Robot Operating System

Dong Li

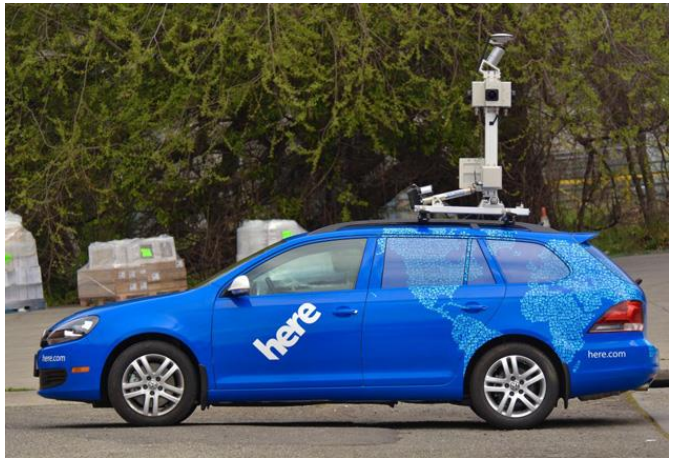
Topics

I. Introduction to ROS
II. ROS Software Development
III. Demo
IV. Todo

Introduction to ROS

- Created by Stanford AI Lab in 2007
- Open source and developed by many groups
- Has been applied to many commercial products and industry

