ROS ON MULTIPLE MACHINES

ROBOTICS

POLITECNICO
MILANO 1863

ROS DISTRIBUTED

Ros can work as a distributed system on multiple devices connected to the same network

We will use the network calle “robotics”, pwd “robo1”.

Do not setup static ip on the network

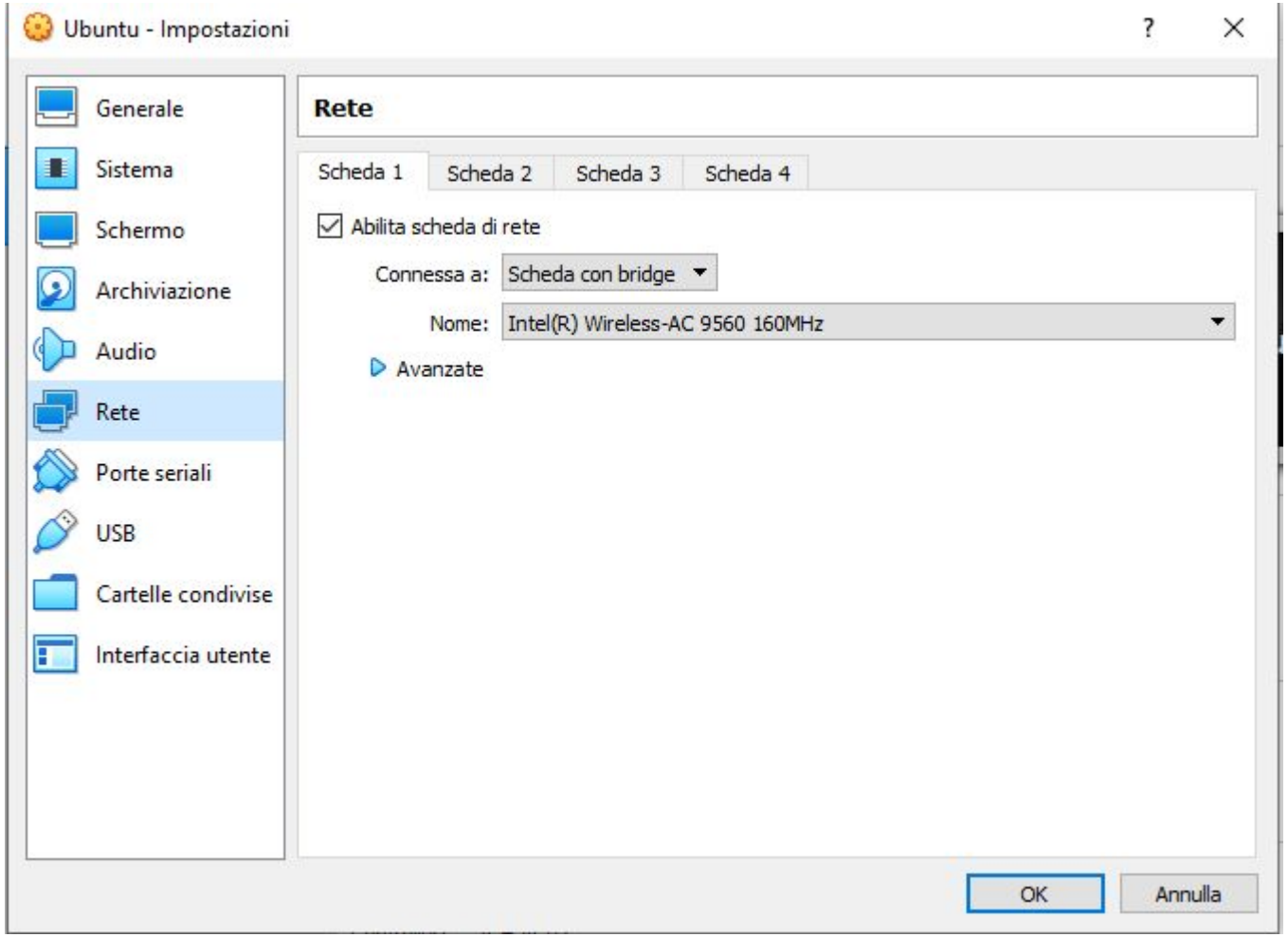
You can also create your hotspot and work as a master of a new network

To use ROS on multiple device you need to run the ROS master, command “roscore”, only on one of the device

For all the other nodes you need to specify the ip of the master

ROS DISTRIBUTED

Virtualbox configuration



COMMON CONFIGURATION

Get your ip: “ifconfig command”-> “inet addr”

