Technical Report

IRI-TR-15-1

Multi-master ROS systems

Sergi Hernandez Juan
Fernando Herrero Cotarelo

January, 2015

Institut de Robòtica i Informàtica Industrial

Abstract

This technical report introduces the concepts, problems and a possible solution for ROS multi-master systems, that is, systems build from two or more ROS networks, each with its own *roscore* node. In general this environment would correspond to multi-robot systems, either mobile platforms or manipulators.
The ROS framework already provides a solution for such systems, *multimaster_fkie*, which is presented and briefly described in this technical report, together with the network setup necessary to make it work properly.
Two different configurations are discussed in this technical report, simple ROS networks with a single computer each, and more complex ROS networks with two or more computers each. In both cases, real examples are provided using robots available at IRI.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
Sergi Hernandez Juan
tel: +34 93 401 0857
shernand@iri.upc.edu
http://www.iri.upc.edu/staff/shernand

Copyright IRI, 2015

1 introduction

Within the ROS framework, in general each robot has its own ROS network (either wired, wireless or a combination of both), where a single *roscore* manages all the communications between all the ROS nodes, either in a single computer or multiple computers in the same network (depending on the complexity and computational needs of the system).

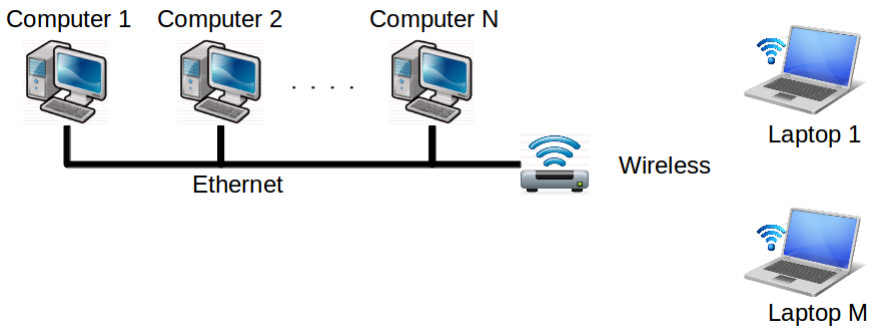

Figure 1: Sketch of a possible hardware setup of a simple ROS system.

See Fig. 1 for a generic hardware setup for such configuration and Fig. 2 for a possible simple node configuration in such architecture. For details on how to setup such system, refer to [1].

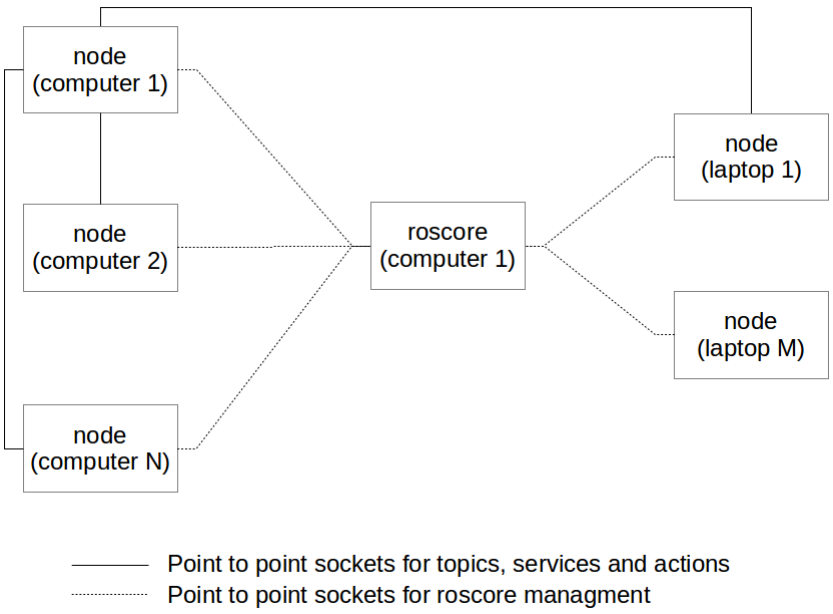

Figure 2: Possible interconnection of several ROS nodes around a single *roscore* node. In brackets, the computer where each node is executed

In Fig. 1 the wireless connectivity is achieved by means of wireless router, but it could also be achieved by using a desktop computer with a wireless interface card.