Introduction to ROS



HERE map car

Hekateros robotic arms

PAL Robotics

http://www.ros.org/news/robots/

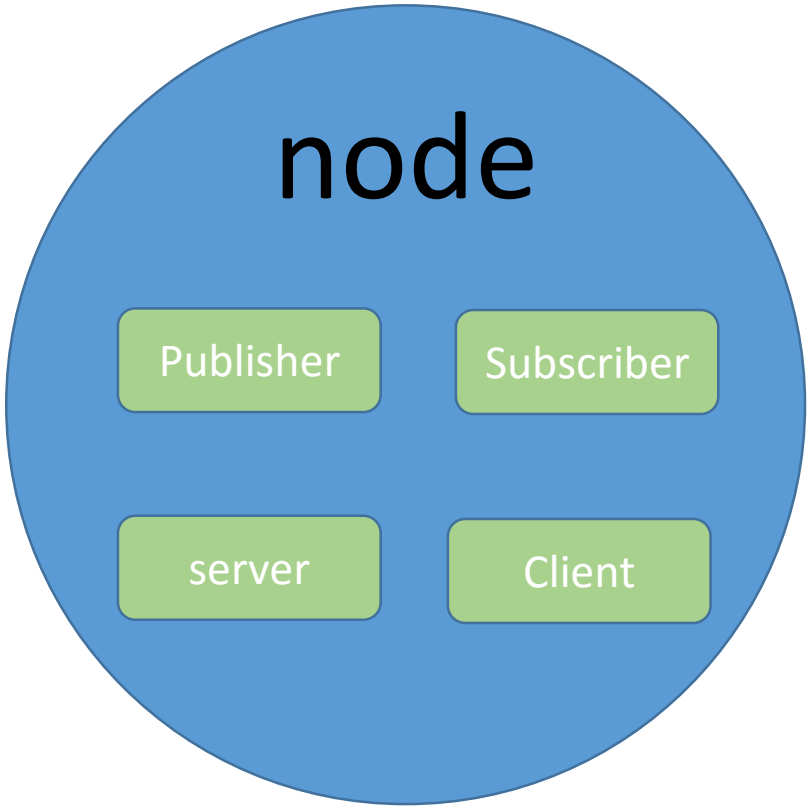
Introduction to ROS

Introduction to ROS

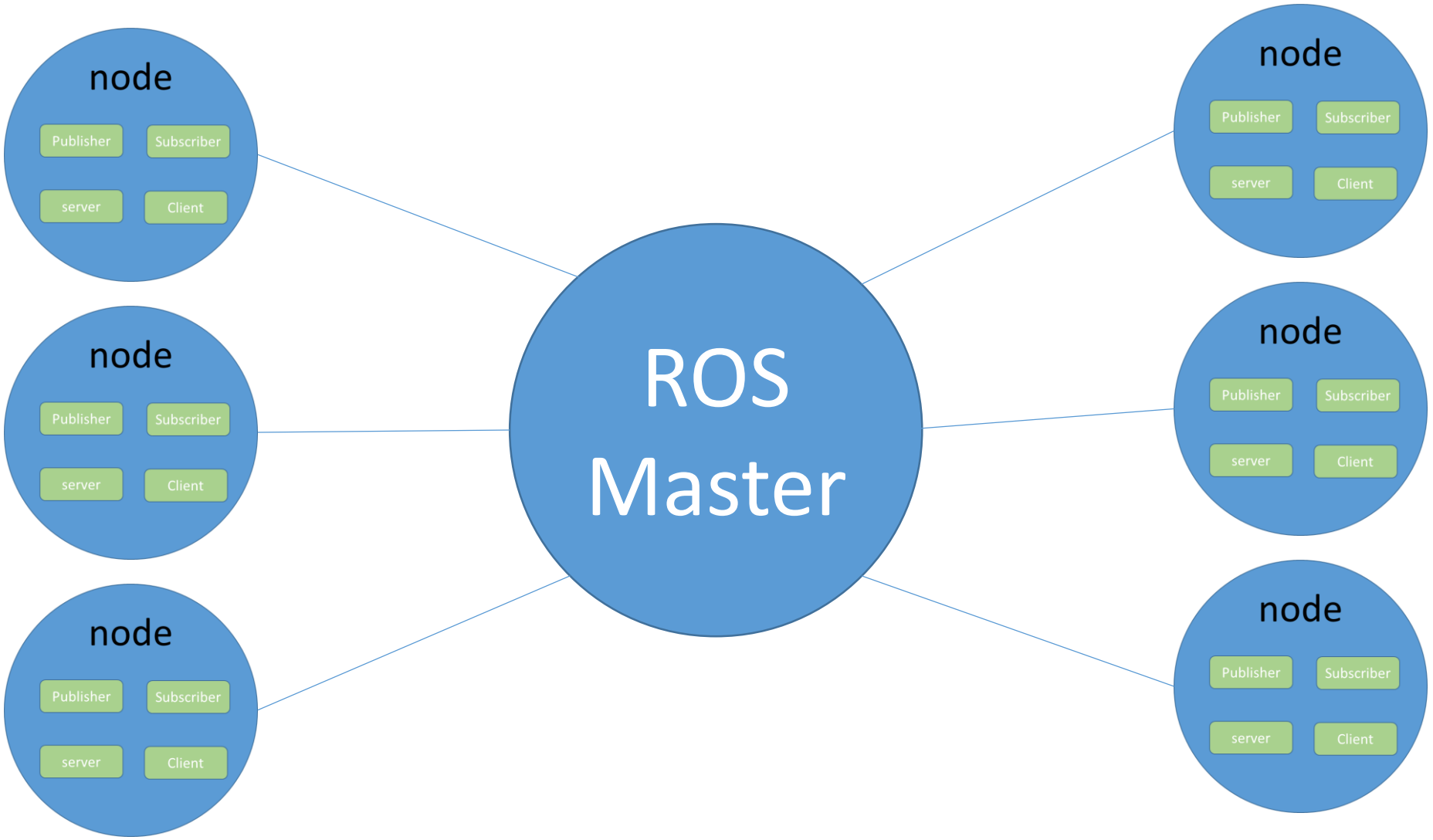
- a flexible framework for writing robot software
- a collection of tools, libraries

