

NS-3 Analýza simulátora

Simulátor NS-3 je spustiteľný na rôznych operačných systémoch - Windows, MacOS a linux na rôznych distribúciach s rôznymi metódami inštalácie. My v našom projekte používame linux, takže ďalšie kapitoly budú opisovať používané simulátora na tejto platforme.

Inštalácia

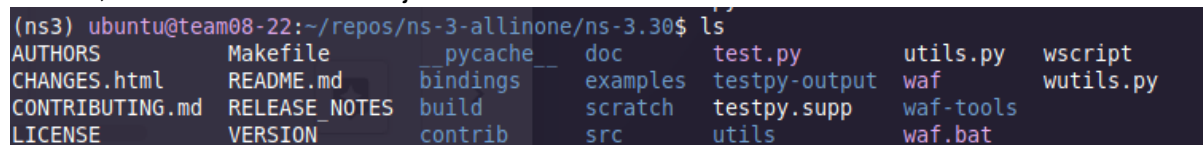
Je prehľadne zdokumentovaná v ns-3 dokumentácii [\[1\]](#). Použili sme manuálnu inštaláciu cez git.

1. Naklonujeme si repozitár
`git clone https://gitlab.com/nsnam/ns-3-allinone.git`
2. Po naklonovaní sa v priečinku ns-3-allinone zobrazí python script, pomocou ktorého vieme stiahnuť hocikakú verziu simulátora. My zvolíme verziu 3.30 a nainštalujeme ju príkazom `./download.py -n ns-3.30`
3. Prejdeme do adresára ns-3.30 príkazom `cd ./ns-3.30`, v ktorom už potrebujeme len zostaviť simulátor z jeho zdrojov. Je na to script `build.py`, ktorý spustíme príkazom `./build.py`

Ns-3 je následne nainštalované. Na jeho používanie ho ešte treba nakonfigurovať, čomu sa budeme venovať v nasledujúcej kapitole.

Konfigurácia NS-3

Po nainštalovaní nami zvolenej verzie ns-3 v priečinku ostane veľa rôznych adresárov a súborov, zobrazené na nasledujúcom obrázku



```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ls
AUTHORS      Makefile      __pycache__  doc           test.py       utils.py      wscript
CHANGES.html README.md     bindings     examples     testpy-output waf           wutils.py
CONTRIBUTING.md RELEASE_NOTES build         scratch      testpy.supp  waf-tools
LICENSE      VERSION      contrib      src          utils        waf.bat
```

Našťastie, veľa z nich nepotrebujeme vôbec riešiť. Tie podstatné si prejdeme v nasledujúcich podkapitolách

Waf je systém založený v pythone na zjednodušenie operácii s ns-3 simulátorom. Poskytuje jednoduchý systém v ktorom sa robí všetko súvisiace s ns-3 simulátorom, hlavne konfigurácia modulov ns-3, kompilácia s spúšťanie scenárov

- a) Konfigurácia modulov sa robí pomocou príkazu `./waf configure`. Tento príkaz musí byť spustený pred každou zmenou konfigurácie, inak waf bude používať tú starú. Po inštalácii je waf nenakonfigurovaný, takže to treba spustiť aspoň raz.

- b) Aby sme vedeli, či simulátor správne funguje musíme nechať zbehnúť testy. Na to najprv treba nakonfigurovať waf jednoduchým príkazom `./waf configure --enable-tests`. Následne môžeme spustiť testy príkazom `./test.py`. Testy sa začnú postupne vykonávať, ale keďže ich je veľa tak to chvíľu trvá.

```
PASS: TestSuite routing-aodv-loopback
PASS: TestSuite routing-aodv-regression
PASS: TestSuite lte-frequency-reuse
256 of 259 tests passed (256 passed, 3 skipped, 0 failed, 0 crashed, 0 valgrind errors)
List of SKIPPed tests:
  ns3-tcp-cwnd (requires NSC)
  ns3-tcp-interopability (requires NSC)
  nsc-tcp-loss (requires NSC)
```

Po zbehnutí testov je ns-3 pripravené na vytváranie a spúšťanie scenárov.