When multiple robots need to exchange information, there exist mainly two possible solutions, create a single large ROS network managed by a single *roscore* node, or create a mechanism to allow information to be exchanged between ROS sub-systems. This technical report focuses on the later, because the former can be easily implemented using common ROS tools.

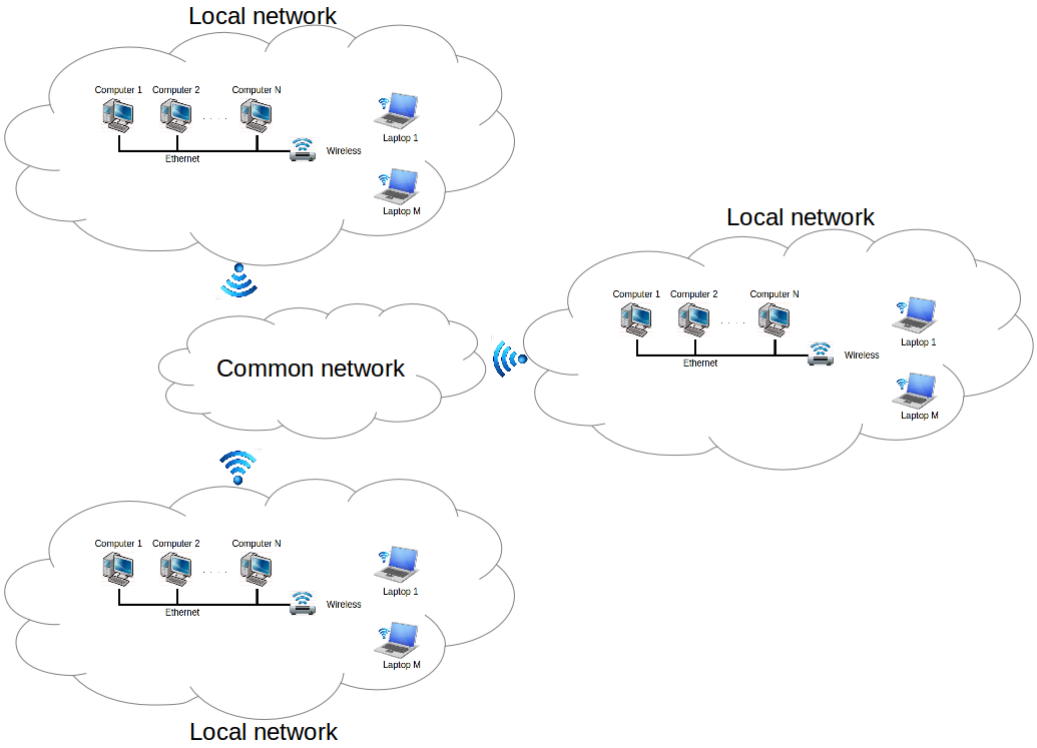

Figure 3: Sketch of a possible hardware setup of multiple ROS sub-systems communicating to each other.

See Fig. 3 and Fig. 4 for a possible hardware and software configurations respectively for this kind of system. Note that in Fig. 3 the common network could be either wireless (for mobile platforms), wired (for static manipulators) or a combination of both.

In Fig. 4, the solid red lines indicate communications between nodes in different ROS subsystems. Theses communications could be either topics, services or actions. Solid black lines indicate local communications.

The ROS framework already provides a solution for multi-master systems called *multimaster_fkie*, which is presented in section 2.

2 multimaster_fkie

Here, a brief description of the *multimaster_fkie* is presented. For a more detailed description of the features and operation of this node, please check the following web page [2].

This standard ROS package can be easily installed as a debian package for your ROS version, in this case, Indigo.

