

گزارش تمرین کامپیوتری دوم

مبانی شبکه های بی سیم

دکتر شریعت پناهی

محیا قینی - ۸۱۰۱۹۶۶۱۵

خرداد ۱۴۰۰

مقدمه

در این تمرین قصد داشتیم ضمن آشنایی با شبیه ساز سناریوهای مختلف شبکه ها با نام Network Simulator 3، یک سناریو Handover را به عنوان نمونه پیاده سازی اجرا کنیم. نتیجه اجرا را هم در شبیه ساز گرافیکی NetAnim و هم در visualizer مشاهده کردیم. در ادامه توضیح مختصری بر مراحل کار می دهیم.

نصب و راه اندازی نرم افزار NS3 و NetAnim

در ابتدا بایستی ابزار مورد نیاز را نصب می کردیم. ابزاری که مورد نیاز بود شامل NS3 و NetAnim می شد. مراحل نصب طبق آموزش هایی که در سایت هر کدام از ابزار ها موجود بود، انجام شد. در اینجا جزئیات و مراحل نصب دیگر آورده نمی شود و تنها تعدادی از مراحل مهم نصب بیان می شوند.

● Build

بعد از نصب NS3 بایستی آن را بر روی سیستم عامل خود build می کردیم. در این مرحله module های مورد نیاز در آینده build می شوند. البته تعدادی از آنها به دلیل عدم نصب بودن کتابخانه های مورد نیاز نمی شوند. که در صورت نیاز بایستی ضمن نصب کتابخانه های مورد نیاز دوباره بایستی این مرحله تکرار شود. تصویر زیر این مرحله را نشان می دهد.

```
mahya@mahya-VirtualBox: ~/workspace/ns-allinone-3.33
build.py netanim-3.108 pybindgen-0.21.0.post15+nga587377 util.py
mahya@mahya-VirtualBox:~/workspace/ns-allinone-3.33$ ./build.py --enable-examples --enable-tests
# Build NetAnim
Entering directory 'netanim-3.108'
=> qmake -v
Could not find qmake in the default path
=> qmake-qt5 -v
Could not find qmake-qt5 in the default path
=> qmake-qt4 -v
Could not find qmake-qt4 in the default path
=> qmake NetAnim.pro
Error building NetAnim. Ensure the path to qmake is correct.
Could not find qmake or qmake-qt5 in the default PATH.
Use ./build.py --qmake-path <Path-to-qmake>, if qmake is installed in a non-standard location
Note: Some systems use qmake-qt5 instead of qmake
Skipping NetAnim ....
Leaving directory 'netanim-3.108'
# Building examples (by user request)
# Building tests (by user request)
# Build NS-3
Entering directory './ns-3.33'
=> /usr/bin/python3 waf configure --enable-examples --enable-tests --with-pybindgen ../pybindgen-0.21.0.post15+nga587377
Setting top to : /home/mahya/workspace/ns-allinone-3.33/ns-3.33
Setting out to : /home/mahya/workspace/ns-allinone-3.33/ns-3.33/build
Checking for 'gcc' (C compiler) : /usr/bin/gcc
Checking for cc version : 9.3.0
Checking for 'g++' (C++ compiler) : /usr/bin/g++
Checking for compilation flag -Wl,--soname=foo support : ok
Checking for compilation flag -std=c++11 support : ok
Checking boost includes : headers not found, please provide a --boost-l
includes argument (see help)
Checking boost includes : headers not found, please provide a --boost-l
includes argument (see help)
Checking for program 'python' : /usr/bin/python3
Checking for python version >= 2.3 : 3.8.5
python-config : /usr/bin/python3-config
Asking python-config for pyembed '--cflags --libs --ldflags --embed' flags : yes
Testing pyembed configuration : yes
Asking python-config for pyext '--cflags --libs --ldflags' flags : yes
Testing pyext configuration : yes
Checking for compilation flag -fvisibility=hidden support : ok
Checking for compilation flag -Wno-array-bounds support : ok
Checking for pybindgen location : ../pybindgen-0.21.0.post15+nga587377 (given)
Checking for python module 'pybindgen' : 0.21.0.post15+nga587377
Checking for pybindgen version : 0.21.0.post15+nga587377
Checking for code snippet : yes
Checking for types uint64_t and unsigned long equivalence : yes
```

● اجرای test.py

در این مرحله جهت اطمینان از درستی نصب شدن بایستی فایل تستی که در فولدر برنامه قرار داشت را اجرا می کردیم و در صورتی که خطایی در گزارش نهایی وجود داشت آنها را رفع می کردیم. در زیر تصاویر گزارش نهایی از این مرحله را مشاهده می نمایید.

```
[675/678] 100%: Example src/monitors/examples/wireless-common.py
675 of 678 tests passed (675 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)
```