Spúšťanie NS-3 simulácii

Spúšťanie simulácii je jednoduché. Treba skopírovať zdrojový kód simulácie do podpriechinky `./scratch`. Následne sa simulácia spustí cez waf príkazom `./waf --run <filename>`. V takomto príkaze sa zdrojový kód simulácie hľadá v priečinku `./scratch`. Názov súboru sa nepíše s koncovkou `.cc`, inak ho waf nenájde.

```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ./waf --run scratch/third
Waf: Entering directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
[1947/1999] Compiling scratch/third.cc
[1959/1999] Linking build/scratch/third
Waf: Leaving directory `/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.150s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
```

Výstup simulácie je priamo definovaný v simulačnom scenári. Medzi rôzne výstupy patria napr. `.xls` súbory, `.pcap` súbory alebo logy v textovom formáte. Tento výstup hlbšie preberieme v kapitole o simulácii.

NS-3 a python

Väčšina simulátorov je napísaná v programe `c++`. NS-3 dnes ponúka aj alternatívu, a to je písanie scenárov v jazyku python. Takéto scenáre majú rýchlejšie prvotné spustenie, nakoľko sa nemusia kompilovať a sú jednoduchšie na písanie, lebo python je jednoduchší jazyk. Na druhú stranu k nim je menej dokumentácie.

Na spustenie ns-3 simulácie v pythone si treba najprv pripraviť prostredie. Od verzie pythonu 3.9 je jedna používaná funkcia už zastaralá a vyhadzuje varovanie. Bohužiaľ waf pri konfigurácii varovania považuje za error a preto to spoľahlivo funguje len s pythonom verzie `<=3.9`.

```
[1853/1997] Compiling build/src/network/bindings/ns3module.cc
src/core/bindings/ns3module.cc: In function 'PyObject* PyInit_core()':
src/core/bindings/ns3module.cc:51659:23: error: 'void PyEval_InitThreads()' is deprecated [-Werror=deprecated-declarations]
51659 |     PyEval_InitThreads();
      |     ~~~~~^
In file included from /usr/include/python3.10/Python.h:130,
               from src/core/bindings/ns3module.h:3,
               from src/core/bindings/ns3module.cc:1:
/usr/include/python3.10/ceval.h:122:37: note: declared here
122 | Py_DEPRECATED(3.9) PyAPI_FUNC(void) PyEval_InitThreads(void);
      | ~~~~~^
cc1plus: all warnings being treated as errors
```

Zvolili sme teda verziu 3.8. Pridáme si do apt repozitárov repozitár deadsnakes príkazom

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

Tento repozitár obsahuje staršie verzie pythonu. Ďalej nainštalujeme našu verziu cez príkaz (aj dev je potrebná, inak to nezbehne)

```
sudo apt install python3.8 python3.8-dev
```

Do pythonu potrebujeme nainštalovať venv modul na spracovanie virtuálnych prostredí príkazom

```
sudo apt install python3.8-venv
```

Následne vytvoríme virtuálne prostredie príkazom `python3 -m pip venv ns3` a aktivujeme ho príkazom `source <path/na/venv>/bin/activate`

NS3 pre spúšťanie potrebuje balíček pybindgen, ktorý nainštalujeme cez pip príkazom `pip install pybindgen`

Ako posledná vec je potrebné nakonfigurovať waf, aby vedela spustiť scenáre definované v pythone príkazom `python ./waf configure`. Tento proces prvýkrát trvá dlhšie, lebo kompiluje potrebné moduly.

Prostredie je teraz nakonfigurované a môžeme spustiť scenár príkazom `./waf --pyrun`

`./path/na/scenar.py`.

```
(ns3) ubuntu@team08-22:~/repos/ns-3-allinone/ns-3.30$ ./waf --pyrun ./examples/tutorial/third.py
Waf: Entering directory '/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Waf: Leaving directory '/home/ubuntu/repos/ns-3-allinone/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.844s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01796s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01796s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
```

Simulácia

Základom simulácie sú uzly (node). Je to abstrakcia predstavujúca nejaký počítač/stroj/zariadenie. Tieto uzly sa následne špecifikujú. V scenároch sa často stáva, že viacero uzlov môže mať rovnakú konfiguráciu. Preto vieme viacero uzlov vložiť do jedného kontajneru a "nainštalovať" im konfiguráciu naraz. Slúži na to prvok `NodeContainer`. Takáto konfigurácia vyzerá nasledovne:

```
NodeContainer p2pNodes;
p2pNodes.Create (2);
PointToPointHelper pointToPoint;
```

```

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

```

Tu vytvárame 2 uzly a point-to-point konfiguráciu s vlastnými atribútami a následne ju inštalujeme na vytvorený kontajner.