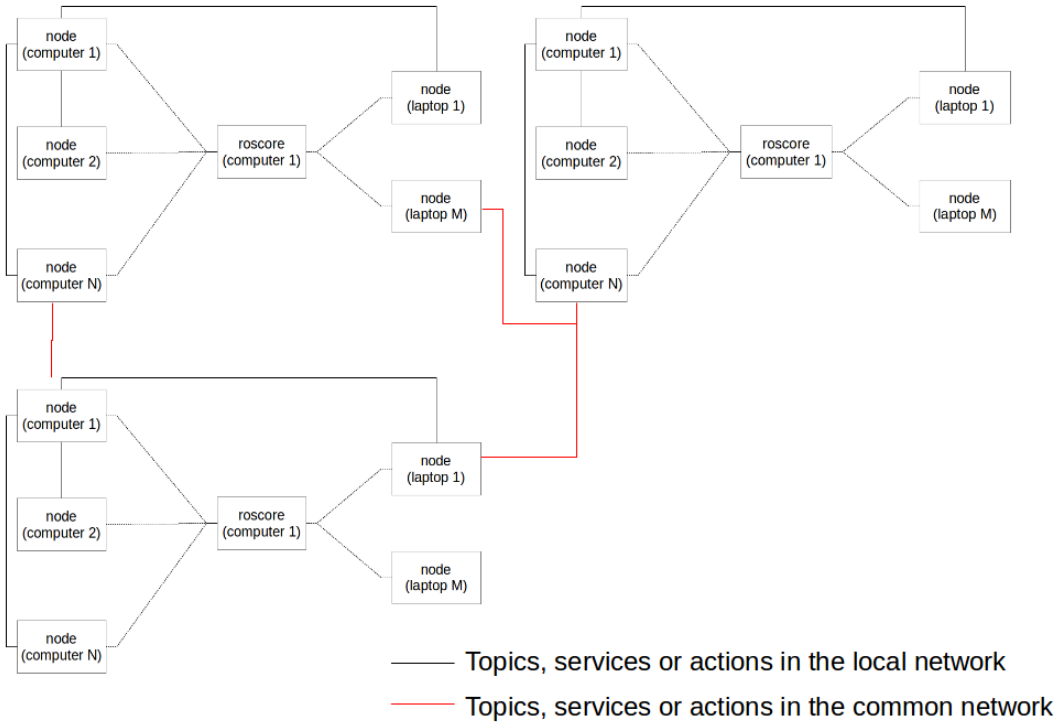

Figure 4: Possible interconnection of several ROS nodes in a set up with multiple ROS sub-systems, each with its own *roscore* node.

```
sudo apt-get install ros-indigo-multimaster-fkie
```

The *multimaster_fkie* consists mainly of two nodes: the *master_discovery* and the *master_sync* nodes. The main features of the *master_discovery* node are summarized next:

- Periodically sends mulicast messages to the common network to make the other possible ROS masters aware of its presence, and also detect any other ROS masters available.
- Checks the local *roscore* for changes in the local network, and notify all other ROS masters in the common network of this changes.

The main features of the *master_sync* node are summarized next:

- Uses the information provided for other *master_discovery* nodes to register remote topics and services to the local *roscore*.
- The information provided by the remote *master_discovery* nodes is also used to update the information on topics and services.
- It can be configured to select which hosts, topics and services to synchronize or ignore. By default all topics and services of all hosts are synchronized, so in order to reduce

the required bandwidth, it is a good practice to synchronize only the absolute minimum number of topics and services.

The *master_sync* node only synchronizes remote topics or services to the local *roscore*. In order to do the opposite, the remote *roscore* must have its own *master_sync* node.

The *multimaster_fkie* solution only supports topics and services, however actions are implicitly supported because they are build on top of 5 topics. The parameter server of each *roscore* can not be accessed remotely.

Similarly to what happens in a standard ROS system, once the topics, services and/or actions are registered on the remote ROS sub-systems, a point to point socket is created to directly connect two or more nodes, and the multi-master mechanism is not used until there is a change in the configuration (a new publisher/subscriber appears or an existing one disappears, etc.).

This multi-master solution is currently supported both in ROS Hydro and ROS Indigo, but only under the catkin framework.

The network configuration for single computer ROS networks is simpler compared to that of multiple computers systems, therefore it is presented first in section 3. Next, the additional configuration steps necessary to properly setup multiple computer ROS networks is presented in section 4.

