Automation and iCub applications

5th iCub Summer School, Sestri Levante, July 2010

Running processes remotely

- Yarp provides a way to run modules remotely
- yarprun is a server that waits for commands on a port
- start/termination/kill monitor lifecycle

  http://eris.liralab.it/yarpdoc/db/dd7/yarprun.html

Overview: yarprun

run a server, which will wait for commands on /machine1

Starting a server
$node1: yarprun –server /node1

Execute a command, from node2:
$node2: yarprun --on /node1 --as TAG --cmd COMMAND [ARGLIST]

Other commands:
$node3: yarprun --on /node1 --isrunning TAG
$node3: yarprun --on /node1 --ps
$node3: yarprun --on /node1 --sigterm TAG

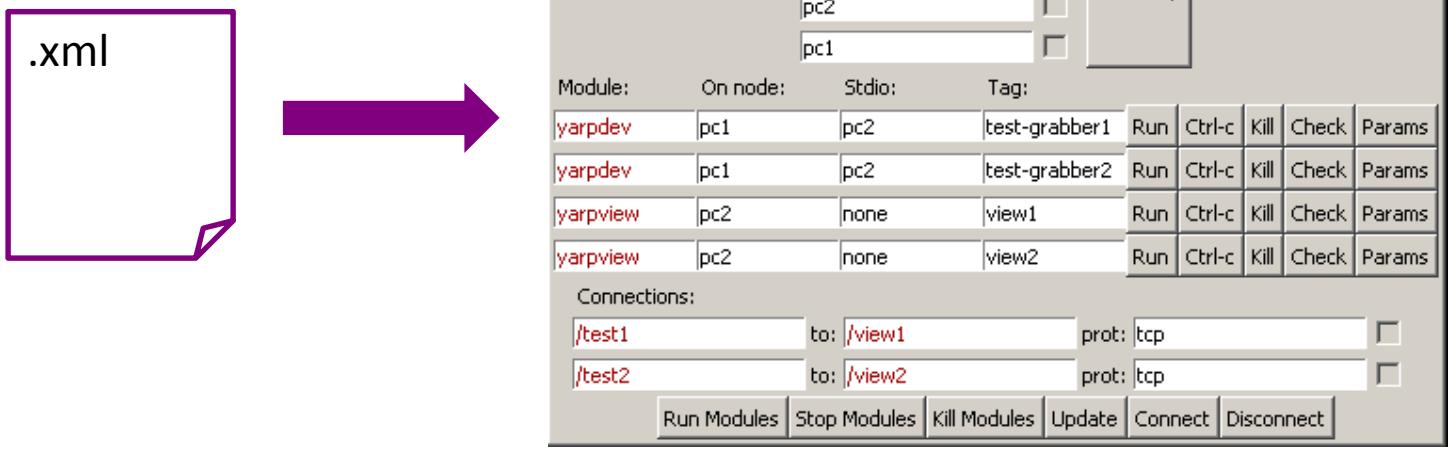
Redirecting stdio

- When you run a module on a remote machine you sometimes want to have access to the module input/output

yarprun --on /node1 --as TAG --cmd COMMAND [ARGLIST] --stdio /node2

The Manager

- The application manager is a python script that provides a graphic interface to yarprun
- It allows to start/stopping/monitor, redirect i/o
- In addition it automates establishing connections between modules



See Section 9.3 of the Manual: Automation

Syntax

```
<application>
        <name>Name of the application</name> //this can be anything, just a symbolic name

        <dependencies>
                <port>/port1 </port>
                <port>/port2 </port>
        </dependencies>

        <module>
                <name>mymodule1 </name>
                <parameters>--threshold 1 --name /myName</parameters>
                <node>node1</node>
                <tag>module-tag</tag>
        </module>

        <module>
                <name>mymodule2</name>
                ...
        </module>

        <connection>
                <from>/port1</from>
                <to>/otherport</to>
                <protocol>udp</protocol>
        </connection>

        <connection>
                …
        </connection>

</application>
```

