grep ROS
ROS_ROOT=/opt/ros/kinetic/share/ros
ROS_PACKAGE_PATH=/opt/ros/kinetic/share
ROS_MASTER_URI=http://localhost:11311
ROS_PYTHON_VERSION=2
ROS_VERSION=1
ROSLISP_PACKAGE_DIRECTORIES=
ROS_DISTRO=kinetic
ROS_ETC_DIR=/opt/ros/kinetic/etc/ros
harman@harman-VirtualBox:~$
CONNECTING ROS KINETIC, UBUNTU, AND PYTHON2
$ printenv (Will show the whole set)

ROS_PACKAGE_PATH: This tells ROS where to find packages, so that rosrun <pkg> <node> actually works.

ROS_PACKAGE_PATH is an optional, but very common environment variable that allows you to add more ROS packages to your ROS path.

ROS_MASTER_URI is a **required setting** that tells nodes where they can locate the master.

PYTHONPATH: This tells python where to look for any python modules defined in your package.

http://library.isr.ist.utl.pt/docs/roswiki/ROS(2f)EnvironmentVariables.html

ROS Packages

- ROS software is divided into packages that can contain programs, images, data and even tutorials.
- A package provides a useful function for ease of use and especially reuse.
- ROS package contains a package.xml file (sometimes called a manifest).
- ROS packages are built with catkin build process.
- Multiple related packages are combined into a metapackage.

Basic ROS building blocks

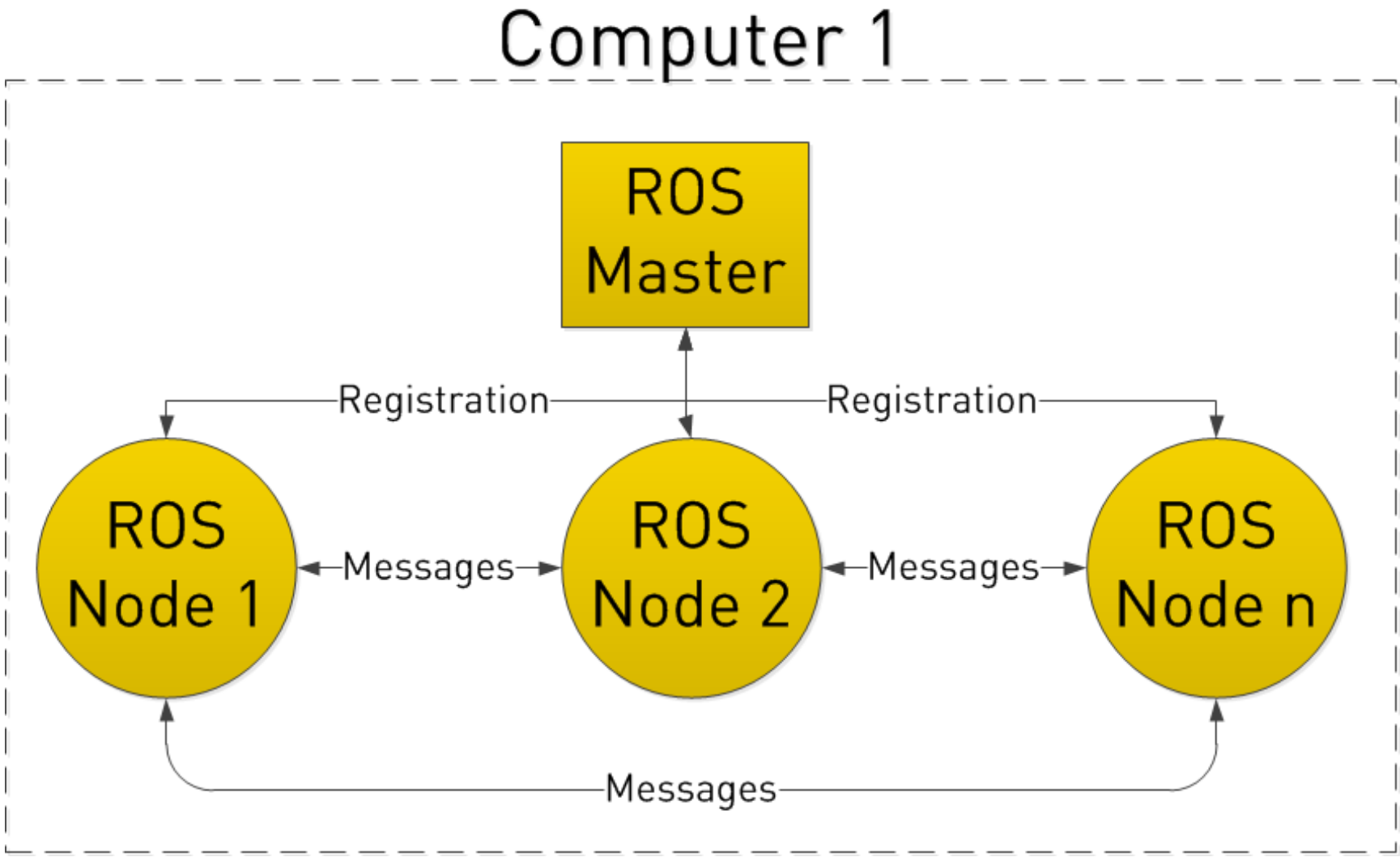
- Packages
- Nodes
- Topics
- Messages
- Services

