Automating Planning
Final Assignment

Rafael Eichelberger
Pontificia Universidade Catolica PUC-RS
PGCC
Porto Alegre, RS

Abstract

This report presents a classical planning for network topologies using Planning Domain Definition Language (PDDL). The main idea is to set up network topologies, interconnecting switches, network devices and hosts to plan service functions paths at minimum cost. Given a set of ordered network devices as goals, the planner will compute forwarding path for network packets throughout these network devices.

Introduction

Most network deployments, besides switches and routers, are composed by specialized network appliances called "middleboxes". Middlebox is defined as any intermediary device performing functions other than forwarding packet on the datagram path between a source and destination hosts. Usually, network services require a set these network functions to be deployed. With the growing of SND (Software-Defined Network) and NFV (Network Functions Virtualization) areas, emerges the possibility of realize a dynamic service function chaining (SFC). SFC is defined as an abstract view of network functions and the order in which they need to be applied. Figure 1 shows a simple example of SFC. There are two hosts and two network functions (middleboxes), and all connected to a single switch. A possible SFC for this example would be host 1 communicating with host 2 passing first through firewall and then proxy ($host1 \rightarrow firewall \rightarrow proxy \rightarrow host2$).

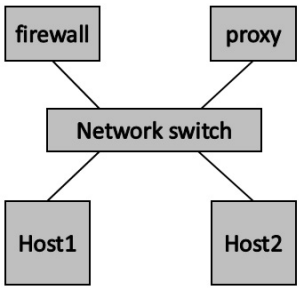

Figure 1: Simple example of network topology.

The assignment idea is to use a classical planner to compute network forwarding paths, given an SFC input. In more complex topologies with multiple switches and network functions, there are multiple path possibilities. A PDDL domain was written with operators for packets being able to navigate in network topologies. Network topologies will be the planner problems. The planner output is the forwarding path at minimum cost. Planner action costs are modeled to differentiate whether the network functions are virtual or physical. Cost actions were also modeled to distinguish switches links capability. The problems goals are the hops of the chain. Following the Figure 1 example, the problem would have three goals, the packet must be at the firewall, then proxy finally host2.
In this report first is presented the planner used, and the necessary modification to adapt the necessity for SFC planning. Then it is explained the domain implementation with PDDL predicates and actions, and a domain validation test. Finally, it is realized SFC experiments using tree and fat-tree network topologies and conclusions.

Planner

The planner used to develop this assignment was Pyperplan, which is a lightweight STRIPS planner written in Python. There was necessary to perform modifications in the code planner to be able to attend the requirements of SFC domain.
SFC demands ordered goals and classical PDDL planner does not support that. It is possible to define multiple goals with the 'AND' clause, however, there is no order guarantee at all. The nearest goal will be computed first. In order to support ordered goals, the planner parser was changed to receive multiple goal statements. A new parser statement was defined for the parser stops to look for goals statements, as shown bellow.

```
(:goal (atPacket nf1))
(:goal (atPacket nf2))
(:goal (atPacket h2))
(:end)
```

The search function, which actually computes the plan, is called once for each different goal. Once a goal plan is obtained, the final state is assigned to the initial state of the next search goal. This is done by extracting the effects of

the last action from the previous solution and assign it to the initial state of the next goal search. Therefore, a plan for an ordered set of goals is achieved.
Pyperplan parser does not support to set cost actions. Then the following statement was added to support cost increasing in the actions effect.

```
(increase (total-cost 4))
```

Default action cost for each action is one, with the above statement the cost action can be changed to any value, four in this example.
Pyperplan implements several search algorithms. The search algorithm used was a-star, which is traditionally used in path-finding and graph traversal. A-star uses a best-first search and finds a least-cost path from a given initial node to one goal node. The Pyperplan implementation of a-star algorithm considers all action of cost one. Hence, the value passed in the total-cost statement is assigned here instead of cost one. When total-cost is not informed the default cost value will be one.

