

VIP/Senior Project Final Presentation Summer 2020

AmLight - Learning OpenFlow/SDN Network Research
and Experimentation

Team Members: Cindy Perez, Jose Mercado, Kenia
Chang He, and Yuyang Leng

Product Owner: Vasilka Chergarova

Professor: Masoud Sadjadi

School of Computing and Information Sciences
Florida International University

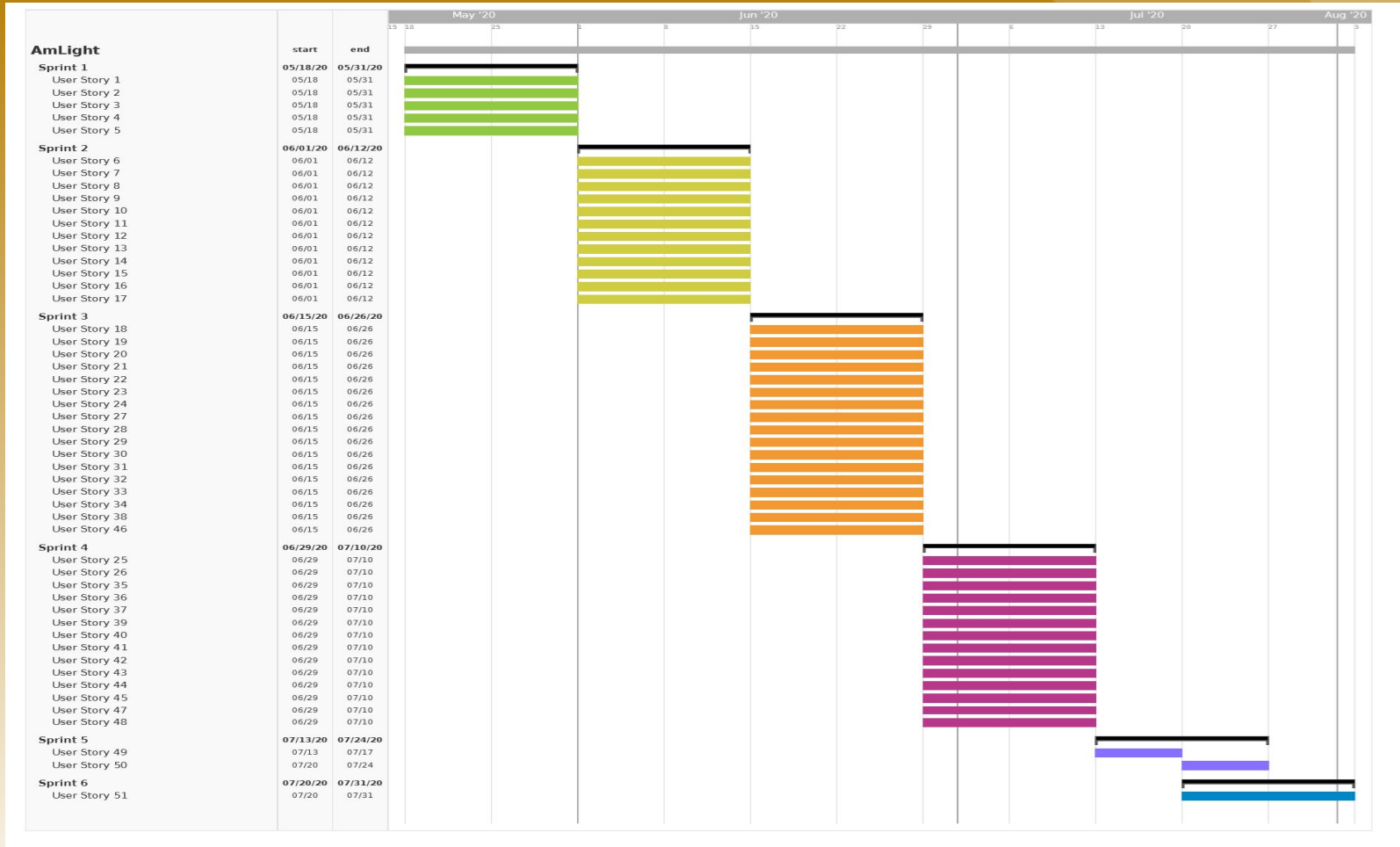
07/28/2020

Problem definition

- Whole Project:
 - Implementation and testing of different match fields for the Kytos OpenFlow of_core version 1.3 as well as the design and implementation of a IPv6 packet structure.
- Cindy Perez:
 - Design, implementation and testing of following match fields with masks: TCP Source, TCP Destination, IPv6 FLabel, ICMPv6 Type, ICMPv6 Code, ARP SPA, ARP TPA, ARP SHA, ARP THA, PBB ISID.
- Jose Mercardo:
 - Design, implementation and testing of following match fields with masks: IPv4 Destination, IP Protocol, Input Port, IPv6 ND Target, IPv6 ND SLL, IPv6 ND TLL, IPv6 EXTHDR, MPLS Label, MPLS TC, MPLS BOS, Tunnel ID.

- Kenia Chang He:
 - Design, implementation and testing of following match fields with masks: DataLink VLAN ID, VLAN Priority Code Point, Datalink Source, Physical Input Port, IP DSCP, IP ECN, IPv6 Source, IPv6 Destination, ARP OP.
 - Create test source for the IPv6 packet structure.
- Yuyang Leng:
 - Design, implementation and testing of following match fields with masks: Datalink Destination, Datalink Type, IPv4 Source, UDP Source, UDP Destination, ICMPv4 Type, ICMPv4 Code, Metadata, SCTP Source, SCTP Destination.

Project Management



User Stories

- AmLight/User Story 5: As a developer, I want to start working on a solution for the issue #75 that was given to us by our Product Owner, so I may implement the solution on the software given to us.