Introduction to ROS: framework

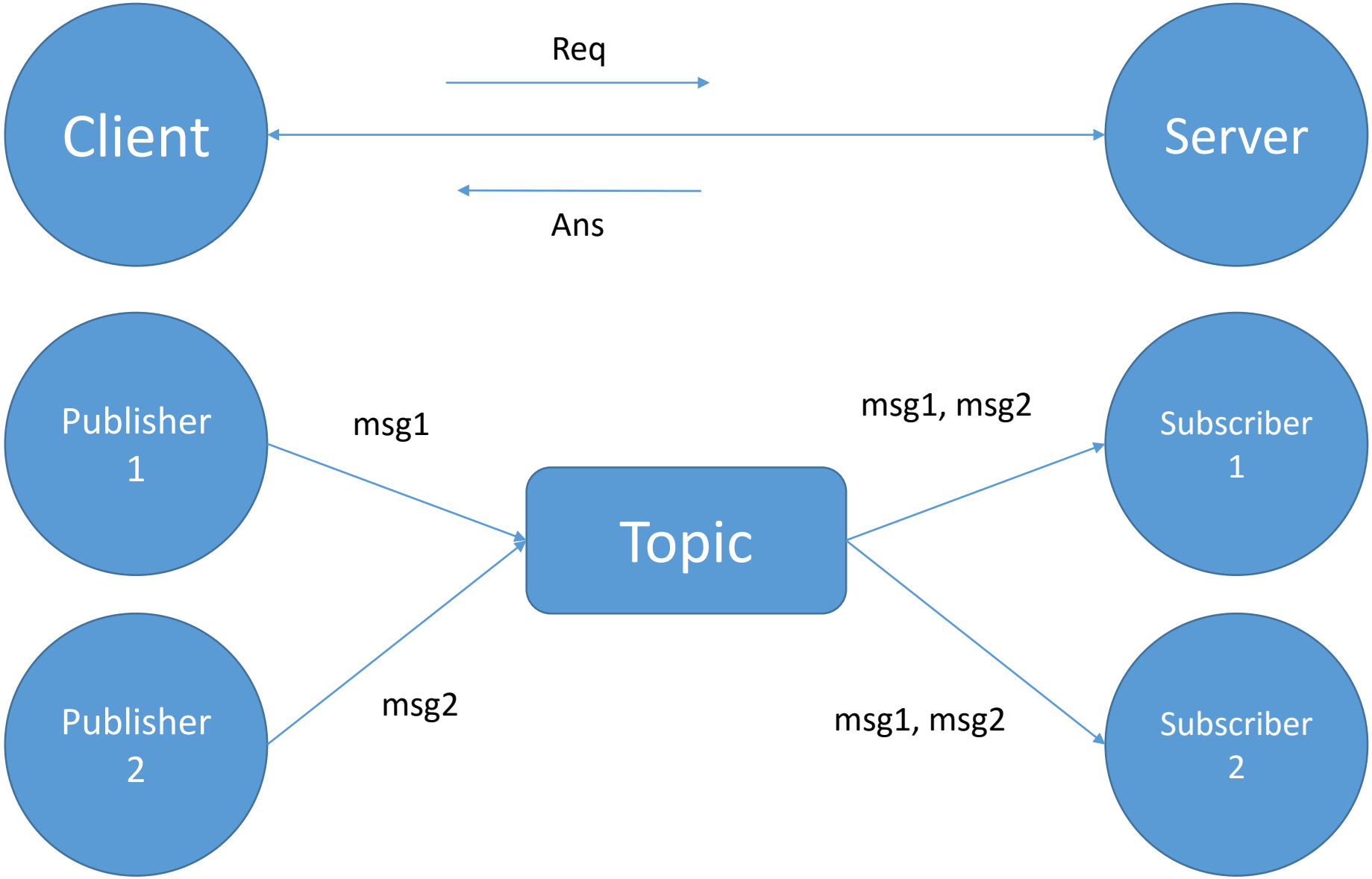
- Basic compoment: node
- Publisher:
  - publish msg with some topic
- Subscriber:
  - receive msg with some topic
- Server:
  - receive the req from client and answer to it
- Client:
  - Send req to server and receive the answer



