ns-3 with CMake

Gabriel Ferreira

University of Brasília

September 30, 2021

My goals with CMake

IDE support

1. Easier debugging and profiling
2. Auto-completion
3. Macro expansion (showing what the actual code looks like)
4. Visually show errors, warnings, unused includes, etc

Unsupported/incomplete features

1. Bake integration
2. NSC examples
3. Planetlab examples
4. Python bindings

   The code is there, but apiscan and generation fail for some modules

What is missing?

1. Implement remaining features

   I can do that!

2. Stylistic and workflow related refactoring

   I need maintainers feedback

3. Update ns-3 official docs with instructions

Current workflow

1. Create a new module
2. Configure CMake
3. Call the generated build system to build (e.g. ninja)
4. Call binaries directly or via the IDE
5. Debug via the IDE

Current workflow

Create a new module

1. Create a new folder with a CMakeLists.txt file. Set library name, list of source files and add the macro at the end.

wscript 1.46 KB

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-

def build(bld):

    module = bld.create_ns3_module('antenna', ['core'])

    module.source = [
        'model/angles.cc',
        'model/antenna-model.cc',
        'model/isotropic-antenna-model.cc',
        'model/cosine-antenna-model.cc',
        'model/parabolic-antenna-model.cc',
        'model/three-gpp-antenna-model.cc',
        'model/phased-array-model.cc',
        'model/uniform-planar-array.cc',
        ]

    module_test = bld.create_ns3_module_test_library('antenna')
    module_test.source = [
        'test/test-angles.cc',
        'test/test-degrees-radians.cc',
        'test/test-isotropic-antenna.cc',
        'test/test-cosine-antenna.cc',
        'test/test-parabolic-antenna.cc',
        'test/test-uniform-planar-array.cc',
        ]

    # Tests encapsulating example programs should be listed here
    if (bld.env['ENABLE_EXAMPLES']):
        module_test.source.extend([
        #   'test/antenna-examples-test-suite.cc',
            ])

    headers = bld(features='ns3header')
    headers.module = 'antenna'
    headers.source = [
        'model/angles.h',
        'model/antenna-model.h',
        'model/isotropic-antenna-model.h',
        'model/cosine-antenna-model.h',
        'model/parabolic-antenna-model.h',
        'model/three-gpp-antenna-model.h',
        'model/phased-array-model.h',
        'model/uniform-planar-array.h',
        ]
```

CMakeLists.txt 933 Bytes

```
set(name antenna)

set(source_files
    model/angles.cc
    model/antenna-model.cc
    model/cosine-antenna-model.cc
    model/isotropic-antenna-model.cc
    model/parabolic-antenna-model.cc
    model/phased-array-model.cc
    model/three-gpp-antenna-model.cc
    model/uniform-planar-array.cc
)

set(header_files
    model/angles.h
    model/antenna-model.h
    model/cosine-antenna-model.h
    model/isotropic-antenna-model.h
    model/parabolic-antenna-model.h
    model/phased-array-model.h
    model/three-gpp-antenna-model.h
    model/uniform-planar-array.h
)

# link to dependencies
set(libraries_to_link ${libcore})

set(test_sources test/test-angles.cc test/test-degrees-radians.cc test/test-isotropic-antenna.cc
                 test/test-cosine-antenna.cc test/test-parabolic-antenna.cc test/test-uniform-planar-array.cc
)

