Master Computer Science and Networking
Advanced Programming 2016/17
Homework 1

Alessandra Fais
Matr. 481017

The interpreter

The values used inside the interpreter are expressions: an expression can be a base type like a *forwarding table*, a *rule* of the forwarding table, a list of *IP addresses*, a list of *ports* (interfaces) or a *filter*. Other expressions are different types of operations: the interpreter offers the possibility to *install* a new forwarding table, *get* new *rules* for a certain IP address, *delete* a rule from a forwarding table (or simply *activate* or *suspend* it, if it is present inside the current forwarding table), activate a *security policy* or handle the *mobility*.
In order to evaluate an expression, the programmer must call the eval function on the expression, in the following way:

- Install(IPaddresses_e(<list of int addresses>), Ports_e(<list of int interfaces>))
  two parameters are required, a list of IP addresses (the network) and a list of interfaces (the output interfaces of the switch); these parameters are used to create a new forwarding table. The association between IP addresses and output interfaces is built up using a simple hash function that given the address produce as result the position of the designated port in the interfaces list; in the table is then inserted the new rule containing the association between the address and the port in position hash(addr).
- GetRules(dest_ip, IPaddresses_e(<list of int addresses>), Ports_e(<list of int interfaces>))
  three parameters are required, an IP address, a list of IP addresses (the network) and a list of interfaces (the output interfaces of the switch); these parameters are used to create a new set of rules for the IP address for which there were not suitable rules in the forwarding table (the function is invoked when the switch is not able to find a rule for the destination IP address and asks the controller for some updates).
- Delete(Rule_e(IPaddr, port, state), Table_e(<list of exp of type Rule_e>))
  two parameters are required, a rule (a tuple of source IP address, destination IP address, port, mode - Active or Suspended - and two counters for the policies) and a forwarding table. The result will be a new forwarding table that does not contain the specified rule. Any possible duplicate of the rule inside the table is removed.
- Suspend(Rule_e(IPaddr, port, state), Table_e(<list of exp of type Rule_e>))
  two parameters are required, a rule (a tuple of source IP address, destination IP address, port, mode - Active or Suspended - and two counters for the policies) and a forwarding table. The result will be a new forwarding table in which the specified rule is marked as Suspended (if it is present).
- Activate(Rule_e(IPaddr, port, state), Table_e(<list of exp of type Rule_e>))
  two parameters are required, a rule (a tuple of source IP address, destination IP address, port, mode - Active or Suspended - and two counters for the policies) and a forwarding table. The result will be a new forwarding table in which the specified rule is marked as Active (if it is present).
- ActivatePolicies(read_num, write_num, Table_e(<list of exp of type Rule_e>))
  three parameters are required, the number of allowed reads and write operations for each destination (for the specified source) and a forwarding table. The counters are initialized to

a value greater than zero (-1 is the uninitialized value) and decremented: when a counter is zero a packet with that operation type that matches that rule is dropped.

- HandleMobility(old_IPaddr, new_IPaddr, Table_e(<list of exp of type Rule_e>))
  three parameters are required, the old IP address, the new one and a forwarding table. Each packet addressed to the old IP address is now forwarded through the output interface associated to the new IP address.

For the Suspend and Activate cases the recursion is stopped at the first occurrence of the specified rule inside the forwarding table: this is possible because when the switch search for a matching rule inside its forwarding table it stops at the first occurrence found. Even if a duplicate rule is present in the forwarding table, the matching rule will always be the first found. The only operation that checks if there are more occurrences of the same rule inside the forwarding table is the Delete, that removes possible duplicates.

A list is a type defined in the interpreter:
- a list of addresses or ports has this form: Cons(1, Cons(0, Nil))
- a list of rules has this form: Cons(Rule_e(IPaddr', port', state'), t)

The controller and the switch

The module capable of calling the interpreter is the controller. It is capable to use the functionalities offered by the defined domain specific language in order to control the behavior of the switch. The two modules are modeled as two different functions: the switch is able to send requests to the controller and the controller discerns between different requests and give back a reply. The switch can request a new forwarding table, missing rules or period updates (they can be updated tables - after delete, suspend, activate operations of some rule -, or updates that concern the policies, a filter or the mobility). The controller invokes the corresponding functionality of the interpreter and create a data structure suitable to be used by the switch. The data types and operations offered by the interpreter are totally hidden to the switch, that is only aware of the controller. The code and the simulation are commented with further explanations.

Final notes:
The project has been discussed and designed with the help of the other colleagues Giuseppe Astuti, Orlando Leombruni, Davide Scano.
The files are:

- The interpreter: homework1_DSL_final.ml
- The modules: homework1_SWITCH_final.ml
- The simulation: homework1_simulation_final.ml
- This report