Introduction to ROS: framework

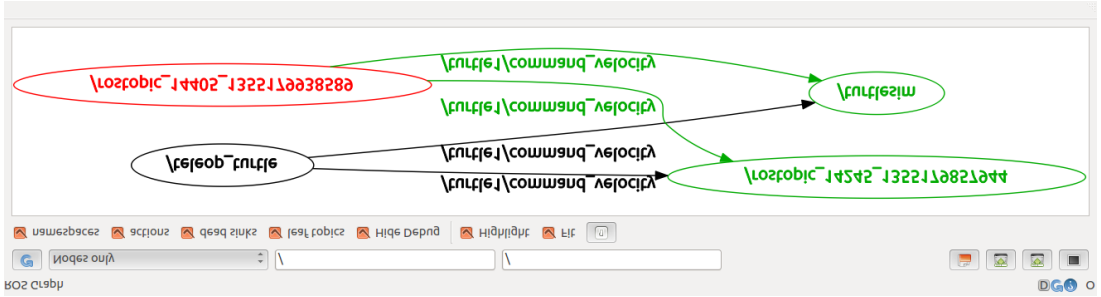


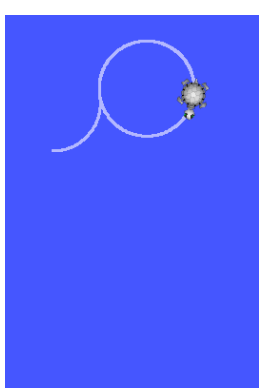
Introduction to ROS: framework

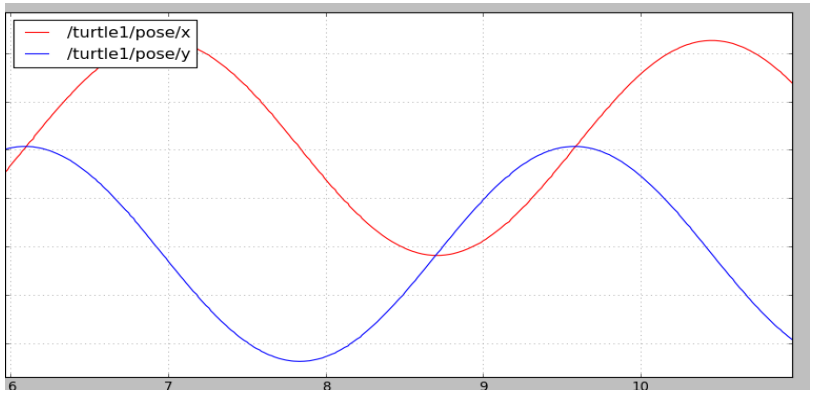


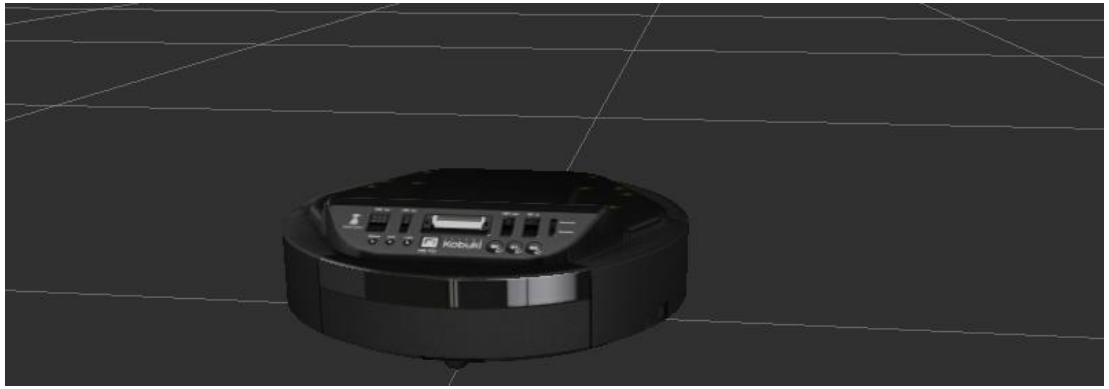
Introduction to ROS: tools

- Command line:
  - roscd, rosls, rosrun etc
  - rostopic pub etc
  - roswtf
- GUI tools: rqt
  - node monitor
  - data visualization
  - model simulator









ROS Software Development

- Platform: Linux, Mac, Windows
- Processor: x86, ARM
- Language:
  - C/C++, Python
  - Java (for Android)
  - Javascript (for web app, ROSjs)
  - Objective-C (for iOS, RBManager)