build_lib("${name}" "${source_files}" "${header_files}" "${libraries_to_link}" "${test_sources}")
```

UnB

Current workflow

Configure CMake

1. This is required every time the CMake is changed

   e.g. to include new source files or settings were changed

2. Previous options are preserved in the CMake cache

   a.k.a. no need to pass all the options again when reconfiguring

3. Can be done either manually via terminal or GUI, or automatically via a config file (setting the flag values)

   e.g. cmake –DCMAKE_BUILD_TYPE=debug –DNS3_EXAMPLES=ON –DNS3_TESTS=ON -G Ninja ..

Current workflow

Call the generated build system to build

1. Call generated build system to build everything (or a specific target and its dependencies)

   e.g. ninja (target)

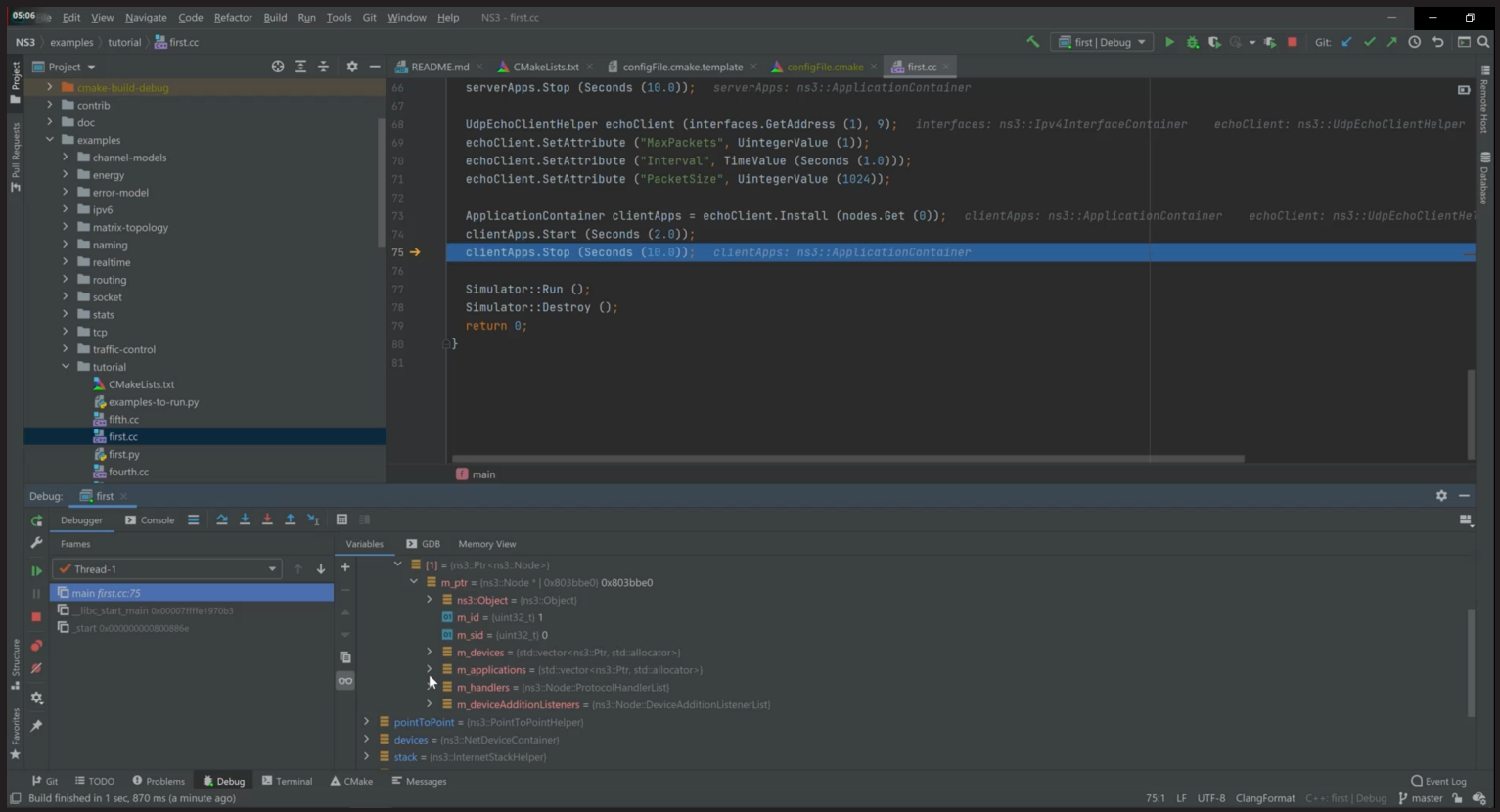
Current workflow

Call binaries directly or via the IDE

1. Works nicely on platforms that support RPATH, which is enabled by default (e.g. Linux and MacOS)
2. Windows requires ns-3-dev/build in the PATH, which can be done automatically
3. Something like waf –run, –gdb, –valgrind, –command-template, –pyrun, –visualize would require a wrapper script

Current workflow

