Distributed Deployment (NUCs)

If you select this option in the Setup Tab of the LCC (Deploy Distributed), the behavior of the whole deployment changes fundamentally . Your software is no longer deployed once locally on your device. If no NUCs are online or set up, **one instance of your selected program is started for each vehicle that is online** at that time, so make sure that your software can deal with getting one vehicle ID each and that the started programs can communicate with each other (if necessary). The same also holds if NUCs are set up - your software is started on the NUCs (HLCs). (Your system must have installed the full setup). Each NUC takes care of one vehicle only. If more vehicles than NUCs are online, the remaining programs will run locally on your PC.

Requirements

Setting up the NUCs correctly

The NUCs need to be set up. They also need an automatically starting task running, which looks for new versions of the required files. It is called crontab task and relies on lab_autostart.bash.

> **Info on lab_autostart.bash**
>
> This file can also be found in the software repository (`./high_level_controller/autostart`). It is the only file that currently is **not automatically updated** on the NUC. Thus, if you change it, you need to re-upload it on the devices for the guest user. The path is described here .
>
> The script is responsible for setting up the environment on the NUCs (e.g. downloading current versions of the cpm library, middleware etc.) so that programs can be deployed on the machines similarly to local deployment.

Provide all required packages

Before turning the NUCs on

1. Start the main PC (IP: 192.168.1.249)
2. Build the lab software (or make sure your desired version was the most recent build)

The *middleware*, *cpm library* and *autostart* software must be built - the build script puts data required for the software to work on the NUCs in folders that allow the NUCs to download it from the Apache Server (`/var/www/html/nuc`).

Within the folder for the autostart software (located in `~/software/high_level_controller/autostart`), you also find the script `create_nuc_package.bash`. Call this script as well do create the final part of the NUC package (for matlab scripts).

Check again if all four scripts can be found within `var/www/html/nuc` and if they are up to date. Only then you should start the NUCs.

> **Reload after changes**
>
> The HLCs use a startup script that takes the cpm library, Middleware etc. from an Apache server located on the Lab's main PC. The according files are updated every single time that these components are built. If you changed them during the lab run / if they are missing, please rebuild them and restart the NUCs/HLCs, so that they get the new software.
>
> This decision was made to reduce upload traffic when files are uploaded to the HLCs during a lab run.

Make sure that all your files are uploaded correctly

Your folder structure must match the recommended folder structure for both the cpm library and the software repository. If your script/program relies on this data, then you should always refer to other programs using paths relative from ~ (so do not use /home/username explicitly). Cpm library and middleware can be found in the same folders on the NUCs. Required environment variables, e.g. for DDS, are set by default, so you do not need to take care of that.

All required data must lie in the same folder as your executable or script. The whole folder containing it gets uploaded to each NUC. If you depend on other data (except for the Middleware or cpm library, which are uploaded separately), put it in that folder, or your program might not work correctly.

> Relative script paths are currently **not supported**. Also, please do not enter folders that may contain a lot of data (e.g. "./"), because all that data is then uploaded to all NUCs (which, although it gets deleted after they are restarted, might not be a good idea)

Usage of MATLAB

If you use Matlab you need to use the **init** script. Load required data (e.g. XML configuration files), setup the reader and writer etc. Alternatively, use a script that is inspired by the init script. QOS_LOCAL_COMMUNICATION.xml is always located in `~/software/middleware/build`, and precompiled DDS IDL Matlab files can always be found in `~/software/cpm_lib/dds_idl_matlab`. Other files cannot be expected to be found on the NUC, unless you put them in the same folder as your own script.

MATLAB example for init script

```
clc
script_directoy = fileparts([mfilename('fullpath') '.m']);
cd(script_directoy)

% Initialize data readers/writers...
init_script_path = fullfile('../', '/init_script.m');
assert(isfile(init_script_path), 'Missing file "%s".', init_script_path);
addpath(fileparts(init_script_path));
[matlabParticipant, stateReader, trajectoryWriter, systemTriggerReader, readyStatusWriter, trigger_stop] =
init_script(matlabDomainID);
cd(script_directoy)
```

General information

Mapping of vehicle IDs

The available NUCs are mapped to vehicles in an ascending order (NUC 11 vehicle 1, NUC 15 vehicle 4, ...).

Each NUC is always responsible for one vehicle only.

Using IDL-Datatypes and the cpm library

The location of the cpm library is always set using LD_LIBRARY_PATH in the local terminal - you do not need to take care of this in your program, as long as you have linked the library properly in your build script.

The Bash Folder

There is a folder called *bash* in the source code of the LCC. It contains scripts that are relevant for (remote) deployment of scripts, uploading scripts etc. You do not need to understand these scripts, but they are also used to load environment variables. You might encounter an error if you e.g. use another version of DDS than specified e.g. in *environment_variables.bash*. Your software might, for example, simply not start if the required variables could not be set. Make sure that you use the same package versions or change the entries accordingly.