اگر به تصویر پایین توجه کنید، متوجه می شوید module مربوط به visualiser که در آینده به آن نیاز خواهیم داشت نصب نشده است. بنابراین به صورت دستی کتابخانه های مورد نیاز از قبیل graphviz و غیره را نصب کردم و این module هم enable شد.

```
mahya@mahya-VirtualBox: ~/workspace/ns-allinone-3.33/ns-3.33$ ./test.py
Waf: Entering directory '/home/mahya/workspace/ns-allinone-3.33/ns-3.33/build'
Waf: Leaving directory '/home/mahya/workspace/ns-allinone-3.33/ns-3.33/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.190s)

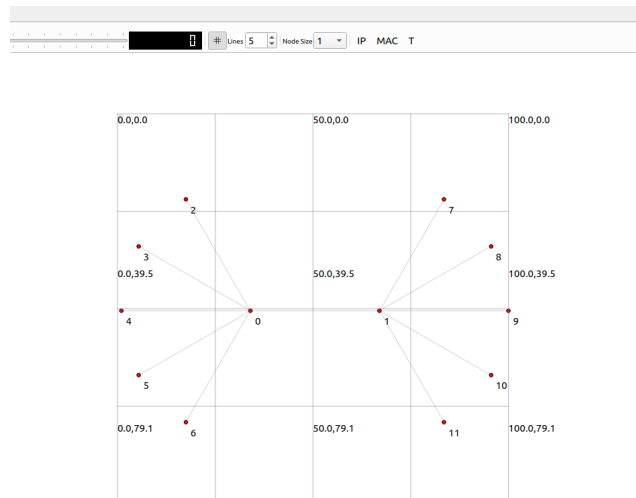
Modules built:
antenna          aodv          applications
bridge           buildings    config-store
core             csma         csma-layout
dsdv             dsr          energy
fd-net-device    flow-monitor internet
internet-apps   lr-wpan      lte
mesh            mobility     netanim
network         nix-vector-routing olsr
point-to-point  point-to-point-layout propagation
sixlowpan       spectrum     stats
tap-bridge      test (no Python) topology-read
traffic-control uan          virtual-net-device
wave            wifi         wimax

Modules not built (see ns-3 tutorial for explanation):
brite           click         dpdk-net-device
mpl             openflow     visualizer

[1/678] PASS: TestSuite attributes
[2/678] PASS: TestSuite attribute-container-test-suite
[3/678] PASS: TestSuite build-profile
[4/678] PASS: TestSuite callback
```

● اجرای فایل xml

برای شبیه ساز گرافیکی NetAnim به فایل xml نیاز داریم. برای این کار کافیت کتابخانه مورد نیاز را در کد include کنیم و AnimationInterface anim(FileName) در کد قرار دهیم. همچنین در wscript هنگام ساخت obj بایستی netanim را هم ذکر کنیم. بعد از این مراحل با اجرای کد فایل xml ساخته می شود. با اجرای آن در NetAnim خروجی گرافیکی را مشاهده می کنیم. در ذیل خروجی گرافیکی کد نمونه dumbbell را مشاهده می کنید.



هدف پروژه

سناریویی که بایستی پیاده سازی شود به شرح زیر است:

۵ دکل در خطی افقی قرار دهند. ۱۰ کاربر از موقعیت های تصادفی اطراف دکل اول شروع به حرکت به سمت دکل پنجم می کنند. در طی مسیر در اثر ضعف سرویس دهی یک دکل و قدرت سرویس دهی دکل بعد بایستی عملیات handover انجام شود. این سناریو در دو حالت: ۱. حرکت با سرعت ثابت ۲. حرکت با سرعت متغیر پیاده سازی می شود. همان طور که در صورت پروژه گفته شده بهترین منبع برای کد نویسی NS3 نمونه های خود برنامه است. برای این سناریو خاص از کد lena-x2-handover-measures موجود در `src/ltc/examples/` استفاده کردم. کد را بنا به نیازمندی هایم تغییر و موارد اضافی را حذف کردیم. حال توضیح مختصری بر کد می دهیم. کد سناریو در فولدر آپلودی و با نام `code.cc` قرار دارد.

● توابع جهت logging

۶ تابع با خروجی void در اول کد دیده می شوند. این توابع در اصل نوشتن های ساده ای هستند. زمانی که در سناریو تغییری رخ می دهد؛ مانند، ایجاد یک اتصال به یک دکل و از سمت یک کاربر، یا شروع رویداد handover و موفقیت آمیز بودن آن و پایان آن توسط این توابع رخداد جدید در فایل تعریف شده به صورت global چاپ می شود.