Domain approach

Planner domain was written regarding a packet forwarding at different topology problems. The predicate "atPacket" defines the current packet place, which could be at switches, network devices (middlesboxes) and clients, also called as hosts. The initial state of "atPacket" will be the client input host. The network packet will move through forwarding actions. Forwarding action can be from switches to switches, from/to switches to/from devices and from/to switch to/from client. Forwarding action can just be performed by connected devices. Connections can be performed by more than one predicate. The "connected" predicate is modeled as a physical connection, which will lead to a forwarding action of default cost. There are others "connected" predicates which are modeled as virtual/busy connection, and will lead to forwarding actions with costs bigger than one.
Problem goals will be multiple chain hops, which are the network devices and a final client host destination. Network devices such as firewalls, proxies, etc, can be both virtual or physical. SFC will define as goals the required devices and does not care if the device is virtual or physical. The objective is to obtain the minimum plan cost. To model several instances of the same device, whether physical or virtual, differents connection are used. Virtual connections are used to forwarding actions with greater cost effects. Figure 2 shows an example of real topology with two Firewall instances and the equivalent planner modeling. The connections "connA" and "connB" are used to distinguish the two firewall instances. Therefore, a predicate, to store the packet previous switch, is used to guarantee that the packet come back to the same switch that came into device.

Problems files can just define the Firewall as a goal for a chain hop. The planner will be able to compute the cheapest cost whatever it uses a virtual or physical firewall.

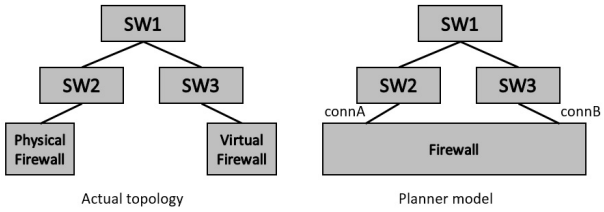

Figure 2: Planner model for virtual and physical devices.

Domain Validation

Described domain was validate using a simple problem. The objective of this validation is to observe planner decision based on different cost action values. Figure 3 shows the proposed problem. Two Firewall instances (FW) are connected. One FW is modeled as physical with connection cost one (conn-cost=1). The other FW, modeled as virtual, is connected with cost different from one (conn-cost=v), where "v" is the cost value defined in the forwarding action for virtual devices in the domain file. Then the cost to access a virtual device is higher than a physical one.

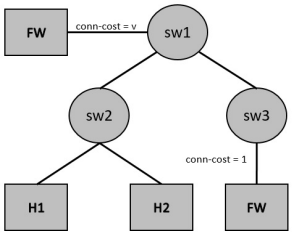

Figure 3: First defined scenario.

The packet initial state is H1 and the SFC goal is `H1 -> FW -> H2` as shown bellow:

```
(:goal (atPacket fw))
(:goal (atPacket h2))
```

First is set the cost v, for virtual connection, to two. The planner solution is:

```
(forwardfromclient h1 sw2)
(forwardpacket sw2 sw1)
(forwardtovirtdevice sw1 fw)
(forwardfromvirtdevice fw sw1)
(forwardpacket sw1 sw2)
(forwardtoclient sw2 h2)
```

The plan length is 6 with total cost of 8. Total cost was incremented by one for each default action and 2 for access the virtual device, in both directions. It is possible to observe that the planner chose the virtual firewall, forwarding the packet to FW connected in SW1. Next test is performed changing the v value to three. Planner solution is:

```
(forwardfromclient h1 sw2)
(forwardpacket sw2 sw1)
```

```
(forwardpacket sw1 sw3)
(forwardtodevice sw3 fw)
(forwardfromdevice fw sw3)
(forwardpacket sw3 sw1)
(forwardpacket sw1 sw2)
(forwardtoclient sw2 h2)
```

Now the cost to access the virtual firewall is higher, then the planner solution chose the physical firewall, with a longer plan with length 8 and the same cost as before, 8.

Experiments

The written domain was used to test different SFC goals in practical network topologies with a greater amount of switches and network devices. It was also exercised bidirectional chains, which is chain specifications in both directions. Then the client destination could response to the source throughout another chain. Experimentations were divided in two different topologies, the tree topology and the fat-tree topology.