Keďže vo VANET sieťach sa budeme zaoberať bezdrôtovými sieťami, tak sa bližšie pozrieme na simuláciu WIFI zariadení v ns-3. V základnej wifi konfigurácii sú 2 typy wifi uzlov. AP (access point) uzol a STA(station node). AP uzly sú stacionárne a STA uzly sa vedia pohybovať.

```

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

```

Následne je potrebné nakonfigurovať jednotlivé vrstvy - aspoň fyzickú a mac vrstvu. Pri fyzickej sa nastavuje aký kanál bude bezdrôtová komunikácia používať. NS-3 poskytuje aj základný kanál, ktorý je možno použiť.

```

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

```

Na mac vrstve sa okrem iného dá nastaviť SSID siete, typ wifi uzla - AP alebo STA a či má aktívne vyhľadávať nové siete v jeho rozsahu.

```

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
             "ActiveProbing", BooleanValue (false));

```

Pri bezdrôtových sieťach je taktiež potrebné zadať fyzickú polohu uzlu, keďže vzdialenosti majú vplyv napríklad na signál. NS-3 na to poskytuje triedu MobilityHelper, vďaka ktorej vieme namapovať uzly na virtuálny 2D priestor, ktorého rozsah vieme špecifikovať.

```

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UIntegerValue (3),
                               "LayoutType", StringValue
("RowFirst"));

```

Správanie jednotlivých uzlov vieme zadať cez mobilitymodel, napríklad aby bol uzol stacionárny, alebo sa náhodne pohyboval v určitom rozsahu

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",  
                           "Bounds", RectangleValue (Rectangle (-50,  
50, -50, 50)));  
mobility.Install (wifiStaNodes);
```

Ostatné typy bezdrôtových sietí budú analyzované v nasledujúcich šprintoch.

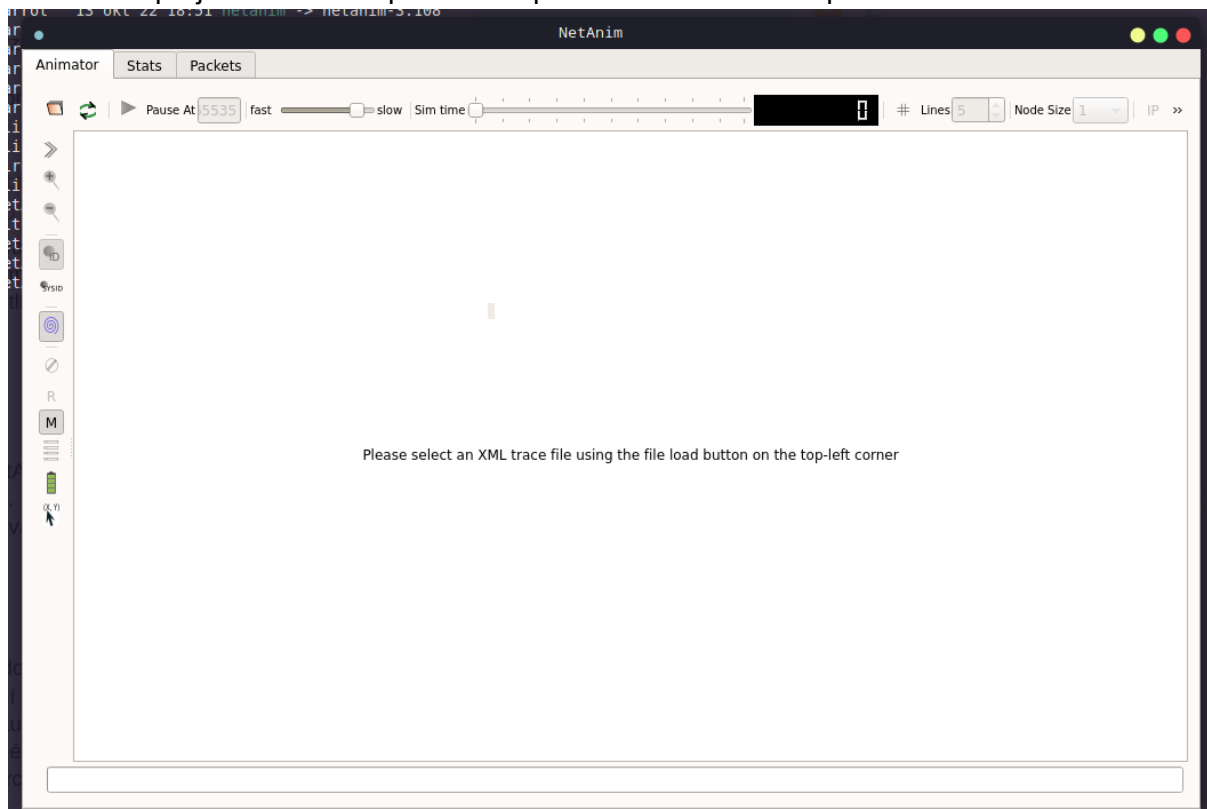
Aby bol scenár aspoň trochu modulárny ns-3 umožňuje konfiguráciu scenára cez command line argumenty. Slúži na to trieda CommandLine.