An example

We want to start a test_grabber device and a viewer

yarpdev –device test_grabber –name /grabber
yarpview –name /viewer

yarp connect /grabber /viewer

```
<application> <name>Name of the application</name> //this can be anything, just a symbolic name

        <module>
                <name>yarpdev</name>
                <parameters>--device test_grabber --name /grabber</parameters>
                <node>lorenzo</node>
                <tag>grabber</tag>
        </module>

        <module>
                <name>yarpview</name>
                <parameters>--name /viewer</parameters>
                <node>lorenzo</node>
                <tag>viewer</tag>
        </module>

        <connection>
                <from>/grabber</from>
                <to>/viewer</to>
                <protocol>udp</protocol>
        </connection>

</application>
```

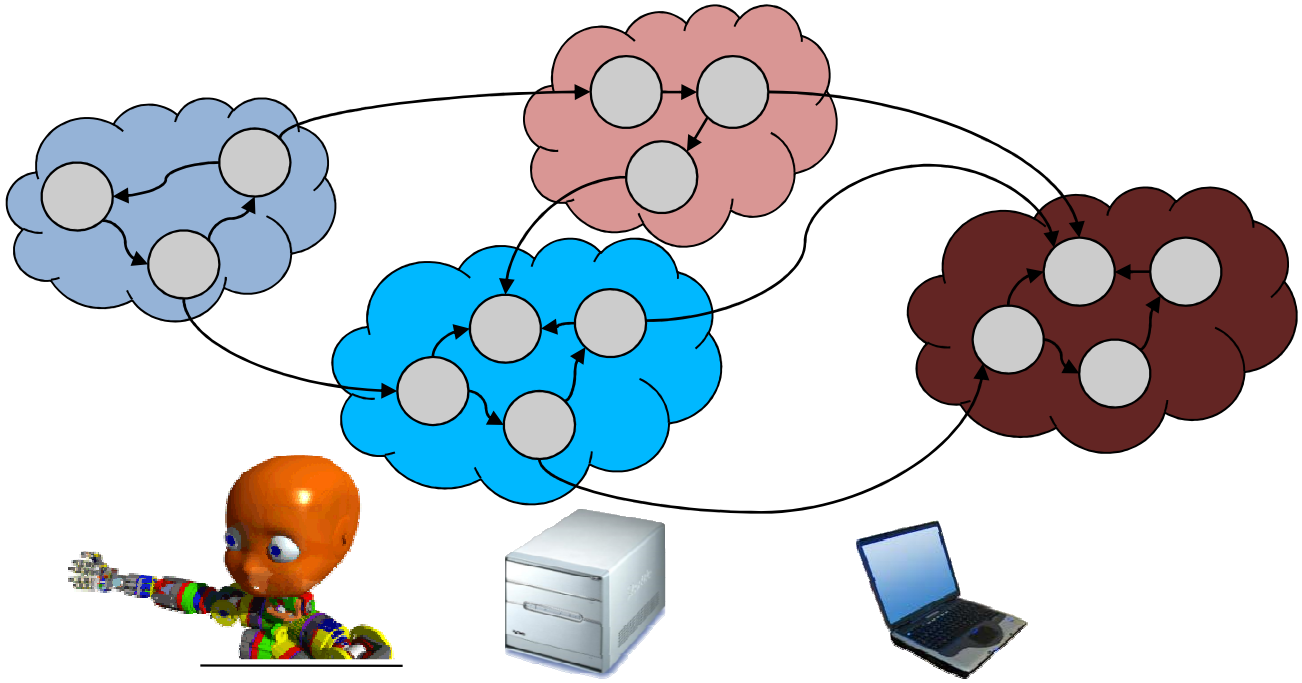
Other tags

```
<dependencies>
    <port>/icub/cam/left</port>
    <port>/icub/cam/right</port>
</dependencies>

<module>
    …
    <workdir>C:/mydir</workdir>
    <stdio>node3</stdio>
</module>
```

iCub Applications

- An *application* is a logical collection of *modules* with a particular purpose (e.g. attention system, reaching, learning of affordances, drumming interaction history…)
- In practice it is just a collection of configuration files and a text description in xml



How are parameters organized

- Parameters of module are organized in a directory called “app”
- Each entry inside app store parameters for using a module in a certain way (*context*)
- The ResourceFinder class in yarp helps to detect these files (resources)
- http://eris.liralab.it/iCub/dox/html/icub_resource_finder_basic.html
- http://eris.liralab.it/iCub/dox/html/icub_resource_finder_advanced.html