- AmLight/User Story 6: As a user, I want to be able to use the mask function for the Datalink VLAN ID matching field of a flow as of OpenFlow v1.3, so that I can utilize the OpenFlow v1.3 more realistically.

- AmLight/User Story 8: As a user, I want to be able to use the mask function for the Datalink Source matching field of a flow as of OpenFlow v1.3, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/User Story 9: As a user, I want to be able to use the mask function for the Datalink Destination matching field of a flow as of OpenFlow v1.3, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/User Story 11: As a user, I want to be able to use the mask function for the IPv4 Source matching field of a flow as of OpenFlow v1.3, so that I can utilize the OpenFlow v1.3 more realistically.

- AmLight/ User Story 12: As a user, I want to be able to use the mask function for the IPv4 Destination matching field of a flow as of OpenFlow v1.3, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 17: As a developer, I want to test the implementation for issue #75, so that I can make sure it is working for the users.
- Other User Stories:
- User Story 7, User Story 10, User Story 13, User Story 14, User Story 15, User Story 16

- AmLight/ User Story 18: As a user, I want to use the physical input port as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 21: As a user, I want to use the UDP Source as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 22: As a user, I want to use the UDP Destination as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 25: As a user, I want to use the IPV6 Source as one of the match fields for my low, so that I can utilize the OpenFlow v1.3 more realistically.

