RoCKIn@Work Central Factory Hub Manual

Sven Schneider

August 12, 2015

1 Introduction and motivation

2 Setup and installation

[1]

2.1 Prerequisites

2.1.1 Arch Linux

1. ...

2.1.2 Ubuntu 12.04

Install pre-built packages via the operating system's package manager:

```
# add-apt-repository ppa:timn/clips
# apt-get update
$ apt-get install libmodbus-dev libclips-dev clips libclipsmm-dev
    protobuf-compiler libprotobuf-dev libprotoc-dev boost1.50-all-
    dev libglibmm-2.4-dev libgtkmm-3.0-dev libncursesw5-dev libyaml
    -cpp-dev libavahi-client-dev git
# apt-get install libssl-dev libelf-dev mongodb-dev mongodb-clients
     mongodb libzmq3-dev
```

2.1.3 Ubuntu 14.04

Install pre-built packages via the operating system's package manager:

```
# apt-get install libmodbus-dev protobuf-compiler libprotobuf-dev
    libprotoc-dev libboost-all-dev libglibmm-2.4-dev libgtkmm-3.0-
    dev libncursesw5-dev libyaml-cpp-dev libavahi-client-dev git
    libxt-dev libxaw7-dev libncurses5-dev autoconf autogen libtool
    libyaml-dev
# apt-get install libssl-dev libelf-dev mongodb-dev mongodb-clients
     mongodb libzmq3-dev
```

[1] `https://github.com/rockin-robot-challenge/at_work_central_factory_hub`

Build and install CLIPS from source

```
$ mkdir <workspace>
$ cd <workspace>
$ wget http://downloads.sourceforge.net/project/clipsmm/clips
    /6.30.0.20090722svn/clips-6.30.tar.gz
$ tar -xf clips-6.30.tar.gz
$ ./configure
$ make
# make install
```

Build and install clipsmm from source

```
$ wget http://downloads.sourceforge.net/project/clipsmm/clipsmm/
    clipsmm-0.3.4.tar.bz2
$ tar -xf clipsmm-0.3.4..tar.bz2
$ ./autogen.sh
$ ./configure
$ make
# make install
```

2.2 Building

```
$ git clone https://github.com/rockin-robot-challenge/
    at_work_central_factory_hub.git -b rockin
$ make
```

Some remarks and hints on the build process:

- If you do not plan to use the real networked devices, you can ignore errors/warnings about missing ZMQ dependencies (also make sure to disable the devices in the configuration).

2.3 Configuration

The CFH offers a common file to store configuration parameters. This file is located at cfg/config.yaml. In the following the major configuration parameters are explained in specific scenarios.

2.3.1 Adding and enabling plugins (for networked devices)

A plugin mechanism enables the simple extension of the CFH. This mechanism is mainly used to integrate new networked devices into the CFH. Each plugin that should be loaded, must be added to the "llsfrb→plugins" section in the cfg/config.yaml file. If supported by the plugin, further configuration parameters can be passed to a plugin. Such a plugin and it's configuration is shown in the following listing by using the RoCKIn conveyor belt as an example:

```
llsfrb:
  ...
  plugins: "conveyor_belt"
  conveyor_belt:
    enable: true
    host: !ipv4 192.168.1.101
    status_port: !tcp-port 55501
    command_port: !tcp-port 55502
```

2.3.2 Running the CFH and robot on the same machine

If you want to execute the CFH and the robot on the same machine (here: localhost), make sure that in the cfg/config.yaml

- the public peer's send port differs from the receieve port,
- the team's private peer send port differs from the receive port; and
- all the previous ports differ from each other.