3 Single computer ROS networks

In this section, first the basic requirements for the network setup will be introduced in section 3.1, and next, a step by step procedure will be presented in section 3.2. Finally, in section 3.3 some basic testing of the *multimaster_fkie* framework is presented.

For the sake of clarity, the system configuration shown in Fig. 5 will be used through out this section to exemplify some of the configuration steps.

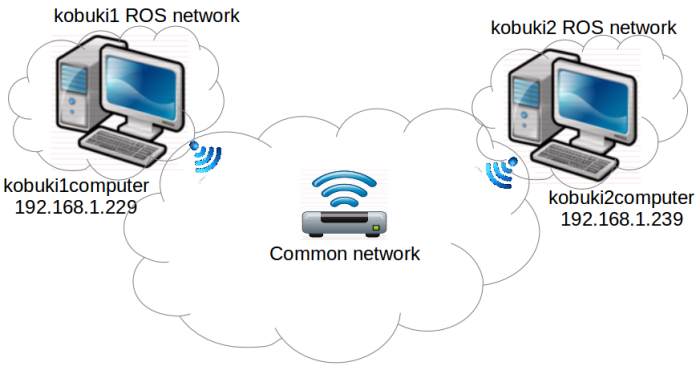

Figure 5: Simple network setup for the single computer ROS network case used in section 3 to exemplify the network configuration

3.1 Network configuration

Since the *multimaster_fkie* does not support to remotely launch a ROS node on a different ROS network, it is not necessary to exchange public keys between the computer on the different ROS networks to allow password-less access to the different computers.

However, if no DNS server is available on the network to provide a binding between the host names and their IP addresses, this binding must be done manually modifying the */etc/hosts* file. This is needed to allow proper operation of the ROS framework.

Each *master_discovery* node sends and receives information to and from other *master_discovery* nodes in the same common network using the multicast feature. Today, almost all network adapters support these feature, however, in Ubuntu systems, it is disabled by default.

The multicast feature is very important in multi-master ROS systems based on *multimaster_fkie*, because if not enabled, although the initial setup and operation appear to be correct, after 10 minutes or so, all the inter-network connections will be closed.

Besides enabling the multicast feature, it is also necessary to choose a multicast address group for all computers which will be used for any *master_discovery* node to notify changes to the others. By default the *226.0.0.0* address is used, but it can be changed using a parameter of the node.

Once the local ROS nodes have the information of the remote nodes provided by the *master_discovery* and *master_sync* nodes, the network traffic intended for remote networks must be properly routed in order to allow the communication between nodes on different ROS networks.

This can be easily achieved by adding static routes to the router, access point or computer used to manage the common network. However, in order to properly route the traffic to the desired destination, each computer must have their own and unique IP address. In a simple single computer case, this may be a problem, because all the local ROS networks will be operating on the localhost (*127.0.0.1*) address by default.

The simplest solution in this case is to use the IP address of the common network assigned to each computer as their *ROS_MASTER_URI* variable. By doing so, no static routes need to be added because all computers share the same network interface, although they are using separate *roscore* nodes.

When the local networks have several computers each, this solution is no longer valid. An alternate solution is presented later in section 4.

3.2 Step by step procedure

In this section, a step by step procedure is presented which implements the configuration explained in section 3.1.

Host name and IP address binding

For each computer, modify the */etc/hosts* file using your favorite text editor. Fig. 6 shows the contents of this file for the kobuki1computer computer in the example presented in Fig. 5.

```
127.0.0.1       localhost
127.0.1.1       kobuki2computer

192.168.1.239   kobuki2computer

192.168.1.229   kobuki1computer

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figure 6: Contents of the */etc/hosts* file for the kobuki2computer computer for the example presented in Fig. 5

In order to check the proper configuration of the network, before launching any ROS nodes, try to ping all other computers from any computer, using both the host name and the IP address. If all the pings return a reply, the network has been properly configured.

Change the *ROS_MASTER_URI* variable

For each computer, to temporarily change the contents of this variable and only in the current console, execute the following command, where *hostname or IP address* must be the IP address or host name of each computer.