Tree topology

Tree topology can be observed in Figure 4. This is a scalable topology with distinct levels of switch connection. Although it is possible to scale, this topology requires high-performance switches on top of the tree. The top level switch concentrates all network traffic from lower level switches. For this experiment, the topology problem was defined as shown in Figure 4, with virtual devices and hosts connected to the bottom of the tree and physical devices connected at higher tree level switches. The switch communication cost is incremented for each tree level, from cost one, in the bottom, to cost four in the top level. Therefore, domain actions are defined to four levels of switch communication to distinguish the cost variation.

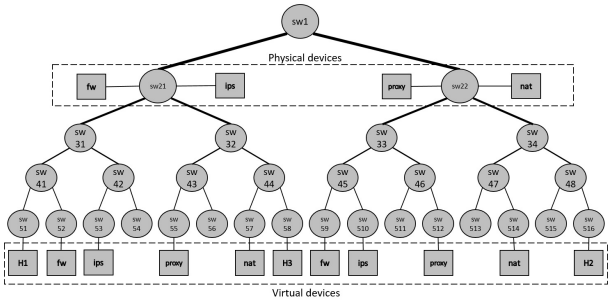

Figure 4: Tree topology problem.

The experimentation was done ranging the cost action to forward the packets to virtual devices. The packet return from device to switch is kept as cost one. The first SFC goal is a bidirectional chain from H1 to H2:

```
(H1 -> FW -> IPS -> NAT -> Proxy -> H2)
(H2 -> NAT -> FW -> H1)
```

To reach H2, the packet will need to traverse the whole tree topology. The experiment was done changing the cost value to reach virtual devices and then check the length and total cost planner solution. Planner length is the number of action taken to reach the goal. If all actions had cost one, the length and cost would remain the same. Results are shown in Figure 5.

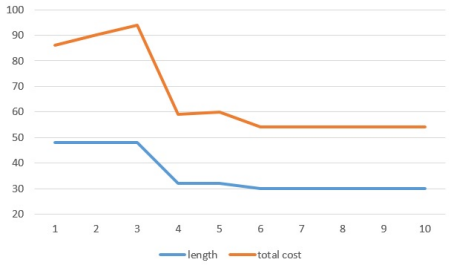

Figure 5: First SFC goal in tree topology.

With cheaper virtual devices access, the planner length is bigger due to the longer path to reach virtual devices. In this circumstance, the planner solution decided to forward to virtual devices because the high cost of switch top level connections. When forwarding action cost to virtual devices increases, planner solution points to physical devices and thus smaller plan length. Smaller length is observed due to physical devices are located in the path to reach the host destination. When virtual cost action reaches a significant value, the length and cost solution keep constant because the planner solution is just using physical devices.
The second SFC goal is a bidirectional chain from H1 to H3. To do that just half of the tree topology is used:

```
(H1 -> FW -> IPS -> NAT -> Proxy -> H3)
(H3 -> NAT -> FW -> H1)
```

Again, the experiment was done ranging the virtual cost for forwarding action to virtual devices. The planner solution is shown in Figure 6.

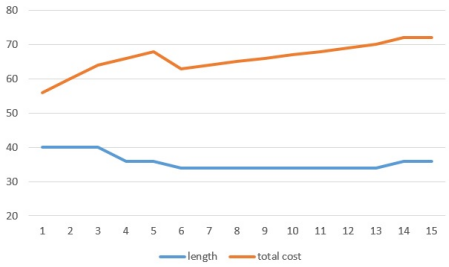

Figure 6: Second SFC goal in tree topology.

The total cost increases with forwarding action cost growing. The steps in the graph are due to planner path solution modification regarding the previous solution. In order to keep the minimum cost, the planner changed from a virtual network device to a physical network device.

Fat-tree topology

Fat-tree topology is intent to solve the main restriction of tradition tree topology. Fat-tree is also scalable, how-

ever, instead of gather all traffic in the same link for high level switches, it uses redundant links with multiple core switches. Figure 7 shows the fat-tree topology.
This topology is divided by pods, with 4 switches. The pods are interconnected through core switches. A pod is formed by edge switches to connect with devices and hosts, and aggregation switches connecting with core switches. Both pod and core switches may be the same, with same performance requirements, as any switch concentrates traffic. Hosts connected at different pods have multiple communication paths, however, the minimum cost is always the same. The topology problem domain used in this experiment is shown in Figure 7. It is possible to observe the clients and network devices connected on edge switches.