ROS Nodes

- A ROS system is comprised of a number of independent nodes, each of which communicates with the other nodes using a publish/subscribe messaging model.
- ROS nodes are built small to control one aspect of a robotic system.
- For example, a USB camera driver is implemented as a node, which publishes image data in a stream of messages. These messages can be consumed by any number of other nodes.
- Nodes in ROS do not have to be on the same computer or even of the same architecture!

ROS Master

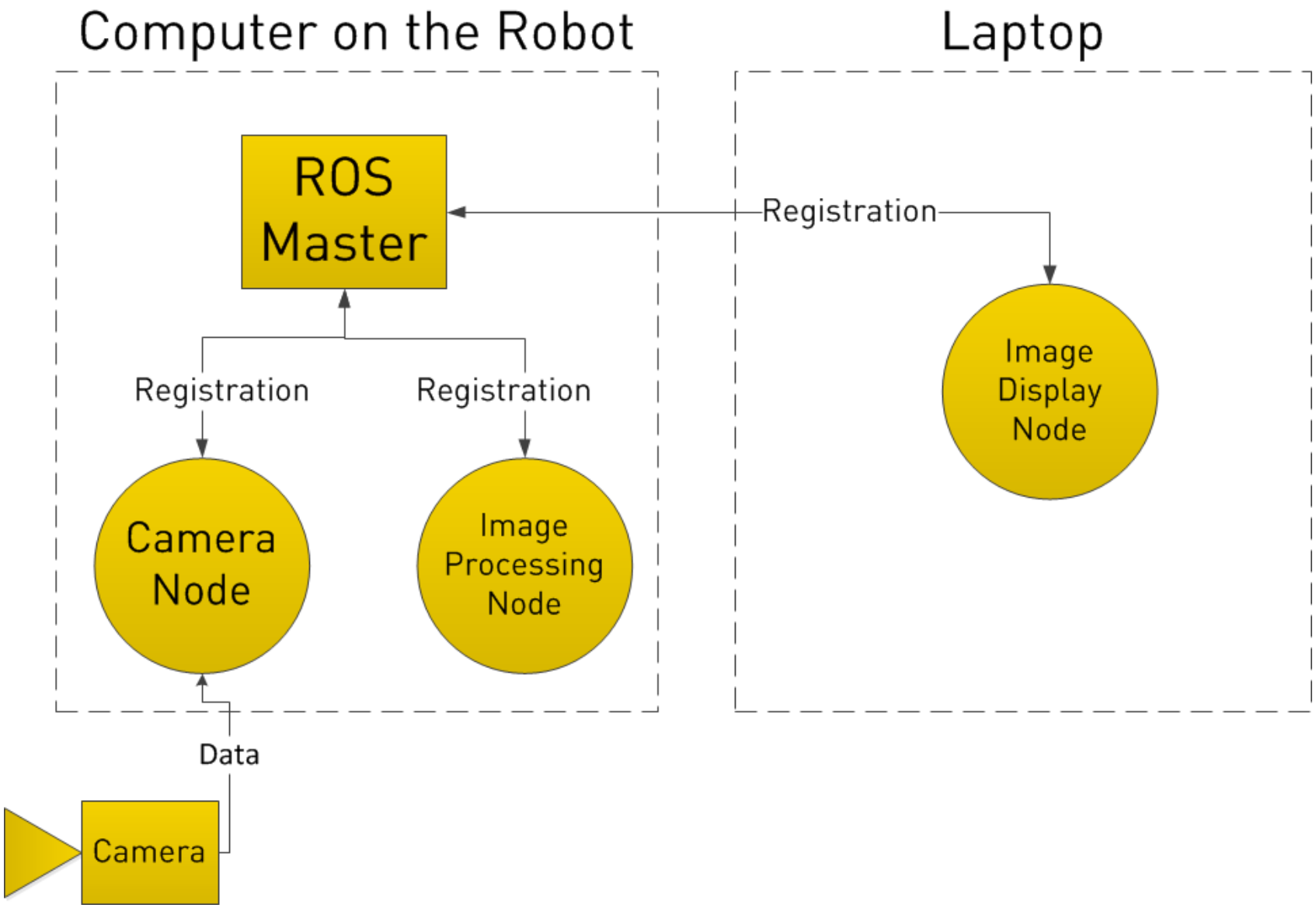
- ROS must start with a ROS Master.
- The Master allows all other ROS software processes (nodes) to find and talk to each other.
- All nodes must register with the Master when they startup.
- After registration, Node 1 can send and receive messages from Node2.