```
CommandLine cmd;  
cmd.AddValue ("verbose", "Tell echo applications to log", verbose);  
cmd.AddValue ("tracing", "Save simulation to trace files", tracing);
```

Program sa s argumentami spúšťa cez waf cez syntax `./waf --run "/scratch/scenar --argument=hodnota --argument2=hodnota2"`. Okrem informácii o logovaní zvykne byť týmto spôsobom konfigurovateľný napríklad počet uzlov v jednotlivých kontajneroch.

Vizualizácia výstupov simulácie

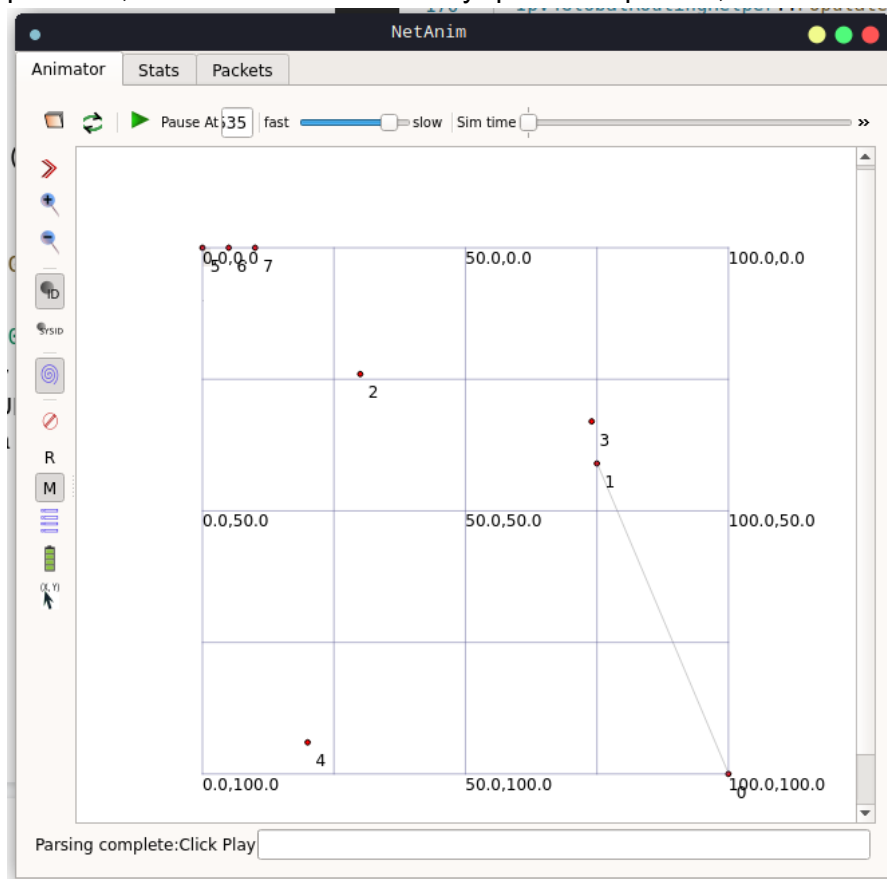
Pri inštalácii ns-3 dostaneme zdrojové kódy programu NetAnim, ktorý umožňuje vizualizáciu takejto simulácie. Zdrojové kódy treba najprv skompilovať. Je to nástroj s GUI, takže ho treba spúšťať na systéme s grafickým rozhraním, nie len terminálom. Keďže je aplikácia napísaná pomocou Qt knižnice, najjednoduchší spôsob je nainštalovať QtCreator, otvoriť v ňom súbor projektu NetAnim.pro a skompilovať ho cez QMake priamo tam.



Po spustení je vidieť, že NetAnim si pýta XML trace súbor. Na vytvorenie takéhoto súboru treba rozšíriť ns-3 simuláciu o pár riadkov kódu