```
export ROS_MASTER_URI=http://<hostname or IP address>:11311
```

In the example presented in Fig. 5, the *ROS_MASTER_URI* variable in the kobuki1 ROS network should be set to either *kobuki1computer* or *192.168.1.229*.

To make this change permanent, edit the *~/.bashrc* file with your favorite text editor and add the previous command at the end. Then close the terminal and open a new one, or apply the changes by executing the following command:

```
source ~/.bashrc
```

Enable multicast

For all computers, first check if the multicast feature is enabled using the following command.

```
cat /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

If this command returns 0, the multicast feature is enabled and you can go ahead to the next section. To temporary enable the multicast feature, execute the following command, however, when the computer restarts, this configuration will be lost.

```
sudo sh -c "echo 0 >/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"
```

To permanently enable the multicast feature, edit the */etc/sysctl.conf* file and add the following line, or uncomment it, if it already exists, and change its default value.

```
net.ipv4.icmp_echo_ignore_broadcasts=0
```

In order for the changes to take effect, execute the following command:

```
sudo service procps restart
```

To check which multicast groups are already defined in a computer, execute the following command.

```
netstat -g
```

This command will report all the IP addresses enabled for multicast for each of the network interfaces available, both for IPv4 and IPv6. The standard IP address for multicast is *224.0.0.1*, that should appear on the list provided by the last command, and it is the one we will use.

At this point, to check whether the multicast feature is working or not, execute the following command, at any computer.

```
ping 224.0.0.1
```

If everything is configured properly, you should get a reply from each computer in the common network at each iteration.

3.3 Testing

Once the network has been configured, it is time to launch the *multimaster_fkie* nodes and take a first look at how it works.

First of all, launch a local *roscore* at each computer.

```
roscore
```

Then, launch the *master_discovery* node at each computer, passing as an argument the *mcast_group* parameter to specify the multicast address to be used.

```
rosrun master_discovery_fkie master_discovery _mcast_group:=224.0.0.1
```

This command will report some information on screen (see Fig. 7), but nothing else happens yet. It is important to note if the reported multicast address used is the one provided to ensure everything is set up properly.

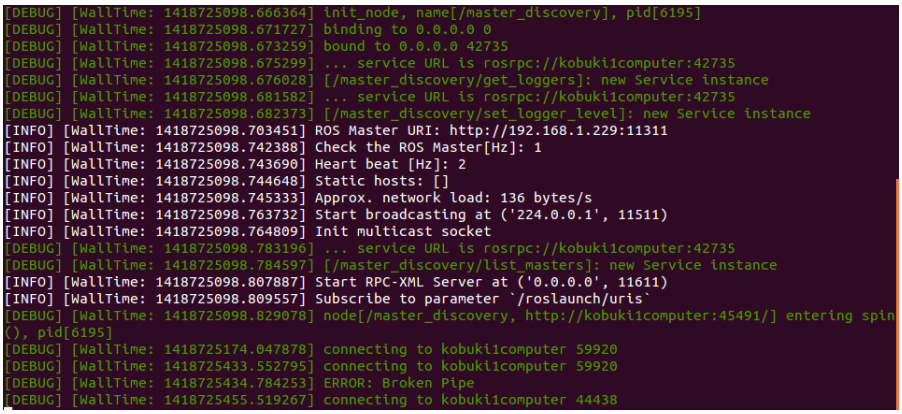

Figure 7: Information reported by the *master_discovery* node for the example system presented in Fig. 5, when it is launched

Afterwards, launch the *master_sync* node at each computer. Without any additional parameter, all topics and services on all computers will be synchronized to all others.

```
rosrun master_sync_fkie master_sync
```

Note that, on each computer, the other computers running a *master_discovery* node in the same common network are automatically detected, and the time stamp difference between them is reported.

When both nodes are running on all computers, executing the following command will list all the available ROS masters on the common network. Fig. 8 shows the information reported by this command for the example system presented in Fig. 5.

```
rosservice call /master_discovery/list_masters
```

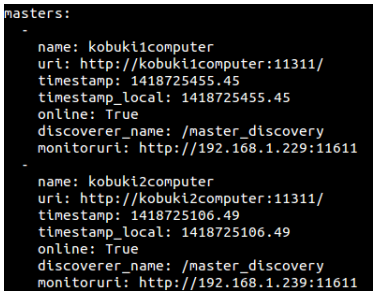

Figure 8: Information reported by the *rosservice call* command on service */master_discovery/list_masters* for the example system presented in Fig. 5.