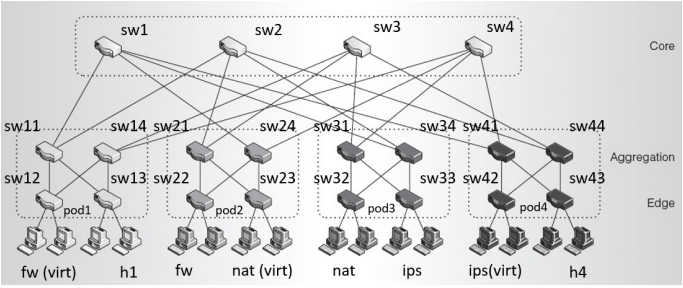

Figure 7: Tree topology problem.

In the experiment, all switch interconnections have cost one. The cost to access physical devices also is one and the cost to access virtual devices is two. Regarding devices connected in this topology, the first SFC goal experimented is:

```
(H1 -> FW -> NAT -> Proxy -> H4)
(H4 -> Proxy -> H1)
```

Bellow, can be seen the planner solution for the proposed SFC goal, where the letters 'V' and 'P' denote whether the device is virtual or physical:

```
(H1 -> FW(V) -> NAT(P) -> IPS(P) -> H4)
(H4 -> IPS(V) -> H1)
```

As can be seen by the planner solution, it will forward packets to virtual devices when the device is connected in the same pod of the previous one. In this case is cheaper access a virtual device than a physical device in another pod. Fat-tree topology can also validate the planner solution for minimum cost. There are multiple paths for devices from different pods to communicate, although the planner always computes the minimum cost. In this case, the maximum possible cost must be six, which is the cost to communicate with a physical device in another pod. When changing pod, the planner must choose a physical device because certainly the cost will be cheaper than a virtual device. Virtual devices are just chosen when they are in the same pod from previous chain hop.
A second experiment was done varying the number of network function in the chain using fat-tree topology, results are shown in Figure 8.

As can be observed, the behavior of chain hops increasing

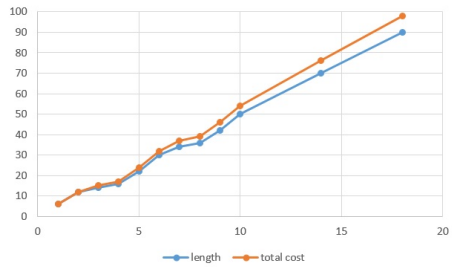

Figure 8: SFC hops variation in fat-tree topology.

is almost linear. The linear behavior is due to an almost constant cost increasing when adding a network function in the chain. This cost increasing value may range from two, when hops are in the same switch, to maximum six, when hops are at separate pods.

Future work suggestion

SDN architecture is defined by decoupling switches control and forwarding plane. With this decoupling, switches are agnostic regarding network topologies and need to be configured by a central controller. Switches flow table are dynamically configured with rules to forward packets. Flow rules are network field (IP, MAC, etc) to be matched with coming network packets and then take proper actions, also defined in the rule. In an SDN environment, multiples SDN application can install flow rules on switches. A possible planner automation would plan switches flow rules regarding best path, possible rules conflicts, flows intersections, etc. Therefore, the SDN controller could compute some strategies regarding the actual network topology.

Conclusion

The assignment showed a real application for classic planning. Although the experiments were done in a high-level perspective, it was possible to observe new possibilities to apply planning automation in network area. The experience changing the planner code to adapt the assignment requirements was satisfying to understand the planning computation and search algorithms. Real network topologies were implemented with multiples network devices providing service function chaining. This planning automation could also be used as a simulation framework input for a variety of SFCs techniques. The results show the accuracy of implemented domain and its computation at minimum cost. A better approach to compute ordered goals could be achieved when the planner knows the full path goals, and chose the network device considering the further devices to optimize the total path instead just partial paths.

References

PyperPlan
https://bitbucket.org/malte/pyperplan.
A search algorithm*
https://en.wikipedia.org/wiki/A*$_{s}earch_{a}lgorithm$