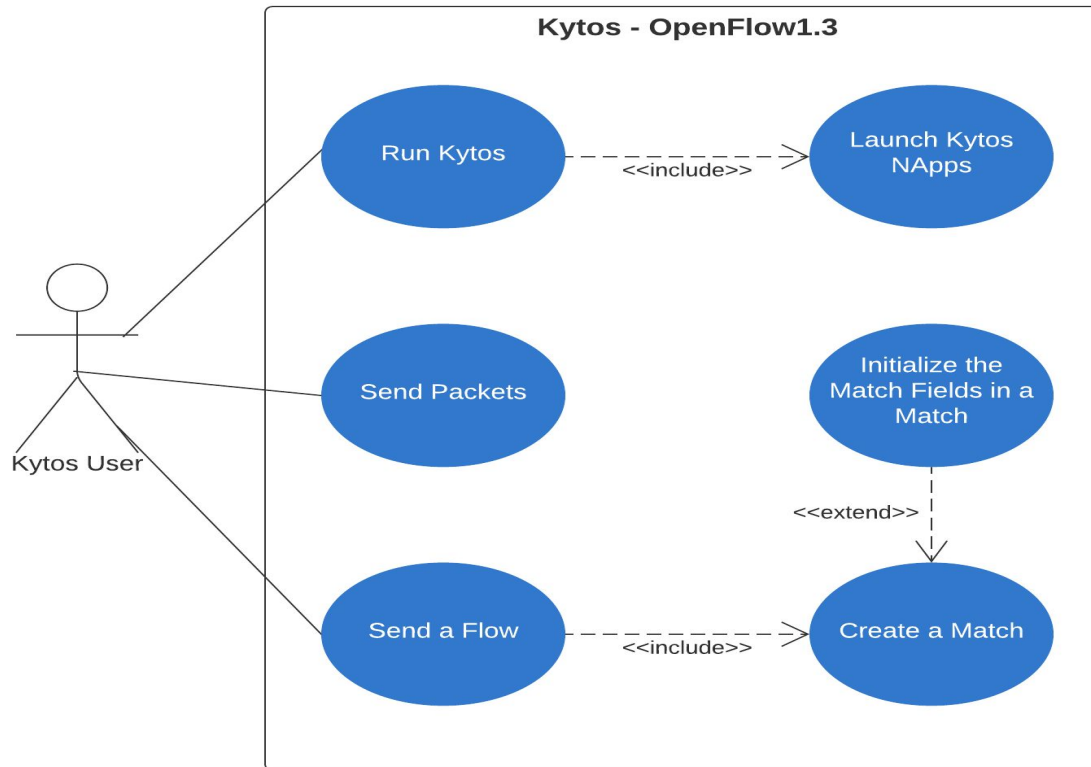
- AmLight/ User Story 26: As a user, I want to use the IPV6 Destination as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 27: As a user, I want to use the IPV6 FLabel as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 30: As a user, I want to use the IPV6 ND Target as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- AmLight/ User Story 33: As a user, I want to use the IPV6 EXTHDR as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.

- Amlight/ User Story 34: As a developer, I want to test the implementation for the first half of issue #77, so that I can make sure it is working for the users.
- Other User Stories:
- User Story 19, User Story 20, User Story 23, User Story 24, User Story 28, User Story 29, User Story 31, User Story 32