● تابع main

در ابتدای ورود به این تابع، متغیر های مورد نیاز تعریف شده اند و بنا به نیاز به راحتی می توان مقدارشان را تغییر داد.

در ادامه برای ایجاد و مدیریت اتصالات LTE پویتری به یک نمونه از کلاس lteHelper ایجاد کردم. در ضمن برای شبکه EPC که هسته اتصالات LTE است یک نسخه از کلاس PointToPointEpcHelper تولید کردم.

همچنین نیازمندی های lteHelper از جمله نوع الگوریتم handover و صفات مورد نیاز آن و مقادیرشان.

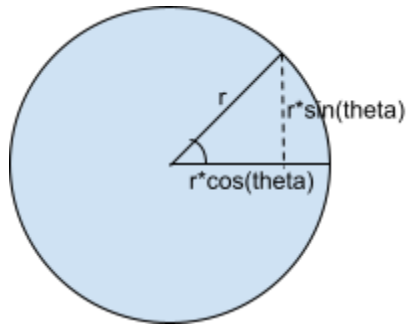
در مرحله بعد بایستی گره های شبکه را تشکیل دهیم. یک دسته (NodeContainer) از گره ها به عنوان دکل ها

و دسته ای دیگر برای کاربران می سازیم. در دسته تشکیل شده به تعداد گره های دلخواه گره می سازیم. حال

بایستی به گره ها مکان اولیه تخصیص دهیم. این کار با کمک ListPositionAllocator و MobilityHelper

و برای هر گره با کمک یک vector انجام می شود. برای موقعیت تصادفی هم مکان کاربران را روی یک دایره

حول دکل اول در نظر گرفتیم. که زاویه کمان به صورت تصادفی تولید می شود. مانند تصویر زیر:



همچنین بایستی مدل حرکت هم مشخص شود. این کار هم با کمک MobilityHelper انجام می شود. برای دکل ها که ثابت هستند مدل حرکتی ConstantPositionMobilityModel قرار می دهیم. برای کاربران اگر حرکت با سرعت ثابت مد نظر است از ConstantVelocityMobilityModel و اگر حرکت با سرعت متغیر مد نظر است از ConstantAccelerationMobilityModel استفاده می کنیم. در ضمن سرعت اولیه و شتاب را هم مشخص می کنیم.

در ادامه هم توان ارسال دکل ها را مشخص می کنیم. گره دکل ها و کاربران که در مرحله قبل ساخته شده است را بر روی lteHelper نصب می کنیم. همچنین گره همه کاربران را به اولین دکل متصل می کنیم. در مرحله بعد هم یک x2Interface روی دکل ها ایجاد می کنیم. این یک مدل جدید است که باعث راحت تر انجام شدن handover می شود.

در آخر هم امکانات ایجاد log های lte را فراهم می کنیم و با دستور AnimationInterface anim ایجاد فایل xml را می دهیم. و شبیه سازی را اجرا می کنیم.

بروزرسانی فایل wscript

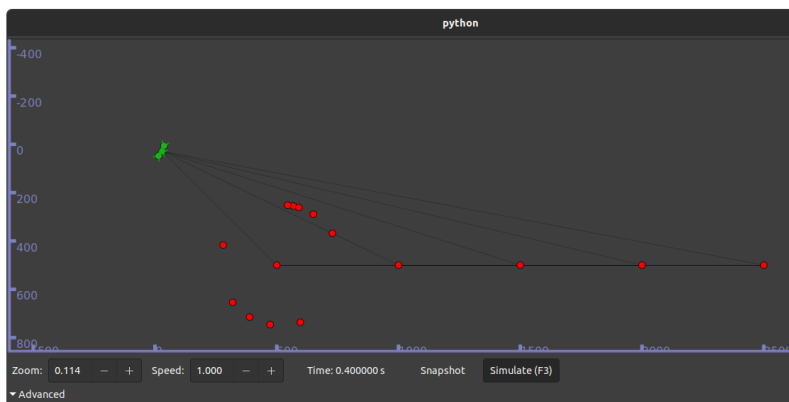
برای اجرا مناسب کد و شبیه سازی قطعه زیر را به فایل wscript اضافه می کنیم.

```
obj = bld.create_ns3_program('code', ['lte', 'netanim'])
obj.source = 'code.cc'
```