```
AnimationInterface anim("third.xml");  
for(uint32_t i=0; i<=nCsma; i++) {  
    anim.SetConstantPosition(csmaNodes.Get(i), 10.0, 10.0*i*2);  
}  
for(uint32_t i=0; i<nWifi; i++ ) {  
    anim.SetConstantPosition(wifiStaNodes.Get(i), 10.0*2*i, 10.0);  
}  
anim.SetConstantPosition(wifiApNode.Get(0), 100.0, 100.0);
```

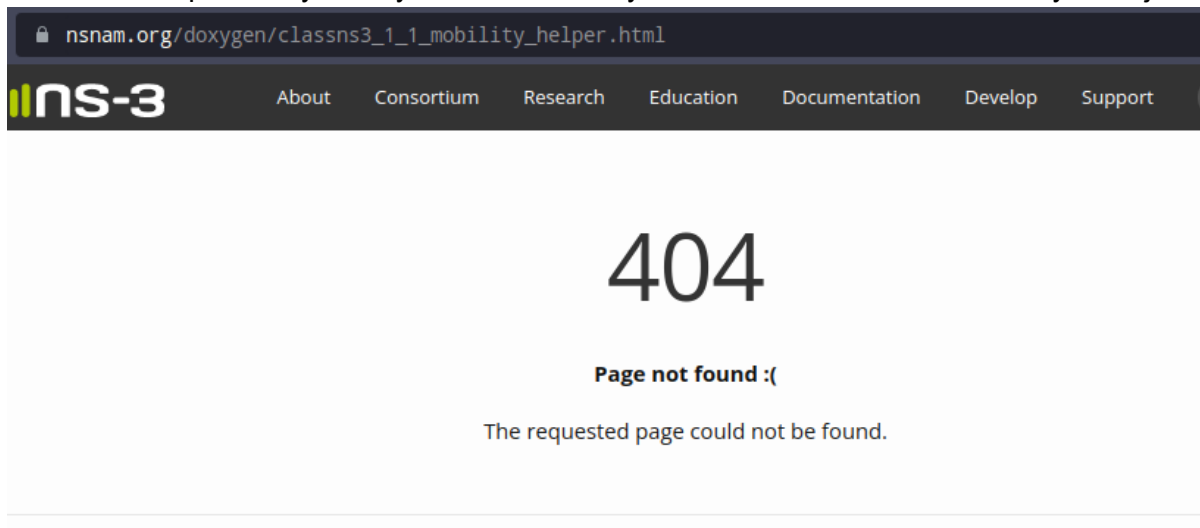
Hlavnou súčasťou kódu je objekt AnimationInterface, ktorý vytvorí/prepíše súbor v konštrukторе a bude do neho vkladať logy. Keďže ide o GUI tool ktorý vizualizuje uzly v 2D priestore, tak každému uzlu musí byť pridelená pozícia, na ktorú sa vykreslí.



Počiatočná pozícia sa vykreslí a simulácia sa dá simulovať pomocou zeleného tlačidla v toolbare programu. V tabe Stats sa dajú pozrieť štatistiky a v tabe Packets odoslané pakety. Program je však labilný takže treba s ním narábať opatrne, inak segfaultuje.

Nedostatky

Bohužiaľ, ns3 je veľmi riedko zdokumentované. Kvalitná dokumentácia končí pri inštalácii, konfigurácii a spustení prerobených simulácií. Pri googlení je polovica zdrojov mŕtva a hádže 404 (obrázok) alebo je dokumentácia limitovaná na ukážku používania bez žiadneho opisu čo/prečo/ako. Najväčší zdroj informácií sú teda predrobené scenáre v priečinku examples a hŕstka videí uploadnutých na youtube od nezávislých tvorcov čo dané scenáre vysvetľujú.



Zdroje

<https://www.nsnam.org/wiki/Installation>