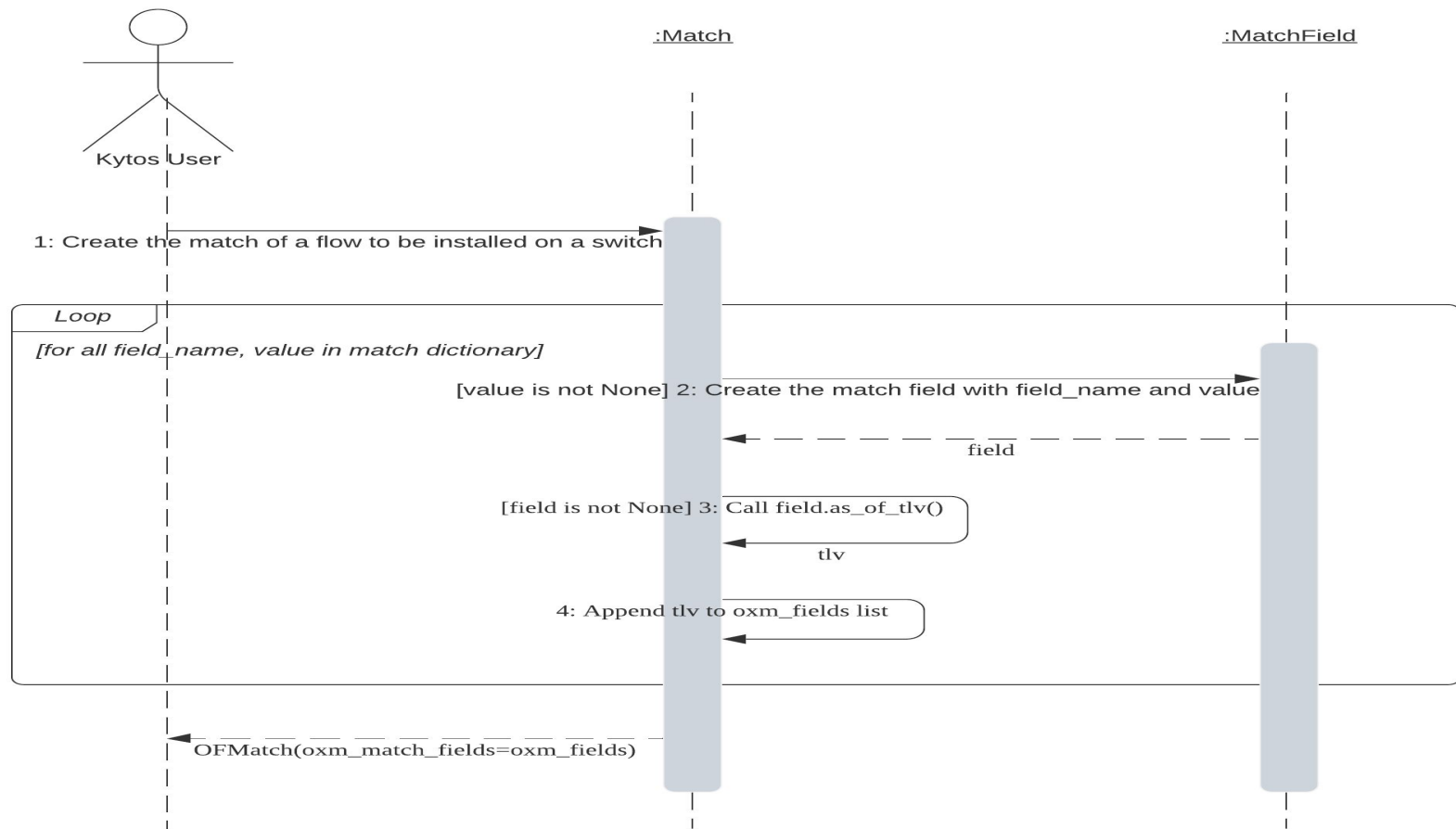
- Amlight/ User Story 35: As a user, I want to use the Metadata as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- Amlight/ User Story 38: As a user, I want to use the ARP OP as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- Amlight/ User Story 43: As a user, I want to use the MPLS Label as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- Amlight/ User Story 46: As a user, I want to use the PBB ISID as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.
- Amlight/ User Story 47: As a user, I want to use the Tunnel ID as one of the match fields for my flow, so that I can utilize the OpenFlow v1.3 more realistically.