Specifying the matlab version for HLCs on the NUCs

In the aforementioned *bash* folder there is a file *lab_control_center/bash/tmux_script.bash* in which the path to the Matlab version that the NUC uses to execute the HLC scripts can be specified. For example to select MATLAB 2022a:

```
...
#Evaluate the matlab script
    SCRIPT_NAME="${SCRIPT_NAME%%.*}" #remove .m
    /usr/local/MATLAB/R2022a/bin/matlab -logfile matlab.log -sd "${PATH_TO_SCRIPT}" -batch "${SCRIPT_NAME}
(${SCRIPT_ARGS})"
...
```

or MATLAB 2020a:

```
/opt/MATLAB/R2020a/bin/matlab -logfile matlab.log -sd "${PATH_TO_SCRIPT}" -batch "${SCRIPT_NAME}
(${SCRIPT_ARGS})"
```

Integrated error checking

IDL-Datatype for error checking

The following data type is used to check for NUC-crashes.

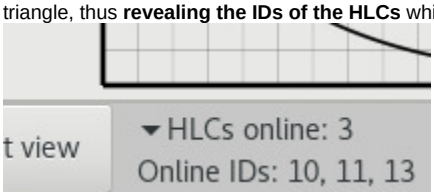
https://github.com/embedded-software-laboratory/cpm_lab/blob/master/cpm_lib/dds_idl/HLCHello.idl

Explanation: Hello messages are sent regularly by each NUC. Missing messages or unexpected entries can be used to detect errors.

- **source_id**: This field can be used to obtain the ID of the NUC that sent the message
- **script_running**: True if a tmux session for the script is currently running. These sessions are started on the NUC when Deploy Remote is used, after the upload has finished. Usually, when a program crashes, the tmux session is stopped as well, thus this value can be used as an indicator to check if the script crashed.
- **middleware_running**: True if a tmux session for the middleware is running. Similar to script_running.

Online-indicators for NUCs

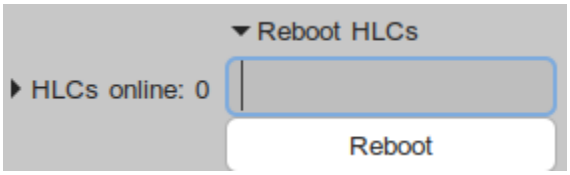
The UI indicates how many NUCs are online. The UI shows you how many NUCs are currently online (matching *might take up to a minute* after the NUCs have been booted and your program has started.) This can be useful for debugging or to find out on how many NUCs you can distributedly deploy your program before, for the rest of your selected vehicles, the computation for those takes place locally. You can expand the section by clicking on the little triangle, thus **revealing the IDs of the HLCs** which are currently online:



If the entry stays empty, you need to check what went wrong. Potential errors are:

- You are using the wrong domain ID (DDS) for the LCC. If LCC and NUCs are not within the same DDS domain, they cannot discover each other. Please check which ID you need to use.
- Packages are missing on the NUCs. In this case, upload packages are missing on the main PC or the NUCs have not been set up correctly (e.g. the autostart task, see requirements)
- The NUCs or your PC are not connected to the same network
- You are not using the main Lab PC (which is recommended for distributed deployment)
- The NUCs crashed

The following section *Reboot HLCs* allows you to **reboot HLCs of your choice**. You again need to click on the little triangle. An input will appear:



You can find out what to enter here if you hover over the input. You can either enter a comma-separated list that the LCC is supposed to reboot (you can check if that worked by examining the online HLCs simultaneously), or you can enter a '*' to tell the LCC to reboot all currently connected NUCs.

Troubleshooting

The NUCs do not show up as online in the LCC despite running?

- Set GCC 10 and G++10 as default on the main PC and all NUCs using `update-alternatives`. It is important that the same version is used on all devices (main PC + NUCs).
- Rebuild the lab software
  - Run `git clean -xdf` and rebuild the lab software on the main PC with `./build_all.bash`
  - Start the LCC
  - Restart NUCs
- Update NUC scripts
  - ssh onto the NUCs, run the "lab_update" script in the "autostart" folder. The "autostart" and "lab_update" scripts should not be run as root, but as "guest" user.
  - The script may freeze, but the NUCs should still appear in the LCC. Do not terminate the "lab_update" script, but restart the NUCs (via the LCC or via the ssh connection).
- If problems occur when running the "autostart" scripts on the NUCs, check and adjust the file/folder permissions. If necessary, you can also examine the scripts by inserting debug output.
- Copying to the NUCs requires free memory on the main PC. The more NUCs are used, the more free memory is required. In general, problems are to be expected with less than 5GB of free memory.

The upload of your HLC-Files crashes due to denied permission?

- The ssh-connection is established using an ssh-key without a passphrase. This ssh-key belongs to the user `cpm` on the Main-PC. Thus switch to this user or adapt the scripts in `lab_control_center/bash` temporarily to establish the ssh-connection using the `guest` users password.

Crash checks for your programs

As soon as you actually deploy your programs, and after the scripts have been uploaded, the LCC also starts checking if their corresponding tmux sessions (middleware and script) are still running.

Errors are reported in the UI if that it not the case anymore (e.g. with a popup similar to local crash checks, and in the monitoring window for each vehicle).