Images from https://www.clearpathrobotics.com/2014/01/how-to-guide-ros-101/

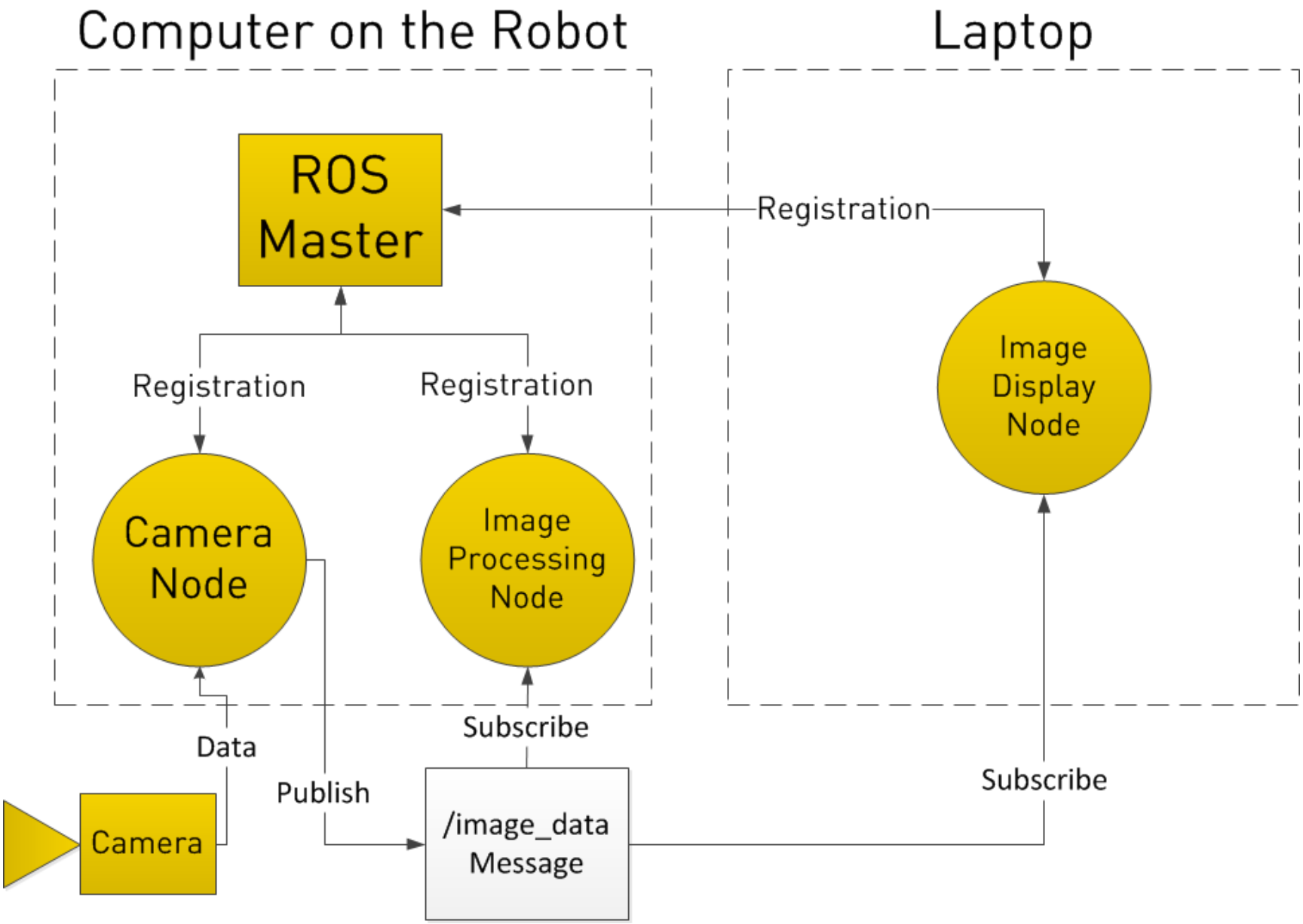
ROS Topics

- ROS topics are published and subscribed to by nodes.
- To see images from the camera, a Camera Node communicates with the camera and an Image Processing Node processes the image data. A third node, an Image Display Node displays images on the computer screen.


Computer on the Robot
Laptop
ROS Master
Registration
Image Display Node
Registration
Registration
Camera Node
Image Processing Node
Data
Camera

ROS Master (continued)

- As these nodes register with the Master:
  - The Camera Node declares that it publishes a topic called */image_data*.
  - The Image Processing Node and the Image Display Node declare that they subscribe to the topic */image_data*.
- Camera Node interfaces with the camera and creates and sends */image_data* messages directly to the other two nodes.