At this point, everything is set up to try communicate between two ROS networks. For simplicity we will use the *rostopic pub* and *rostopic echo* commands, but any existing set of nodes should be able to work on different ROS networks as they did in a single one.

If the following command is executed in one local ROS network, all other computers should be able to see the topic with the *rostopic list* command, and also to gets its contents with the *rostopic echo* command.

```
rostopic pub -r 1 /test std_msgs/Int32 1
```

Also the *master_sync* node of all computers, will also report when someone subscribes or unsubscribes from the published topic, and also when a new topic is advertised or unadvertised.

Finally, the *rostopic info* command will show that the connection is established directly between the publisher and subscriber, and the *master_discovery* and *master_sync* nodes are not involved (see Fig. 9).

4 Multiple computers ROS networks

All the concepts introduced in section 3 are also applicable to the multi-computer ROS networks case. In this case, however, additional configuration steps are necessary to allow all the computers in all the local ROS networks to exchange information.

For the sake of clarity, the system configuration shown in Fig. 10 will be used through out this section to exemplify some of the configuration steps.

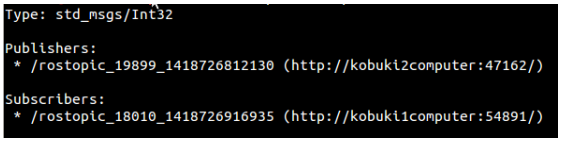

Figure 9: Publisher and subscriber information about the simple topic used in this example. The *master_discovery* and *master_sync* nodes are not involved.

4.1 Network configuration

In this configuration, all the computers in a local ROS network will share a local network, but only one computer (or access point or router) will have direct access to the common network for the multi-master ROS system.

Therefore, the device with access to both networks (local and common) will work as the gateway between them. That is, all local traffic intended for a computer on an remote network, will be first routed to the local gateway, then to the gateway of the desired remote network, and from there to the desired computer on that remote network.

In the example presented in Fig. 10, the *tibi-visio*, *dabo-visio* and *car-visio* computers work as the gateways for each of the local ROS networks.

To achieve this behavior, it is first necessary to explicitly set the IP address of the gateway device on each of the computers of the local ROS network, so that the remote traffic will be routed there. The gateway device must be also configured to forward the traffic intended for other networks (*ip_forward* property).

Also, on the device that handles the common network (either a computer, router or access point), it is necessary to add the static routes in order to properly route the traffic to the desired network.

As explained in section 3, the communication in ROS are point to point between two or more nodes. So, there must exist a binding between the IP address and the host name for all the computers in all the networks in order for the ROS framework to work properly.

The *ROS_MASTER_URI* variable can be set to any computer inside the local ROS network, it does not need to be the computer with access to the common network.

4.2 Step by step procedure

In this section, a step by step procedure is presented which implements the configuration explained in section 4.1. Some steps from section 3.2 are repeated here with little or no change at all for clarity.

Change the *ROS_MASTER_URI* variable

For all computers on a local ROS network, and for all ROS networks, execute the following command to make the changes only temporarily, where *hostname or IP address* must be the IP address or host name of the computer chosen to execute the *roscore* node.

```
export ROS_MASTER_URI=http://<hostname or IP address>:11311
```

In the example presented in Fig. 10, the *ROS_MASTER_URI* variable for all the computers in the dabo ROS network could be set, for example, to *192.168.10.101* or *dabo-vis*.