ROS Software Development

- One ROS app is called a package in ROS workspace
  - src (for C++/C)
  - scripts (for Python)
  - msg
  - srv
  - launch
  - include (dependencies)

ROS Software Development

- Msg: msgName.msg
  - Define the format of msg
  - Both publisher and subscriber nodes must have the file so that the msg can be decode correctly
  - E.g

```
string name
int32 age
```

  - Special msg: numpy.array, sensor_msg like video (predefined)

ROS Software Development

- Msg: msgName.msg
  - Publisher

```
pub = rospy.Publisher('custom_chatter', MPerson) #define topic name
rospy.init_node('custom_talker', anonymous=True) #define node name
r = rospy.Rate(100) #10hz
msg = MPerson()
msg.name = "ROS User"
msg.age = 4

while not rospy.is_shutdown():
    #raw_input("enter:")
    # if len(name) == 0:
    rospy.loginfo(msg)
    pub.publish(msg)
    r.sleep()
```

ROS Software Development

- Msg: msgName.msg
  - Subscriber

```
def callback(data):
    rospy.loginfo("%s is age: %d" % (data.name, data.age))

def listener():
    rospy.init_node('custom_listener', anonymous=True)
    rospy.Subscriber("custom_chatter", MPerson, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

ROS Software Development

- Srv: msgName.srv
  - Define the format of srv
  - Both server and client nodes must have the file so that the req and ans can be decode correctly
  - E.g

```
int64 a
int64 b
---
int64 sum
```

ROS Software Development

- Srv: msgName.srv
  - Service

```
def handle_add_two_ints(req):
    print "Returning [%s + %s = %s]"%(req.a, req.b, (req.a + req.b))
    return AddTwoIntsResponse(req.a + req.b)

def add_two_ints_server():
    rospy.init_node('add_two_ints_server')
    s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
    print "Ready to add two ints."
    rospy.spin()

if __name__ == "__main__":
    add_two_ints_server()
```

ROS Software Development

- Srv: msgName.srv
  - Client

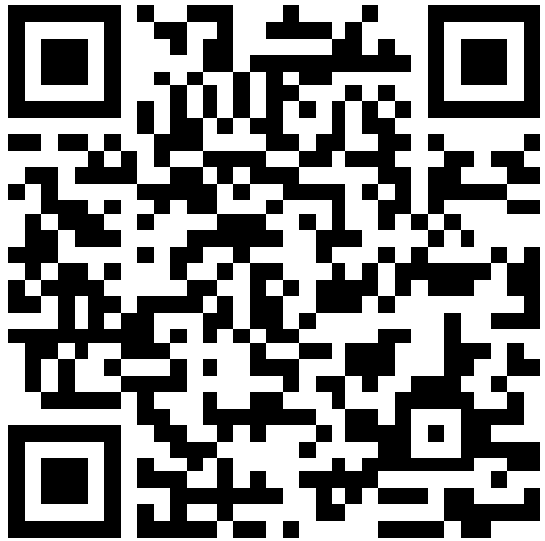
```
def add_two_ints_client(x, y):
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
        resp1 = add_two_ints(x, y)
        return resp1.sum
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e

def usage():
    return "%s [x y]"%sys.argv[0]

if __name__ == "__main__":
    if len(sys.argv) == 3:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
    else:
        print usage()
        sys.exit(1)
    print "Requesting %s+%s"%(x, y)
    print "%s + %s = %s"%(x, y, add_two_ints_client(x, y))
```

Demo

- Publisher and subscriber
  - Hello word
  - Camera (android and labptop)
- Server and Client
  - Add two number
- Robot
  - rqt simulator
  - Speech control
  - Look at me

Todo

- Implant current code to the Kobuki board
- Enhance the camera
  - recognize more objects like face, wall, desk etc
  - openCV, deep learning
  - Speech recognition: Api.ai, Google Cloud Speech API
- Setup ROS master on server
- Gitbook
  - https://www.gitbook.com/book/jellylidong/ros-development-note/details
  - Document what have achieved



Thank you