Export all the variables to properly configure the master, to set all those variables for every new terminal add them at the end of your ~/.bashrc

$ gedit ~/.bashrc

MASTER CONFIGURATION

First we set the master IP:

```
export ROS_MASTER_URI=http://master_ip:11311
```

First we set the master URI

your IP

Standard ROS port, you can also run on different ports

MASTER CONFIGURATION

Tell ROS master my IP:

```
export ROS_IP=master_ip
```

← your IP

Tell ros my ip

CLIENTS CONFIGURATION

First we set the master IP:

```
export ROS_MASTER_URI=http://master_ip:11311
```

First we set the master URI

master ip

Standard ROS port, you can also run on different ports

CLIENTS CONFIGURATION

Tell ROS master my IP:

```
export ROS_IP=master_ip
```

← your IP

Tell ros my ip

ROS DISTRIBUTED

On the **master** pc use the command “roscore”

To test if everything is working on the **clients** open a new terminal and call “rostopic list” without previously running “roscore”. You should be able to see topics on the ROS network.

Now all client are on the same network and can communicate and start node on the distributed ROS netowrk

TIME SYNCHRONIZATION

Recording high-throughput bags often requires to split the recordings on different ROS devices, to use the bags all together they need to have coherent timestamp.

We then need to synchronize the clock of all the devices on the ROS network.

The standard procedure to synchronize multiple devices on a local network is tu use an ntp server on a master device and a chrony client for all the other devices.

In a ROS network the procedure is to install the NTP server on the master and chrony on all the other nodes.

MASTER CONFIGURATION (/etc/ntp.conf)

```
driftfile /var/lib/ntp/ntp.drift

statistics loopstats peerstats clockstats

filegen loopstats file loopstats type day enable

filegen peerstats file peerstats type day enable

filegen clockstats file clockstats type day enable

pool 0.ubuntu.pool.ntp.org iburst

pool 1.ubuntu.pool.ntp.org iburst

pool 2.ubuntu.pool.ntp.org iburst

pool 3.ubuntu.pool.ntp.org iburst

server 127.127.1.0

fudge 127.127.1.0 stratum 10

pool ntp.ubuntu.com

restrict -4 default kod notrap nomodify nopeer noquery limited

restrict -6 default kod notrap nomodify nopeer noquery limited

restrict 192.0.0.0 mask 255.0.0.0 nomodify notrap

restrict 127.0.0.1

restrict ::1
restrict source notrap nomodify noquery
```

CLIENT CONFIGURATION (/etc/chrony/chrony.conf)

```
server 192.168.0.100 minpoll 2 maxpoll 4
initstepslew 2 192.168.0.100
keyfile /etc/chrony/chrony.keys

commandkey 1
driftfile /var/lib/chrony/chrony.drift
maxupdateskew 5
dumponexit
dumpdir /var/lib/chrony
pidfile /var/run/chronyd.pid
logchange 0.5
rtcfile /etc/chrony.rtc
rtconutc
rtcdevice /dev/rtc
sched_priority 1
local stratum 10
allow 127.0.0.1/8
```

← Master IP

TIME SYNCHRONIZATION

The chrony configuration file can be found in \etc\chrony\chrony.conf

Then stop and restart chrony to make those changes effective:

$ sudo service chony stop

$ sudo service chony start

Then to monitor how synchronization is doing:

$ chronyc tracking

ROSPY

ROBOTICS

POLITECNICO
MILANO 1863

ROSPY

Python 2.7

With some changes ROS works also with 3.6

Files saved in the script folder

Start node with same syntax as c++ node but using the name of the executable instead of the node:

rosrun package_name python_file.py

ROSPY (publisher)

```
import rospy
from std_msgs.msg import String
```

← Standard rospy include and std_msgs include

```
if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

← Main function

← Start the tolker

ROSPY (publisher)

```
def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10)
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()
```

Create the publisher

Initialize the node

Set the loop rate

Keep spinning

Rospy version of ROS_INFO

Publish the message

ROSPY (subscriber)

```
import rospy
from std_msgs.msg import String

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

Standard include

Main function

ROSPY (subscriber)

```
def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():


    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    rospy.spin()
```

Callback function

Init the node

Create the subscriber

ROSPY (bag operation)

```
import rosbag
bag = rosbag.Bag('test.bag')
for topic, msg, t in bag.read_messages(topics=['chatter', 'numbers']):
    print msg
bag.close()
```

import lib to handle bag files

Import the bag file

Filter specific topic

Cycle through the topic and messages