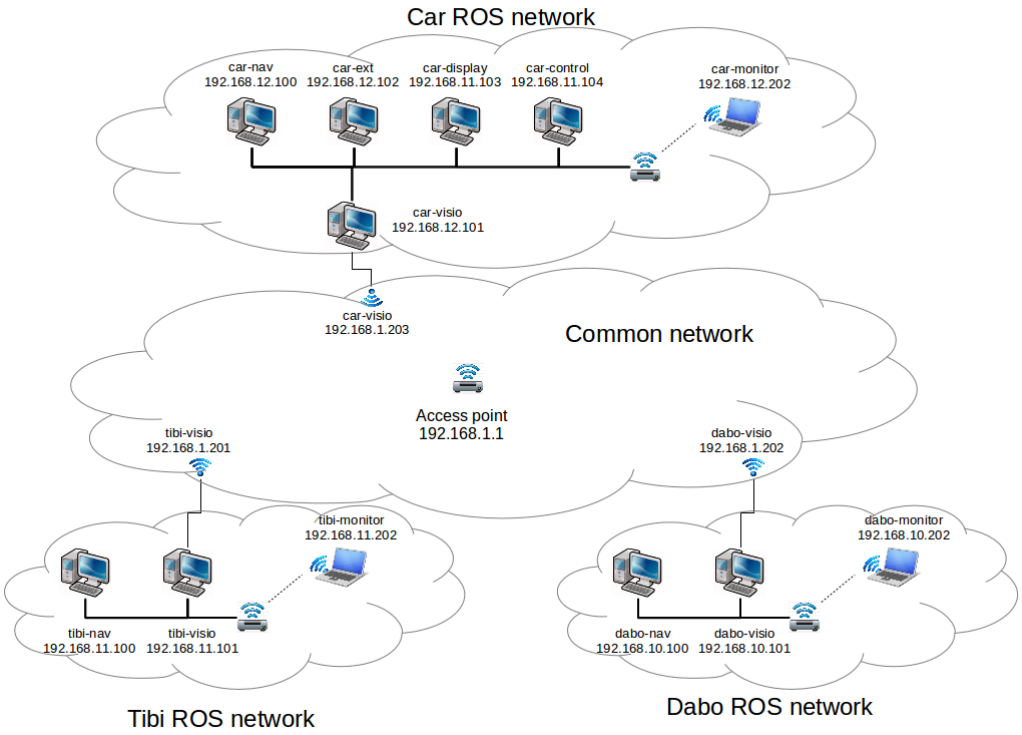

Figure 10: Network setup for the multiple computer ROS network case used in section 4 to exemplify the network configuration

To make this change permanent, edit the *~/.bashrc* file with your favorite text editor and add the previous command at the end. Then close the terminal and open a new one, or apply the changes by executing the following command:

```
source ~/.bashrc
```

Set the default gateway

For all the computers on a local ROS network (except for the one working as gateway) and for all the ROS networks, do the following. To temporary set the default gateway, execute the following command, where *IP address* is the IP address of the device working as the gateway.

```
route add default gw <IP address>
```

If multiple network interfaces are available, it will be necessary to append *dev <interface>* to the previous command, where *interface* is the network interface through which the current computer is connected to the gateway.

To make this change permanent, it is necessary to create a simple script that will be executed whenever the desired network interface is brought up. In the */etc/network/if-up.d/* folder, create a new file (with any name you want) and add the code shown in Fig. 11. When done, add execution permissions to the file so that it can be called whenever necessary.

```
cd /etc/network/if-up.d
vim multimaster
chmod a+x multimaster
```

```
#!/bin/sh

if [ "$IFACE" = "<interface>" ]; then
  route add default gw <gateway IP address>
fi
```

Figure 11: Simple script to set the IP address of the default gateway on a network

In Fig. 11, *interface* refers to the network interface (*eth0,wlan0, etc.*) through which the current computer is connected to the gateway, and *gateway IP address* is the IP address of the gateway device.

Note that with this script it is not necessary to append the *dev <interface>*, because the interface is already specified as the condition to execute the command.

Enable *ip_forward* on the gateway

Only for the computer working as the gateway on each of the local ROS networks, first check if the IP forwarding feature is enabled using the following command.

```
cat /proc/sys/net/ipv4/ip_forward
```

If this command returns 1, the IP forwarding is enabled and you can go ahead to the next section. To temporary enable this feature, execute the following command, however, when the computer restarts, this configuration will be lost.

```
sudo sh -c "echo 1 >/proc/sys/net/ipv4/ip_forward"
```

To permanently enable the IP forwarding feature, edit the */etc/sysctl.conf* file and add the following line, or uncomment it if it already exists, and change its default value.

```
net.ipv4.ip_forward=1
```

In order for the changes to take effect, execute the following command:

```
sudo service procps restart
```

Add static routes between networks

Only for the device managing the common network do the following. In the case of using a desktop computer under Linux, execute the following command:

```
route add -net <local network> netmask 255.255.255.0 gw <common network gateway>
```

Where *local network* is any of the local ROS network domains, and *common network gateway* is the IP address, on the common network, of the computer that works as the gateway for the corresponding local ROS network.

