

ns-3 with CMake

Gabriel Ferreira

University of Brasília

September 29, 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

1 # -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-
2
3 def build(bld):
4
5     module = bld.create_ns3_module('antenna', ['core'])
6
7     module.source = [
8         'model/angles.cc',
9         'model/antenna-model.cc',
10        'model/isotropic-antenna-model.cc',
11        'model/cosine-antenna-model.cc',
12        'model/parabolic-antenna-model.cc',
13        'model/three-gpp-antenna-model.cc',
14        'model/phased-array-model.cc',
15        'model/uniform-planar-array.cc',
16    ]
17
18     module.test = bld.create_ns3_module_test_library('antenna')
19     module.test.source = [
20         'test/test-angles.cc',
21         'test/test-degrees-radians.cc',
22         'test/test-isotropic-antenna.cc',
23         'test/test-cosine-antenna.cc',
24         'test/test-parabolic-antenna.cc',
25         'test/test-uniform-planar-array.cc',
26     ]
27
28     # Tests encapsulating example programs should be listed here
29     if (old.env['ENABLE_EXAMPLES']):
30         module_test.source.extend([
31             '# test/antenna-examples-test-suite.cc',
32         ])
33
34     headers = bld(features='ns3Header')
35     headers.module = 'antenna'
36     headers.source = [
37         'model/angles.h',
38         'model/antenna-model.h',
39         'model/isotropic-antenna-model.h',
40         'model/cosine-antenna-model.h',
41         'model/parabolic-antenna-model.h',
42         'model/three-gpp-antenna-model.h',
43         'model/phased-array-model.h',
44         'model/uniform-planar-array.h',
45     ]
```

```
CMakeLists.txt 101 lines

1 set(name antenna)
2
3 set(source_files
4     model/angles.cc
5     model/antenna-model.cc
6     model/isotropic-antenna-model.cc
7     model/cosine-antenna-model.cc
8     model/parabolic-antenna-model.cc
9     model/three-gpp-antenna-model.cc
10    model/phased-array-model.cc
11    model/uniform-planar-array.cc
12    )
13
14 set(header_files
15     model/angles.h
16     model/antenna-model.h
17     model/isotropic-antenna-model.h
18     model/cosine-antenna-model.h
19     model/parabolic-antenna-model.h
20     model/phased-array-model.h
21     model/three-gpp-antenna-model.h
22     model/uniform-planar-array.h
23    )
24
25 # Link to dependencies
26 set(libraries_to_link ${libraries})
27
28 set(test_sources test/test-angles.cc test/test-degrees-radians.cc test/test-isotropic-antenna.cc
29                 test/test-cosine-antenna.cc test/test-parabolic-antenna.cc test/test-uniform-planar-array.cc
30                )
31
32 build_libr("name") "${source_files}" "${header_files}" "${libraries_to_link}" "${test_sources}"
```

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

The screenshot shows the NS3 IDE interface. The top part displays the code for `main.cc`, which includes the configuration of an application container and its start/stop logic. The bottom part shows the `Variables` view, which is currently displaying the `Memory View`. It lists several pointers and their corresponding memory addresses, such as `m_pNet`, `m_pDevice`, `m_pApplication`, `m_pScheduler`, `m_pForwarder`, `m_pDeviceList`, `m_pDeviceContainer`, and `m_stack`.

```
serverApps.Stop (Seconds (10.0)); serverApps = ns3::ApplicationContainer();

UpdPchClientHelper echoClient (interfaces.GetAddress (1));
echoClient.SetAttribute ("MaxPackets", Uinteger32 (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PowerSize", Uinteger32 (1000));

ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps = ns3::ApplicationContainer (echoClient);
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
clientApps = ns3::ApplicationContainer ();

Simulator::iRun ();
Simulator::iDestroy ();
return 0;
}

NS3 IDE Project Explorer:
```

- create-build.sh
- ctrl
- doc
- examples
- examples-for-run.py
- glue
- error-model
- gavl
- matrix-topology
- net
- ns3
- ns3-ns2
- ns3-ns3
- ns3-ns3c
- ns3-ns3d
- ns3-socket
- ns3-stats
- ns3-sip
- ns3-traffic-control
- tutorial
 - Makefile
 - examples-for-run.py
 - glue.cc
 - main.cc
 - net.cc
 - empty
 - fourcc

Debug: first

Debugger | Console | Variables | Threads | Memory View

Variables View

- [1] m_pNet [0x1000000000000000]
- > ns3::Object [0x1000000000000000]
- > m_pDevice [0x1000000000000000]
- > m_pDevice [0x1000000000000000]
- > m_pApplication [0x1000000000000000]
- > m_pScheduler [0x1000000000000000]
- > m_pForwarder [0x1000000000000000]
- > m_pDeviceList [0x1000000000000000]
- > m_pDeviceContainer [0x1000000000000000]
- > m_stack [0x1000000000000000]



Current workflow

Debugging via the IDE: Code::Blocks

The screenshot shows the Code::Blocks IDE interface during a debugging session. The main window displays the source code of a C++ file named `basic_ipng.cc`. The code implements various network helper functions, including `InternetStackHelper`, `Ipv4AddressHelper`, and `UdpEchoClientHelper`.

The **Call stack** window shows the current call stack, which includes frames for `InternetStackHelper::Initialize`, `Ipv4AddressHelper::Initialize`, and `UdpEchoClientHelper::Initialize`.

The **Breakpoints** window lists a single breakpoint at line 63 of the `basic_ipng.cc` file.

The **Registers** window shows the CPU registers, with the `arg1` register containing the value `0x1907110`.

The **Locals** window shows local variables for the current function frame, including pointers to `NodeContainer`, `InternetStackHelper`, `Ipv4AddressHelper`, and `UdpEchoClientHelper`.

The **Log & others** window displays the output of the `cout` statements from the code, showing the configuration of the `stack` object.

The status bar at the bottom indicates the file is `I/C++`, the build target is `Windows (C++)`, the operating system is `WINDOWS-1252`, the line is `Line 54, Col 1`, and the column is `Pos 1618`.



Current workflow

Debugging via the IDE: XCode



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?

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

Thank you for your time