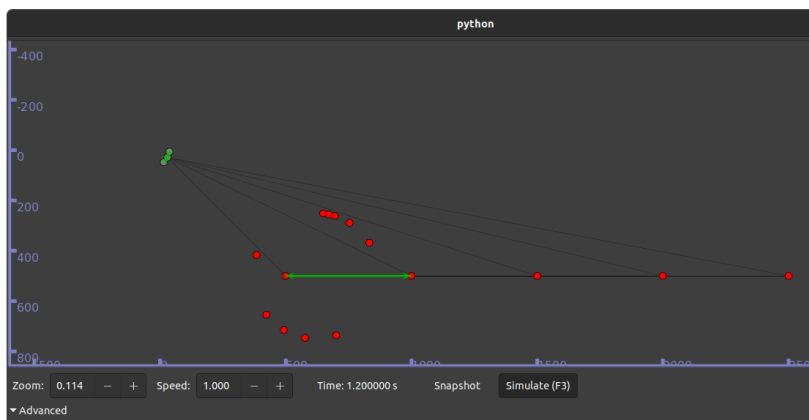
نتایج و خروجی شبیه سازی

سرعت ثابت

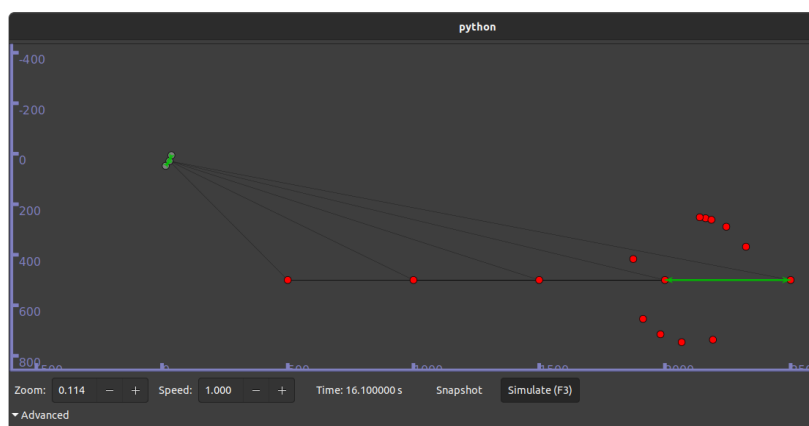
آغاز

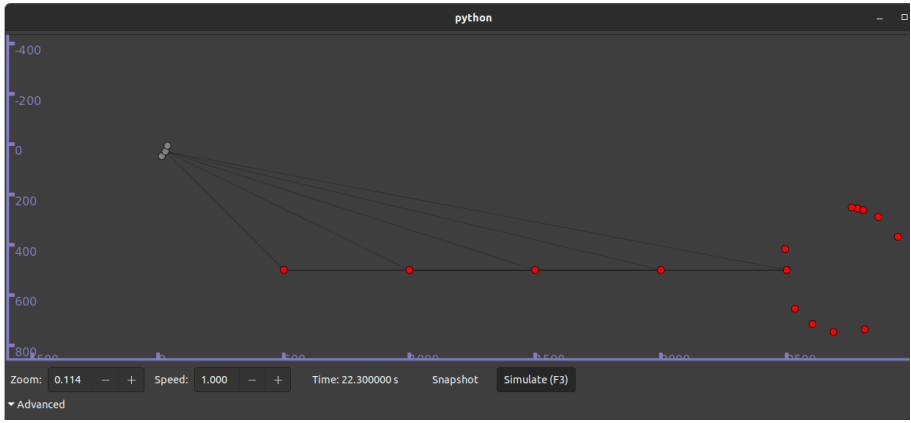
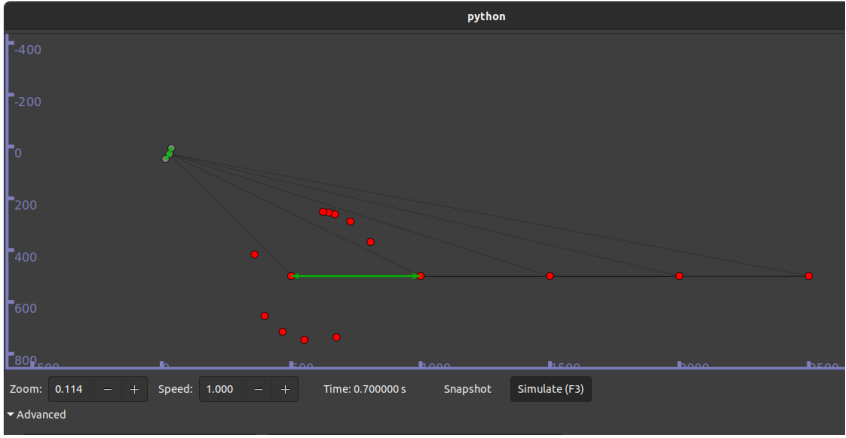
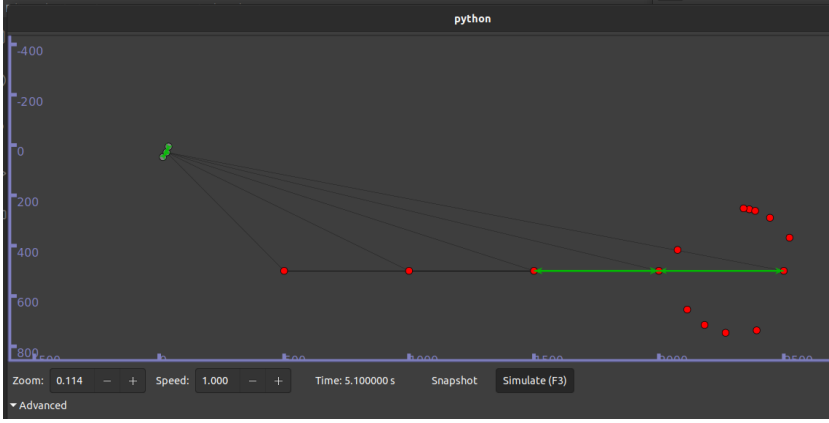