The following listing shows an example of such a configuration:

```
llsfrb:
  ...
  comm:
    ...
    public-peer:
      host: !ipv4 127.0.0.1
      send-port: !udp-port 4444
      recv-port: !udp-port 4445
    RoCKInNRoLLIn-peer:
      host: !ipv4 127.0.0.1
      send-port: !udp-port 4446
      recv-port: !udp-port 4447
```

2.3.3 Configuring streaming clients

In contrast to peers, there is only one TCP port defined for streaming client connections: llsfrb→comm→server-port. Also make sure that all clients use the right IP address and port for connecting to the CFH. This configuration is exemplified in the following:

```
llsfrb:
  ...
  comm:
    ...
    server-port: !tcp-port 4444

  shell:
    refbox-host: !ipv4 127.0.0.1
    refbox-port: 4444
```

2.3.4 Updating the list of known teams

We assume that the name of the new team is RoCKInNRoLLIn. Three adaptions are required in the configuration file:

1. Add a new peer communication end-point under llsfrb→comm with the team name (note that the team name is suffixed with "-peer"), the peer's host and UDP port.
2. Add the team name to the array in llsfrb→game→teams.
3. Add a new crypto key for the team under llsfrb→game→crypto-keys (and also provide this key to the team, so that it can decipher the communication!) [Note: Although this is not used in RoCKIn, it should still be provided!].

To remove a team, simply delete these entries. An example configuration is similar to the following:

```
llsfrb:
  ...
  comm:
    ...
    RoCKInNRoLLIn-peer:
      host: !ipv4 192.168.1.255
      port: !udp-port 4446

  game:
    teams: [RoCKInNRoLLIn]
    RoCKInNRoLLIn: randomkey
```

2.4 ROS environment

3 Usage

After the CFH has been setup and configured, it can be executed via the following command:

```
$ bin/llsf-refbox
```

Now, several tools are able to interact with the CFH:

- The **rockin-controller** provides a graphical user interface (GUI) to control the execution of the benchmark (e.g. benchmark selection, starting or stopping of a benchmark), but also control elements for the diverse networked devices.
- The **rockin-viewer** is a GUI that visualizes the current state of a benchmark and the testbed, including the elapsed time, connected robots, and the status of the networked devices, inventory and orders.

- The **rockin-fake-robot** is a command line example that emulates a robot and utilizes most of the CFH's communication capabilities. The program takes exactly two arguements: the name of the robot to emulate and the name of the team to which the robot belongs. This team name should be specified in the CFH's configuration file.
- To select benchmarks and control their state, there is also a command-line tool available which is called **rockin-benchmark-ctrl**. The program takes up to three arguments. Without arguments, it only shows the current benchmark state. With the "-p" switch a specific benchmark can be selected (possible parameters: fbm1, fbm2, tbm1, tbm2, tbm3, none). The "-s" switch forces a new benchmark state (possible parameters: init, running, paused, finished), but is strongly disadvised, since it can bring the CFH into an inconsistent state. Instead a transition request should be send to the CFH with the "-e" switch (possible parameters: reset, start, stop, pause), which enables the CFH to only perform safe transitions.
- Another command-line tool is the **rockin-device-ctrl** for controlling the networked devices. This program takes exactly two arguments. The name of the device (possible values: drilling_machine, conveyor_belt) and the command (possible values: up, down, start, stop).
- Important information is sent out by the CFH to all listening controllers as attention messages. To show these messages on the command line, use the **rockin-attention-msg-vis** program.

4 Interfacing a robot with the CFH

4.1 protobuf_comm library – Peers

4.2 Message description

5 Interfacing a controller or viewer with the CFH

5.1 protobuf_comm library – Clients

6 Interfacing a device with the CFH

6.1 Plugins

Add plugins to the rockin/plugins folder.

6.2 Message description

Define device-specific messages (plugin $\leftrightarrow$ device communication) in the rockin/plugins/msgs folder.

7 CFH-Benchmark proxy

The CFH's communication infrastructure is based on protobuf_comm (Boost ASIO and ProtoBuf), while the benchmarking system utilizes the Robot Operating System (ROS) for communication. The CFH-Benchmark proxy is an adapter to let these two systems communicate with each other to run and evaluate a benchmark.

The proxy is contained in a ROS package [2] and can be build by following the instructions in the repository. To run the proxy, execute

```
$ roslaunch at_work_cfh_bm_proxy proxy.launch
```

The node waits for the CFH and benchmarking system to become ready and does not required any further input.

8 References

[2] https://github.com/rockin-robot-challenge/at_work_cfh_bm_proxy