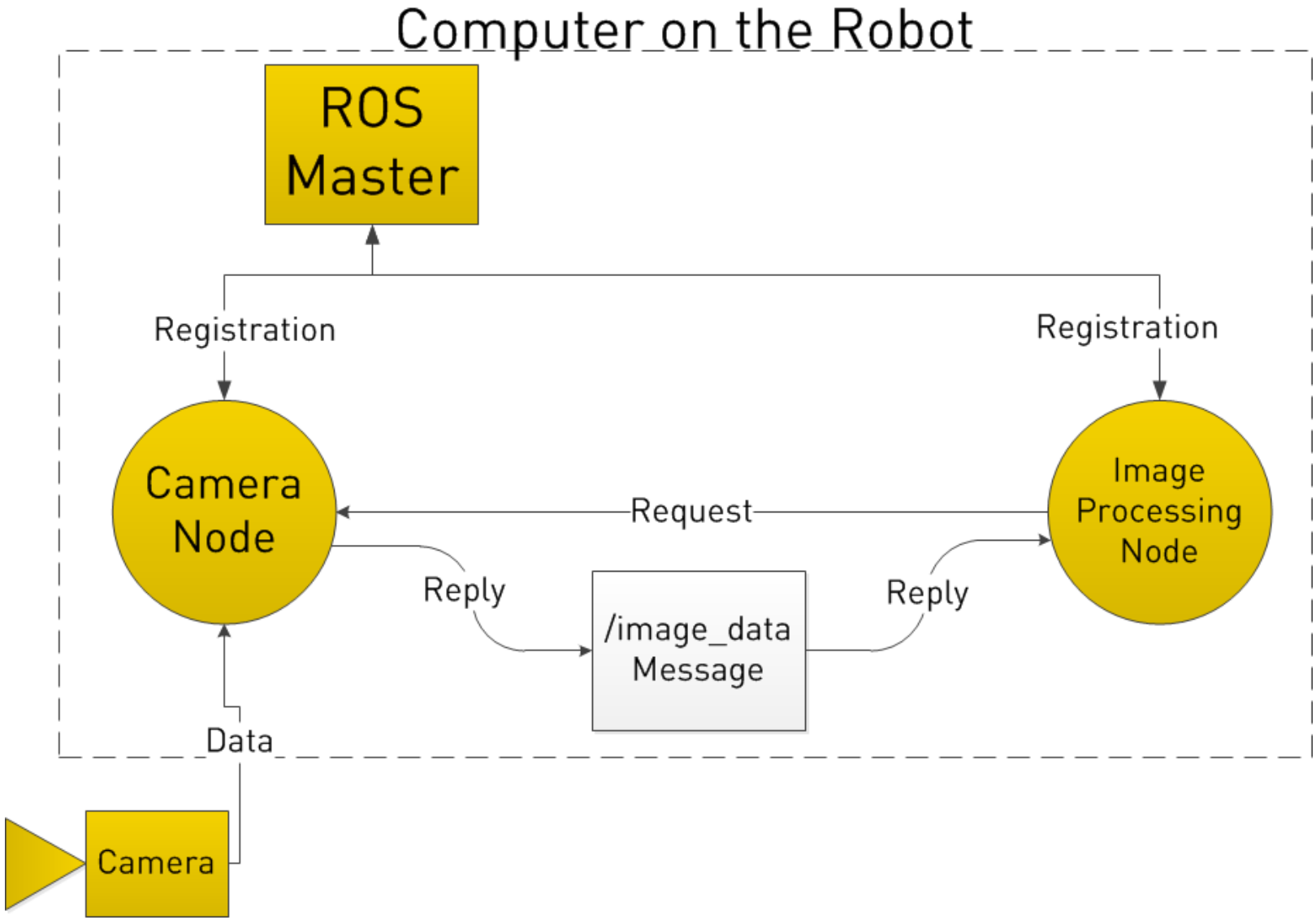

Computer on the Robot
Laptop
ROS Master
Registration
Image Display Node
Registration
Registration
Camera Node
Image Processing Node
Data
Subscribe
Publish
Subscribe
Camera
/image_data Message

ROS Messages

- ROS message is a strictly typed data structure.
- Analogy:
  - Topic is the suitcase
  - Messages are the contents

ROS Services

- Nodes register a specific service with the ROS Master (just like topics).
- ROS services work on a request and response cycle.
- For example, the Image Processing Node requests the service for */image_data*. The Camera Node acquires the data from the camera and sends the reply.


Computer on the Robot
ROS Master
Registration
Registration
Camera Node
Request
Image Processing Node
Reply
/image_data Message
Reply
Data
Camera

ROS Parameter Server

- ROS Parameter Server is a dictionary of parameters shared between the nodes.
- Accessed by the ROS Master (seamless)
- Exchange of parameters (data) between nodes at runtime

ROS help

- ROS wiki http://wiki.ros.org/
- ROS getting started http://wiki.ros.org/ROS/StartGuide
- ROS tutorials http://wiki.ros.org/ROS/Tutorials
- ROS answers http://answers.ros.org/questions/
- ROS Discorse https://discourse.ros.org/
- ROS blog http://www.ros.org/news/

ROS Blog Retired