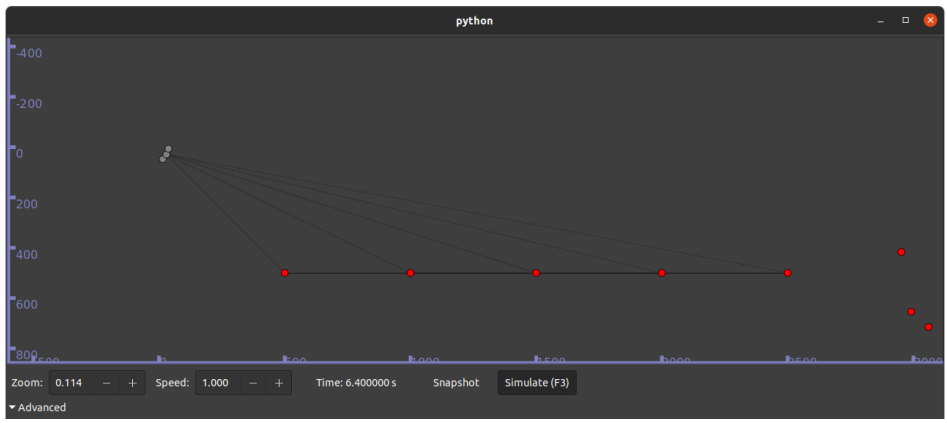
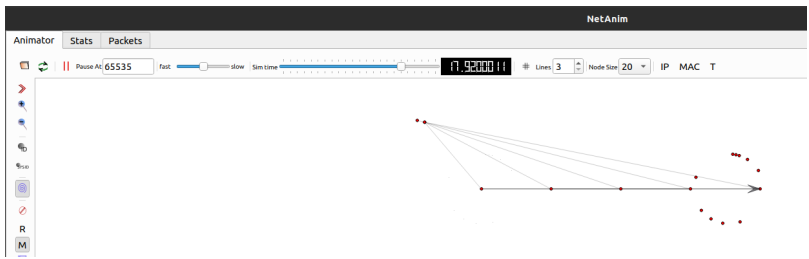
Handover - از دکل
اول به دوم



Handover - از دکل
چهارم به پنجم



	<p>پایان</p>
<p>سرعت متغیر - شتاب ثابت</p>	
	<p>Handover - از دکل اول به دوم</p>
	<p>Handover - از دکل چهارم به پنجم</p>

	<p>پایان</p>
<h2>NetAnim</h2>	
	<p>در این فضا به دلیل مشابه بودن با بالا تنها یک تصویر به نمایندگی گذاشته می شود.</p>

فایل‌های آپلودی

فایل `code.cc` کد سناریو قابل مشاهده است.

در فایل‌های `constantAccelerationLog.txt` و `constantVelocityLog.txt` به ترتیب لاگ خروجی حالات سرعت ثابت و سرعت متغیر قابل مشاهده است.

همچنین فایل `code.xml` جهت اجرا در Netanim قرار گرفته است.

به همراه `wscript` که نمونه کوچکی از `wscript` برای این فایل سناریو است.