MHRD
Ministry of Human Resource Development
Government of India

GLOBAL INITIATIVE OF ACADEMIC NETWORKS

Software Based Networks:

SDN and Integration of Virtualization in Networks

December 19 - 23, 2017

National Institute of Technology Karnataka (NITK), Surathkal, India

Introduction to P4 (Programming Protocol-Independent Packet Processors)

Objectives of the tutorial:

1. Installation of P4 compiler and Behavioural Model
2. Programming aspects of P4
3. Using P4

Prepared by:

Wireless Information Networking Group (WiNG),

National Institute of Technology Karnataka, Surathkal,

Mangalore, India.

Email address: wing [at] nitk [dot] ac [dot] in

1. Installation of P4 compiler and Behavioural Model

P4 compiler and behavioural model are two sub-repositories of p4lang github repository. The behavioural model is a C++ software switch that will behave according to P4 program. The P4 compiler that we are going to use today is a compiler for behavioural model. It takes P4 program as input and generates a JSON file as output, which can be loaded into the behavioural model. There is one more compiler that comes under the parent repository of p4lang, known as p4c. It provides a standard frontend and midend which can be combined with a target-specific backend to create a complete P4 compiler.

Step 1: Open a new terminal:

```
Ctrl + Alt + t
```

(above command will open a new terminal)

Step 2: Install git:

```
sudo apt-get install git
```

(above command will install git version control in your machine)

Step 3: Go to Desktop via terminal:

```
cd Desktop
```

Step 4: Clone behavioural model source code from github:

```
git clone https://github.com/p4lang/behavioral-model.git bmv2
```

Step 5: Install dependencies of behavioural model:

```
cd bmv2
sudo ./install_deps.sh
```

(above command will install all the dependencies related to compile the behavioural mode)

Step 6: Configure the behavioural model source code:

```
./autogen.sh
./configure
```

(After configuring you should get Makefile (with no extension) in the same directory)

Step 7: Build and install the behavioural model:

```
make -j 6
sudo make install -j 6
```
(will take 5 minutes)

(Above commands will compile and generate the binaries for bmv2)

Step 8: Come out of bmv2 directory:

```
cd ..
```

Step 9: Clone p4c-bmv2 compiler source code from github:

```
git clone https://github.com/p4lang/p4c-bm.git p4c-bmv2
```

(Above command clone the p4 compiler for bmv2)

Step 10: Go inside p4c-bmv2 directory:

```
cd p4c-bmv2
```

Step 11: Installing the dependencies of p4c-bmv2

```
sudo pip install -r requirements.txt
sudo pip install -r requirements_v1_1.txt
```

Step 12: Build and install the p4c-bmv2

```
sudo python setup.py install
```

Step 13: Check p4c-bmv2 is installed or not

```
p4c-bmv2 -h
```

(Above command will display the help page of the p4c-bmv2)

Step 14: Come out of p4c-bmv2 directory.

```
cd ..
```

2. Programming aspects of P4

Before using P4, refer the following tutorial-slides for understanding its programming aspects:

https://github.com/p4lang/tutorials/raw/master/SIGCOMM_2016/p4-tutorial-slides.pdf

3. Using P4

After the successful installation, follow these steps to run a P4 example program.

Step 1: Clone the sample program from tutorial section of p4lang:

```
git clone https://github.com/p4lang/tutorials
```

(Above command will get the sample p4 program from github.)

Step 2: Go inside the source_routing folder:

```
cd tutorials/SIGCOMM_2015/source_routing
```

Step 3: Extract the solution.tar.gz file:

```
tar xvzf solution.tar.gz
```

(Above command will untar the solution directory)

Step 4: Copy the P4 program and command.txt from solution directory to p4src directory:

```
cp solution/commands.txt ./
cp solution/p4src/source_routing.p4 p4src/
```

(Above commands will copy the existing example to their respective places)

Step 5: Open P4 program

```
gedit p4src/source_routing.p4
```

Step 6: Compile the p4 program from command prompt after closing gedit.

```
p4-validate p4src/source_routing.p4
```

(above command will check the syntax and semantics of p4 program)

Step 7: Install mininet (behavioural model uses mininet to simulate p4 program)

```
sudo apt-get install mininet
sudo pip install scapy thrift networkx
```

(above command will check the syntax and semantics of p4 program)

Step 8: Run run_demo.sh python script

```
./run_demo.sh
```

(run_demo.py will generate JSON file for P4 program and load the topology into mininet)

Step 9: Play with mininet terminal by putting following commands

```
nodes
h1 ifconfig -a
s1 ifconfig -a
```

(run_demo.py will generate JSON file for P4 program and load the topology into mininet)

Step 10: open two xterm for h1 and h3

```
xterm h1
xterm h3
```

(Above commands with open two x-terminal for two host h1 and h3)

Step 11: Now run receive.py on h3 xterm

```
./receive.py
```

(above command will put h3 xterm into listen mode)

Step 12: Now run send.py on h1 xterm

```
./send.py h1 h3
```

(send.py will take two arguments: sender host and receiver host respectively)

Step 13: Now you would be able to send messages from h1 xterm to h3 xterm encapsulated in packets with following format:

00000000 00000000 | 00000002 | 03 | 01 | Hello

16 bit preamble | No. of hosts | Host 1 | Host 2 | Payload

Learn more!

Why P4?: http://onrc.stanford.edu/p4.html

About P4: https://p4.orgl

P4 source code: https://github.com/p4lang

P4 installation: https://github.com/p4lang/tutorials/blob/master/SIGCOMM_2015/README.md

P4 tutorials: https://github.com/p4lang/tutorials

P4 tutorial at SIGCOMM-16: https://www.youtube.com/watch?v=OsPGtJNsTil

P4 based Applications:

1. PISCES: http://pisces.cs.princeton.edu/
2. NetPaxos: http://www.inf.usi.ch/faculty/soule/netpaxos.html
3. In-band Network Telemetry: https://p4.org/p4-spec/docs/In-band%20Network%20Telemetry%20(INT).pdf