- Amlight/ User Story 48: As a developer, I want to test the implementation for the second half of issue #77, so that I can make sure it is working for the users.
- Other User Stories:
- User Story 36, User Story 37, User Story 39, User Story 40, User Story 41, User Story 42, User Story 44, User Story 45
- Amlight/ User Story 49: As a developer, I want to implement and test the IPv6 packet structure, so that the IPv6 packets can be used.

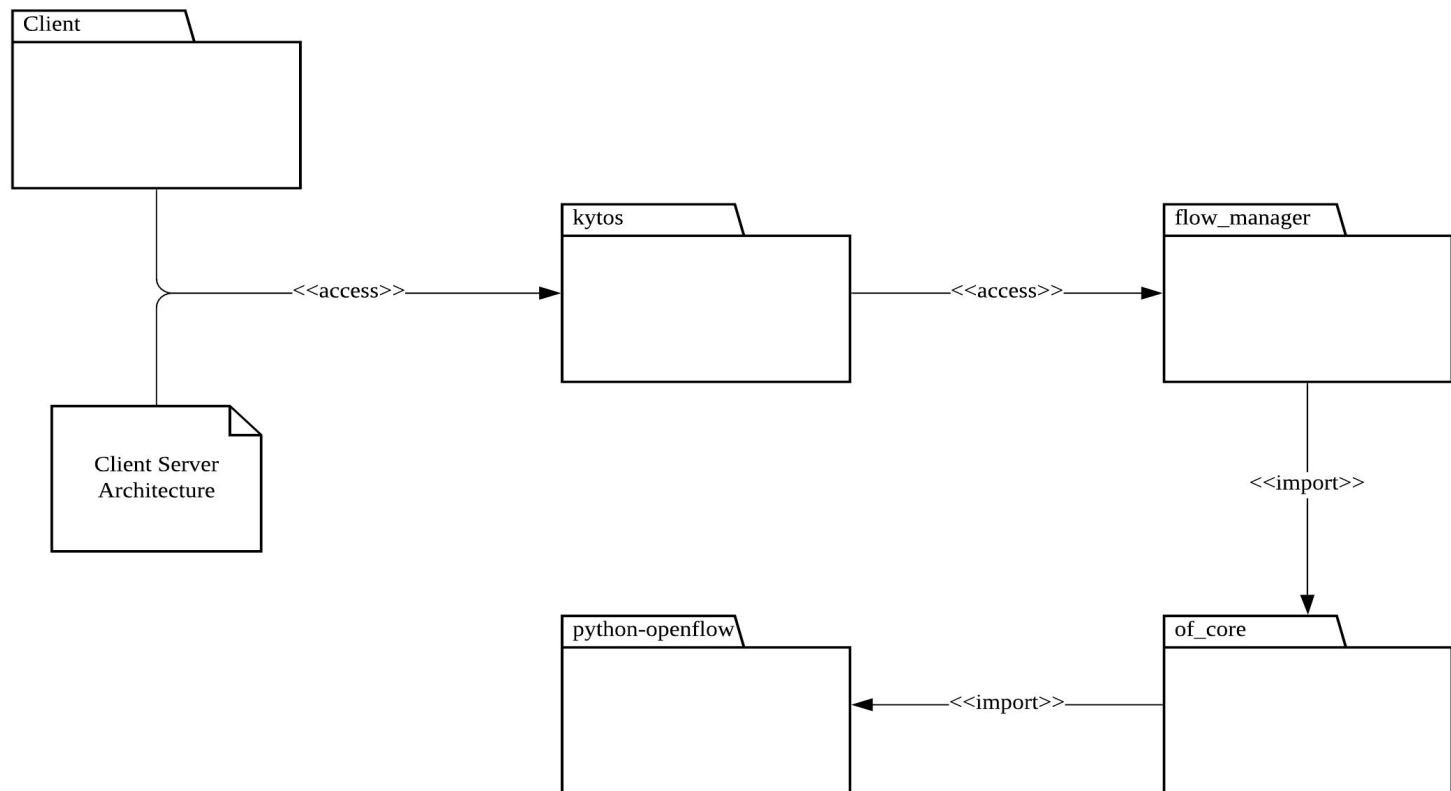
Use Case Diagram



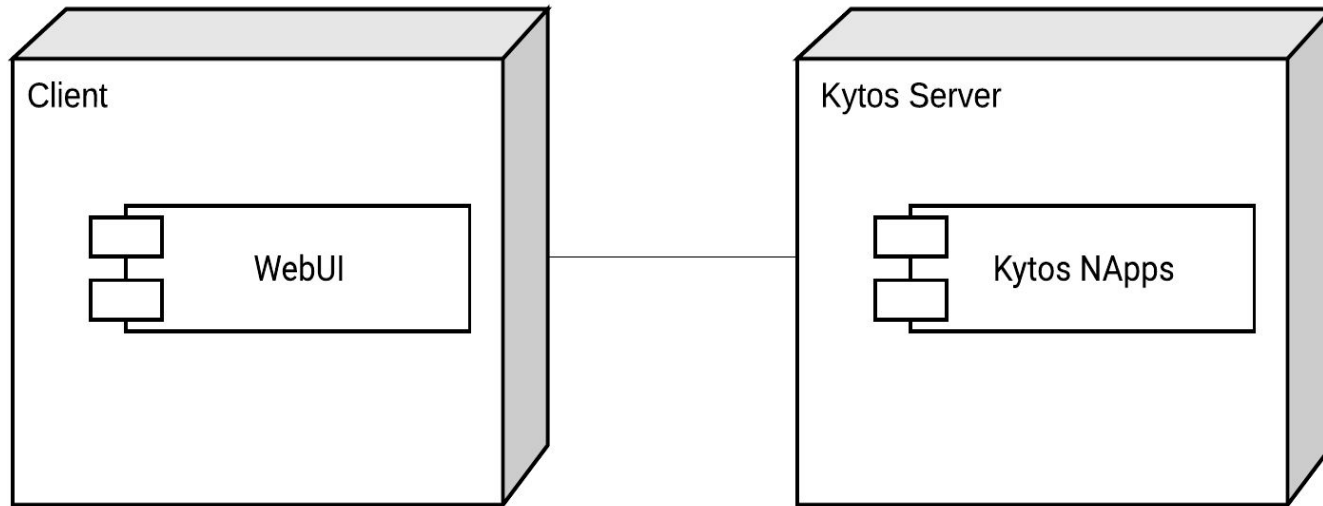
Sequence Diagrams



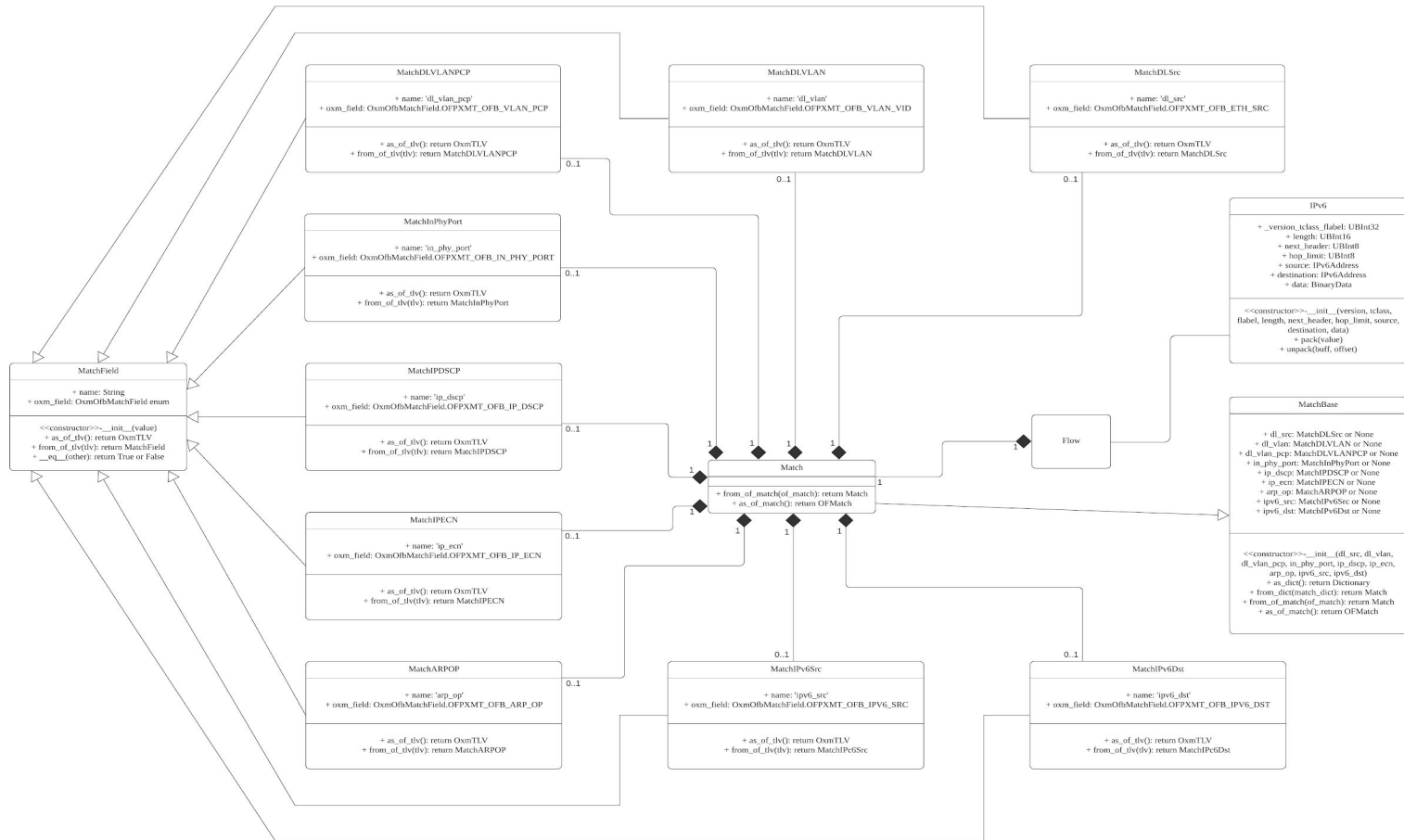
System Design: Architecture



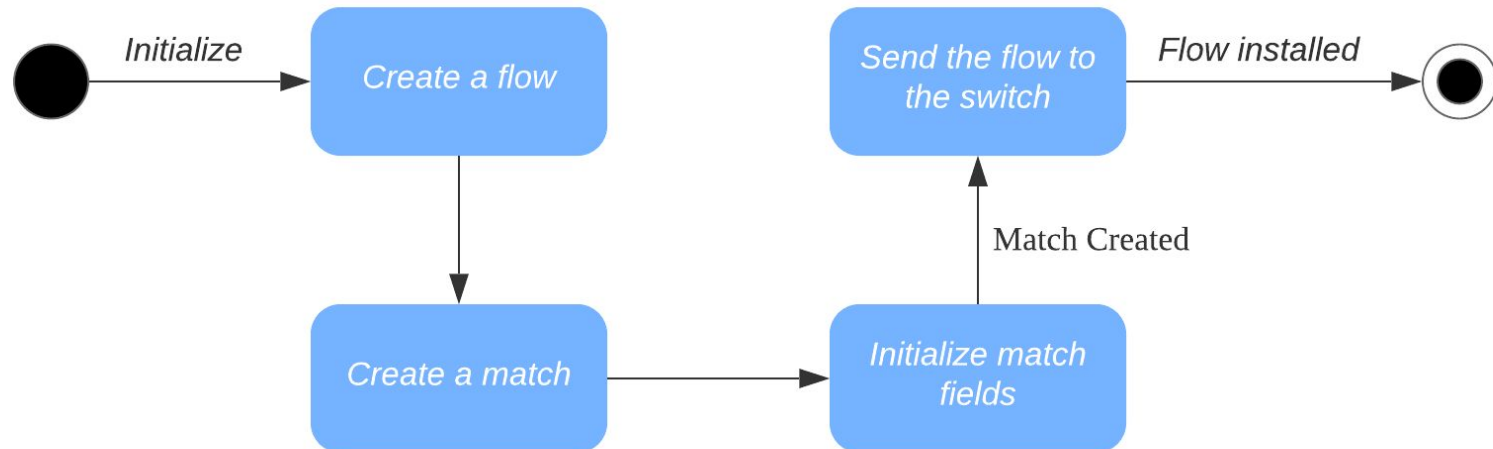
System Design: Deployment



Minimal Class Diagram

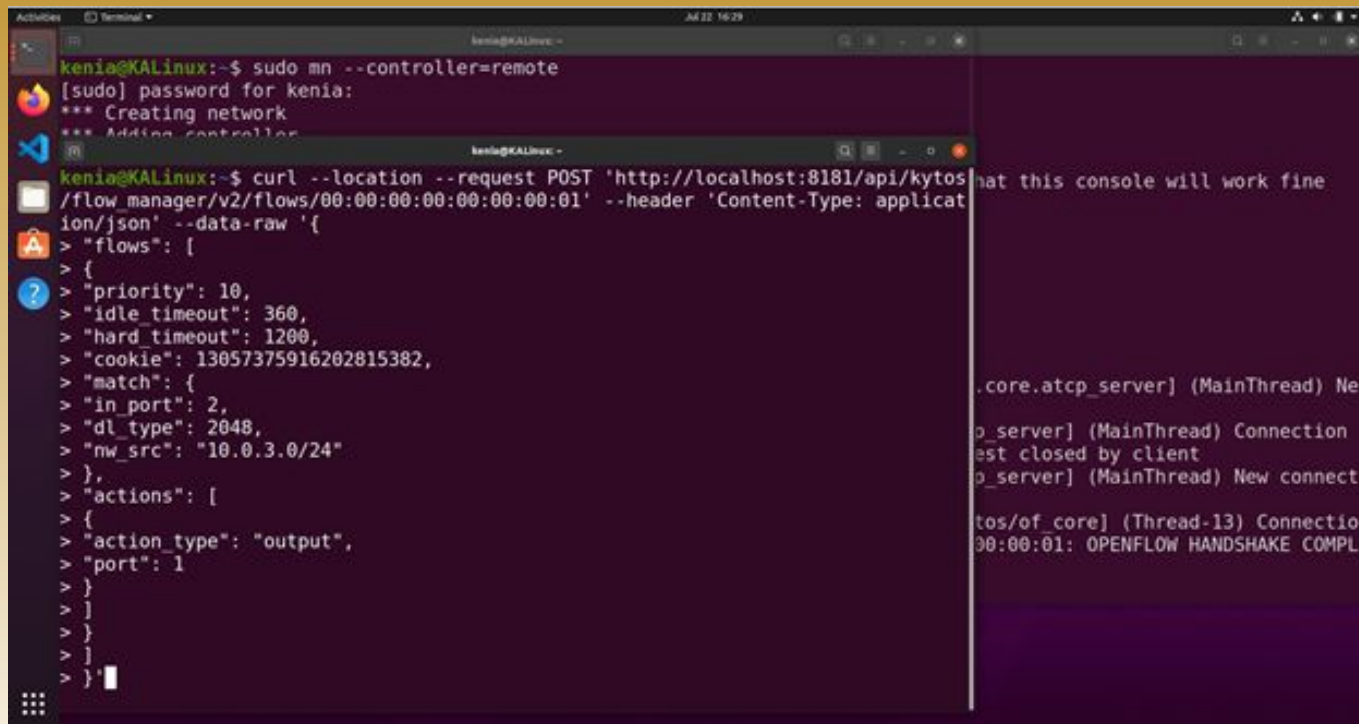


Statechart Diagrams



Test Suites and Test Cases

- Match Fields Test Cases



The screenshot shows a terminal window with the following commands and output:

```
kenia@KALinux:~$ sudo mn --controller=remote
[sudo] password for kenia:
*** Creating network
*** Adding controller

kenia@KALinux:~$ curl --location --request POST 'http://localhost:8181/api/kytos/flow_manager/v2/flows/00:00:00:00:00:00:01' --header 'Content-Type: application/json' --data-raw '{
> "flows": [
> {
>   "priority": 10,
>   "idle_timeout": 360,
>   "hard_timeout": 1200,
>   "cookie": 13057375916202815382,
>   "match": {
>     "in_port": 2,
>     "dl_type": 2048,
>     "nw_src": "10.0.3.0/24"
>   },
>   "actions": [
>     {
>       "action_type": "output",
>       "port": 1
>     }
>   ]
> }
> ]
> }'
```

The output of the curl command is a JSON response:

```
{
  "flows": [
    {
      "priority": 10,
      "idle_timeout": 360,
      "hard_timeout": 1200,
      "cookie": 13057375916202815382,
      "match": {
        "in_port": 2,
        "dl_type": 2048,
        "nw_src": "10.0.3.0/24"
      },
      "actions": [
        {
          "action_type": "output",
          "port": 1
        }
      ]
    }
  ]
}
```

Test Suites and Test Cases

- IPv6 Packet Unit Test

```
class TestIPv6(unittest.TestCase):
    """Test IPv6 packets."""

    def test_IPv6_pack(self):
        """Test pack/unpack of IPv6 class."""
        packet = IPv6(next_header=6, hop_limit=64, source="::1",
                      destination="::2", data=b'testdata')

        packed = packet.pack()
        expected = b'\x00\x00\x00\x00\x00\x00\x06@\x00\x00'
        expected += b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
        expected += b'\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
        expected += b'\x00\x00\x00\x02testdata'
        self.assertEqual(packed, expected)

    def test_IPv6_unpack(self):
        """Test unpack of IPv6 binary packet."""
        raw = b'\x00\x00\x00\x00\x00\x00\x06@\x00\x00'
        raw += b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
        raw += b'\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
        raw += b'\x00\x00\x00\x01somedata'
        expected = IPv6(next_header=6, hop_limit=64, source="::2",
                      destination="::1", data=b'somedata')
        expected.pack()
        unpacked = IPv6()
        unpacked.unpack(raw)
        self.assertEqual(unpacked, expected)
```

Summary

- The Project was a success, we were able to implement the matchfields including in our implementation the masks for each that were accepted by Kytos OpenFlow. We were also able to add our implementation of the IPv6 packet structure
- We would like to thank our product owner Vasilka Chergarova, as well as Antonio Francisco and Humberto Diogenes
- Questions?

Contact Information

- Cindy Perez:
 - cpere459@fiu.edu
 - <https://github.com/cpere459>
 - <https://www.linkedin.com/in/cindy-perez-b419571a9/>
- Jose Mercado
 - jmerc053@fiu.edu
 - <https://github.com/jmercado101>
 - <https://linkedin.com/in/jose-mercado-6a9674152/>

- Kenia Chang He
 - kchan065@fiu.edu
 - <https://github.com/keniachang>
 - <https://www.linkedin.com/in/kenia-chang-he-1889621a9/>
- Yuyang Leng
 - yleng003@fiu.edu
 - <https://github.com/YuyangLeng>
 - <https://www.linkedin.com/in/yuyang-leng-1832241a8/>