From the example presented in Fig. 10, the command to properly route any traffic intended for the Tibi ROS network, will be:

```
route add -net 192.168.11.0 netmask 255.255.255.0 gw 192.168.1.201
```

For a router or an access point, the main idea remains the same, but the specific details would depend on the manufacturer and model of the device. Check the product help to properly configure the traffic routing.

Enable multicast

For all computers on a local ROS network, and for all ROS networks, first check if the multicast feature is enabled using the following command.

```
cat /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

If this command returns 0, the multicast feature is enabled and you can go ahead to the next section. To temporary enable the multicast feature, execute the following command, however, when the computer restarts, this configuration will be lost.

```
sudo sh -c "echo 0 >/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"
```

To permanently enable the multicast feature, edit the */etc/sysctl.conf* file and add the following line, or uncomment it, if it already exists, and change its default value.

```
net.ipv4.icmp_echo_ignore_broadcasts=0
```

In order for the changes to take effect, execute the following command:

```
sudo service procps restart
```

To check which multicast groups are already defined in a computer, execute the following command.

```
netstat -g
```

This command will report all the IP addresses enabled for multicast for each of the network interfaces available, both for IPv4 and IPv6. The standard IP address for multicast is *224.0.0.1*, that should appear on the list provided by the last command, and it is the one we will use.

At this point, to check whether the multicast feature is working or not, execute the following command, at any computer.

```
ping 224.0.0.1
```

If everything is configured properly, you should get a reply from each computer in the common network at each iteration.

Host name and IP address binding

For all computers on a local ROS network, and for all ROS networks, modify the */etc/hosts* file using your favorite text editor. Fig. 12 shows the contents of this file for the tibi-monitor computer in the example presented in Fig. 10.

```
127.0.0.1       localhost
127.0.1.1       tibi-monitor

192.168.11.202   tibi-monitor
192.168.11.100   tibi-nav
192.168.11.101   tibi-visio

192.168.10.100   dabo-nav
192.168.10.101   dabo-visio
192.168.10.202   dabo-monitor

192.168.12.100   car-nav
192.168.12.101   car-visio
192.168.12.102   car-ext
192.168.12.103   car-display
192.168.12.104   car-control
192.168.12.202   car-monitor
```

Figure 12: Contents of the */etc/hosts* file for the tibi-monitor computer for the example presented in Fig. 10

In order to check the proper configuration of the network, before launching any ROS nodes, try to ping all other computers from any computer, using both the host name and the IP address. If all the pings return a reply, the network has been properly configured. If not, review the steps listed before.

4.3 Testing

The operation of the *multimaster_fkie* with multiple computer ROS networks is exactly the same as with single computer ROS networks. So, refer to section 3.3 to test whether the network has been properly configured or not.

The only difference is that, in this case, the communication should be possible between any two computers on any two ROS networks.

5 Conclusions

The *multimaster_fkie* documentation webpage is pretty complete regarding the configuration and operation of its nodes, but it makes little or no comment on how to set up the system so that the multi-master system works properly. This is the main reason why this document has been written.

While the operation of this framework is pretty good for wired connections, its performance rapidly decreases when using wireless interfaces due to power issues and interferences. This has nothing to do with the framework itself, but special care should be taken when using wireless connections.

Compared to previous versions of multi-master solutions in ROS, the one presented in this document presents several advantages: simplicity of use, since the user only have to select which information to share, and periodic updates, which keeps the local network synchronized with the global network information.

References

[1] IRI. Multiple machines. `http://wiki.iri.upc.edu/index.php/Multiple_machines`, july 2014.

[2] Alexander Tiderko. multimaster_fkie. `http://wiki.ros.org/multimaster_fkie`, October 2014.

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.