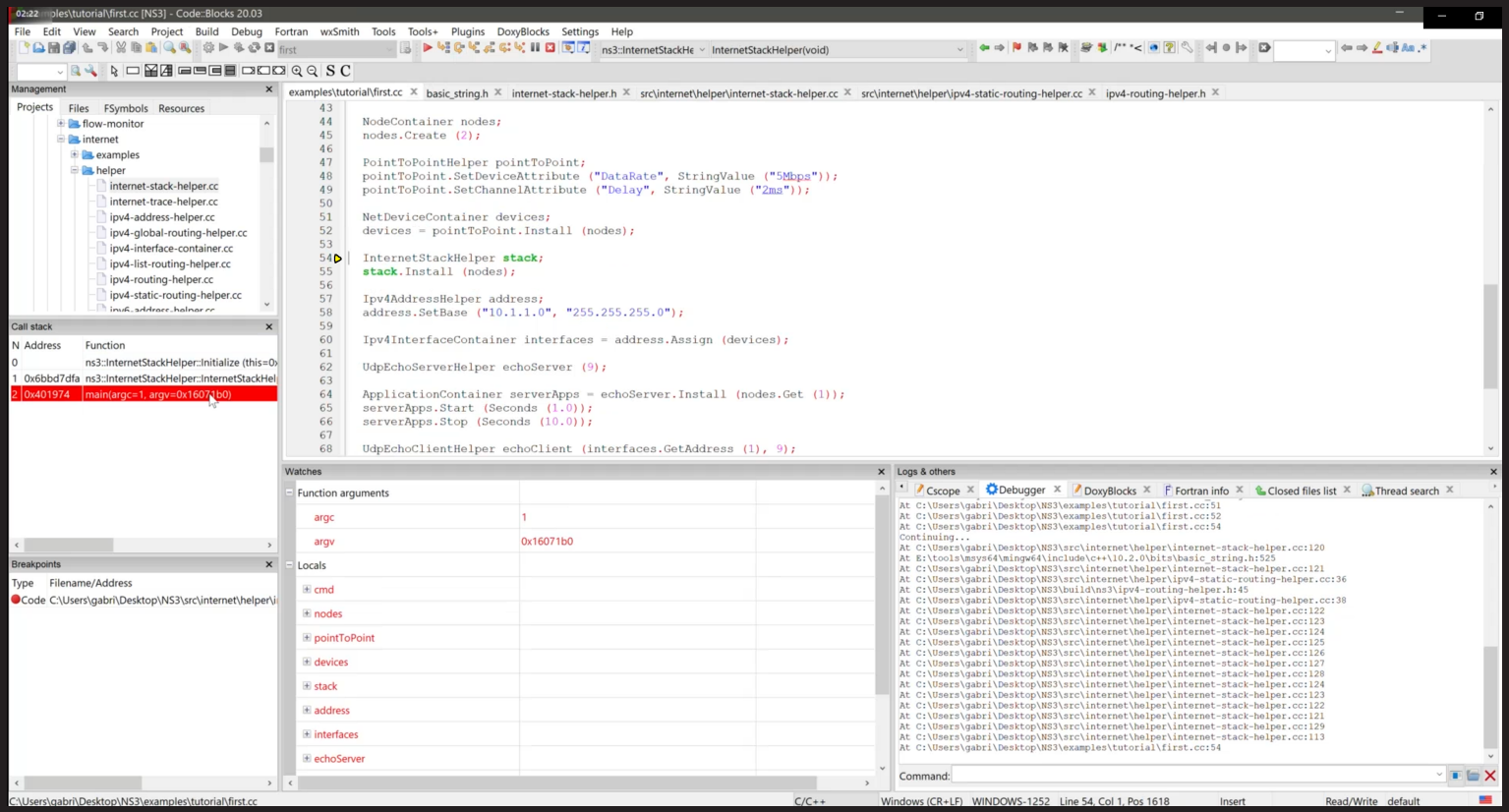
Debugging via the IDE: CLion



UnB

Current workflow

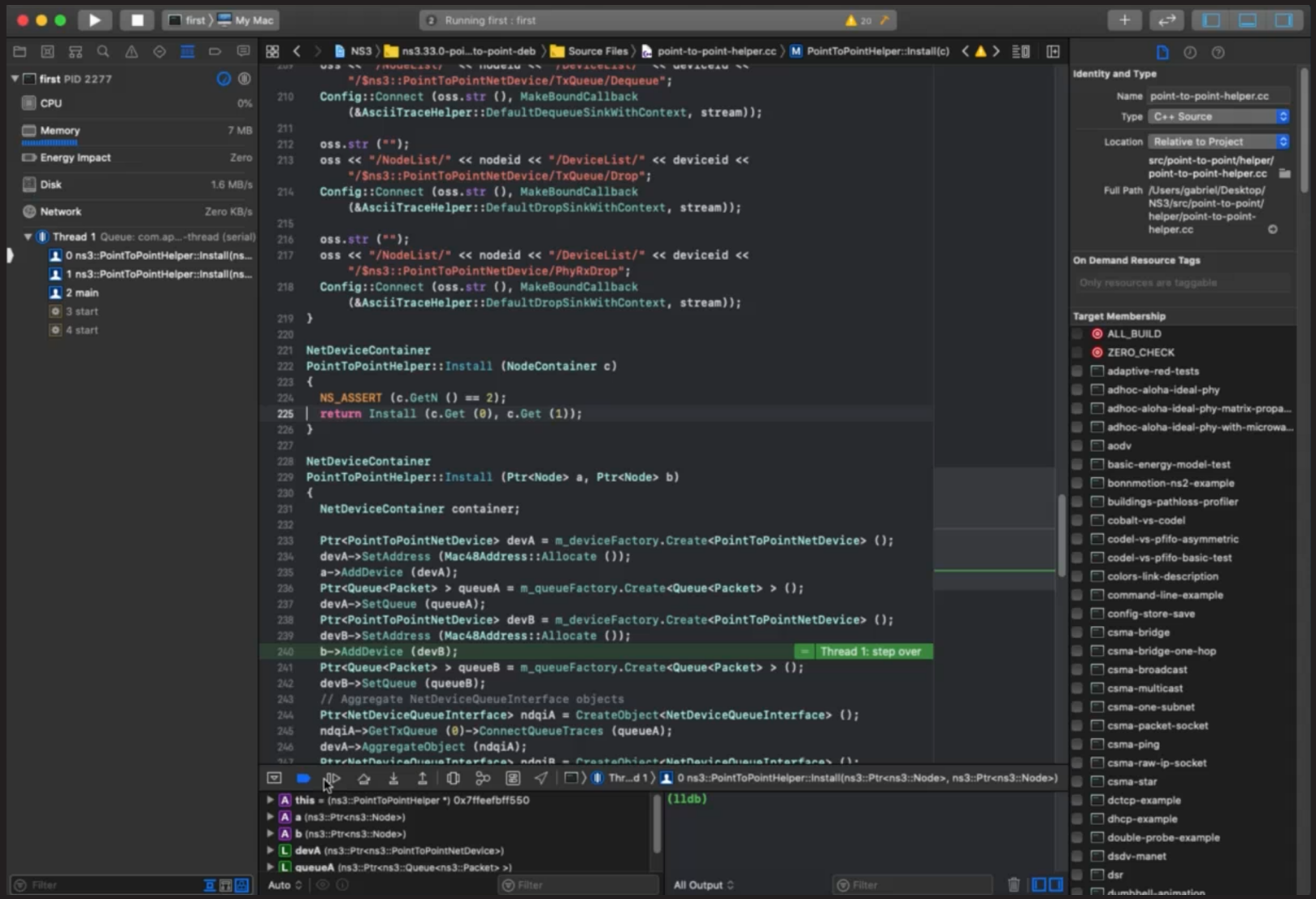
Debugging via the IDE: Code::Blocks



UnB

Current workflow

Debugging via the IDE: XCode



UnB

Where is the code and instructions?

The instructions are in the MR 460:

https://gitlab.com/nsnam/ns-3-dev/-/merge_requests/460

The sources are in the branch for the MR 460:

https://gitlab.com/Gabrielcarvfer/ns-3-dev/-/tree/buildsystem-cmake

Questions? Concerns? Suggestions?

You can find me on Zulip or via the email
gabrielcarvfer@gmail.com

Thank you for your time