



NetIDE

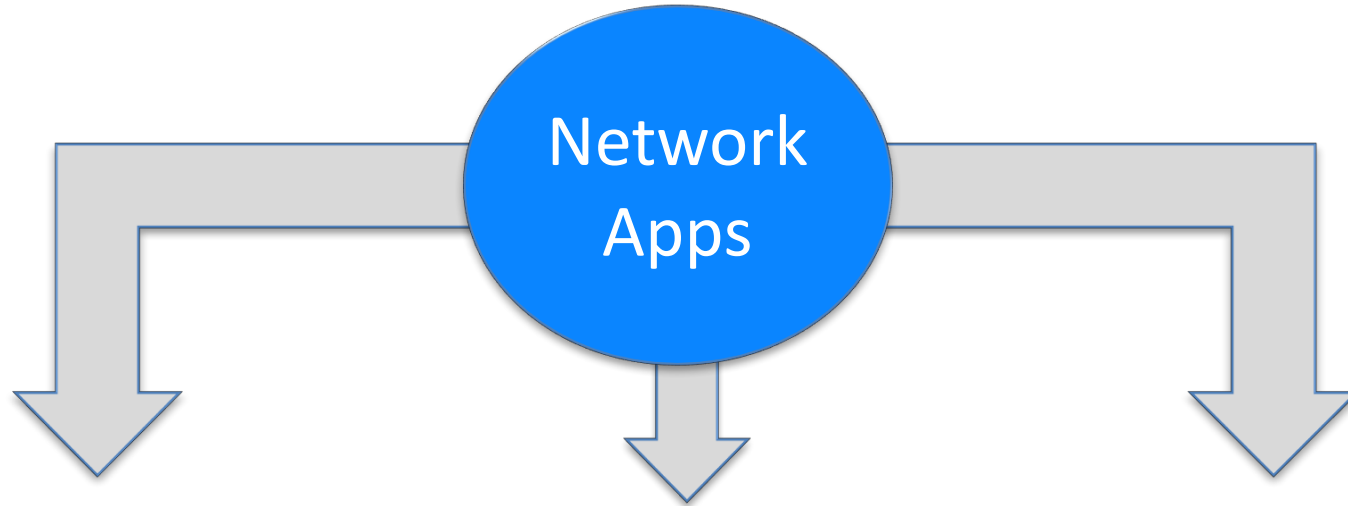
**Develop, Deploy and Deliver with
NetIDE: An Integrated Service Level
Network Programming Framework**

Pedro A, Aranda (TID)



This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619543

Challenges for the current SDN landscape



Can't easily port them:

You implement for Controller X, you can't make same code run on Controller Y

Can't easily combine them:

You can't run an LB app together with an FW app on top of the same network

Can't easily debug them:

Only few SW development tools are available for SDN, in most cases controller-specific

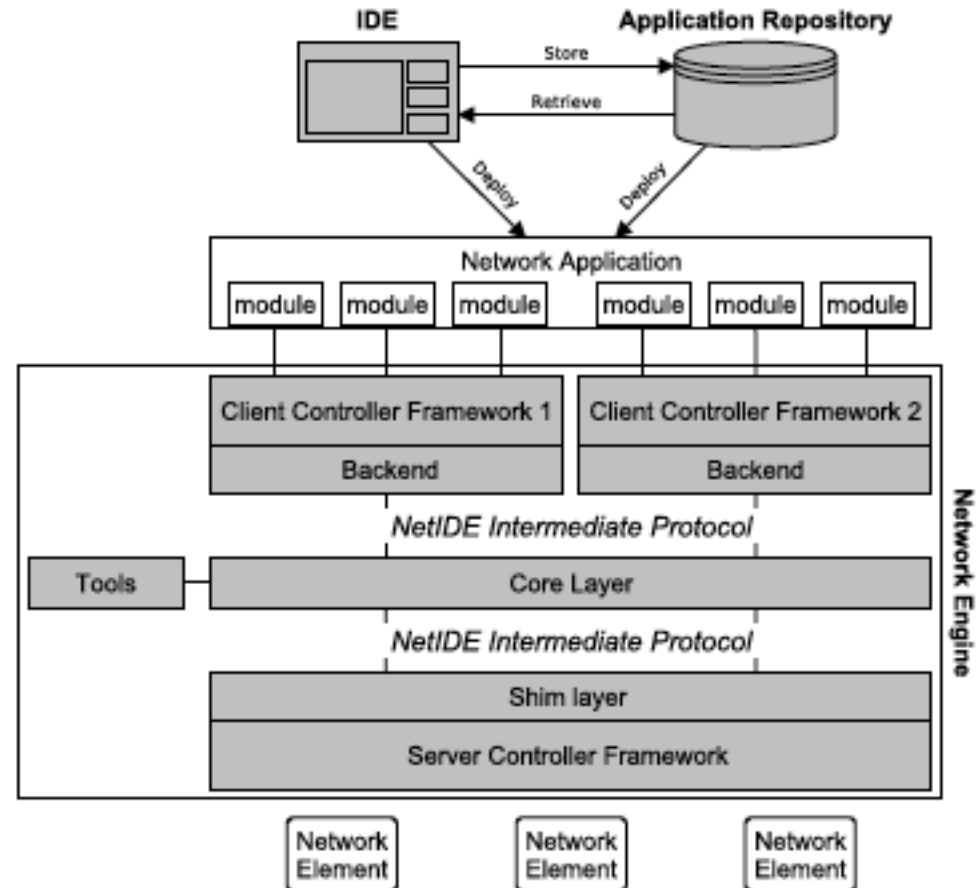
The NetIDE Framework

NetIDE aims at supporting the whole development lifecycle of network apps in a platform-independent fashion:

- * Integrated SDN development environment
- * Covering the full lifetime of SDN applications
- * It brings all the goodies of Software Design and Development to Networking:
 - * Platform independence
 - * Code re-usability
 - * Developer tools (debugger, profiler, logger, etc.)

The Network Engine

- * Client/Server SDN controller paradigm of ONF
- * **Network Application's** modules are given the runtime environment they expect in the client controller
- * **Multi-controller** support (ONOS, OpenDaylight, Ryu, Floodlight, ...)
- * **Backend:** southbound plugin
- * **Core Layer:** provides a controller-independent means to resolve conflicts between apps, interfaces with the tools
- * **Shim Layer:** northbound plugin

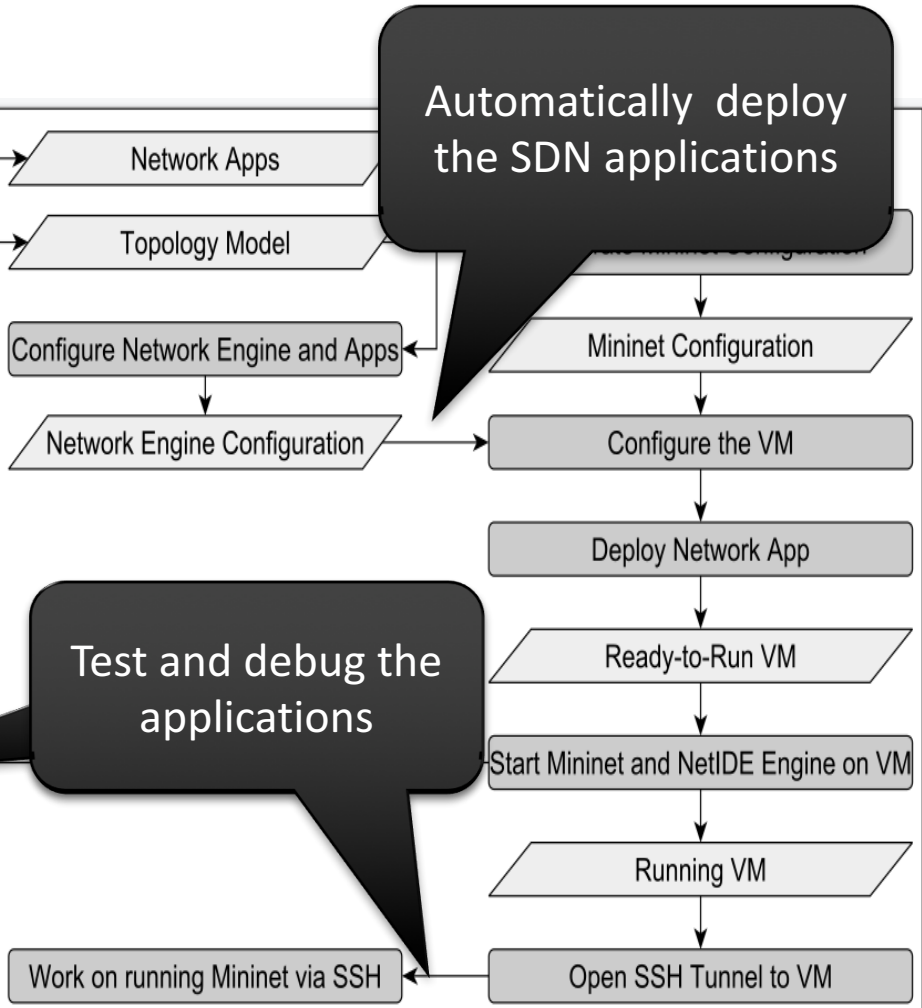


Develop, Deploy and Test

```

1
2
3 from ryu.base import app_manager
4 from ryu.controller import ofp_event
5 from ryu.controller.handler import M
6 from ryu.controller.handler import s
7
8 class L2Switch(app_manager.RyuApp):
9     def __init__(self, *args, **kwar
10         super(L2Switch, self).__ini
11
12     @set_ev_cls(ofp_event.EventOF
13     def packet_in_handler(self,
14         msg = ev.msg
15         dp = msg.datapath
16         ofp = dp.ofproto
17         ofp_parser = dp.ofpr
18
19         actions = [ofp_par
20         out = ofp_parser.
21         datapath=dp,
22         actions=act
                
```

Develop the code and configure the topology



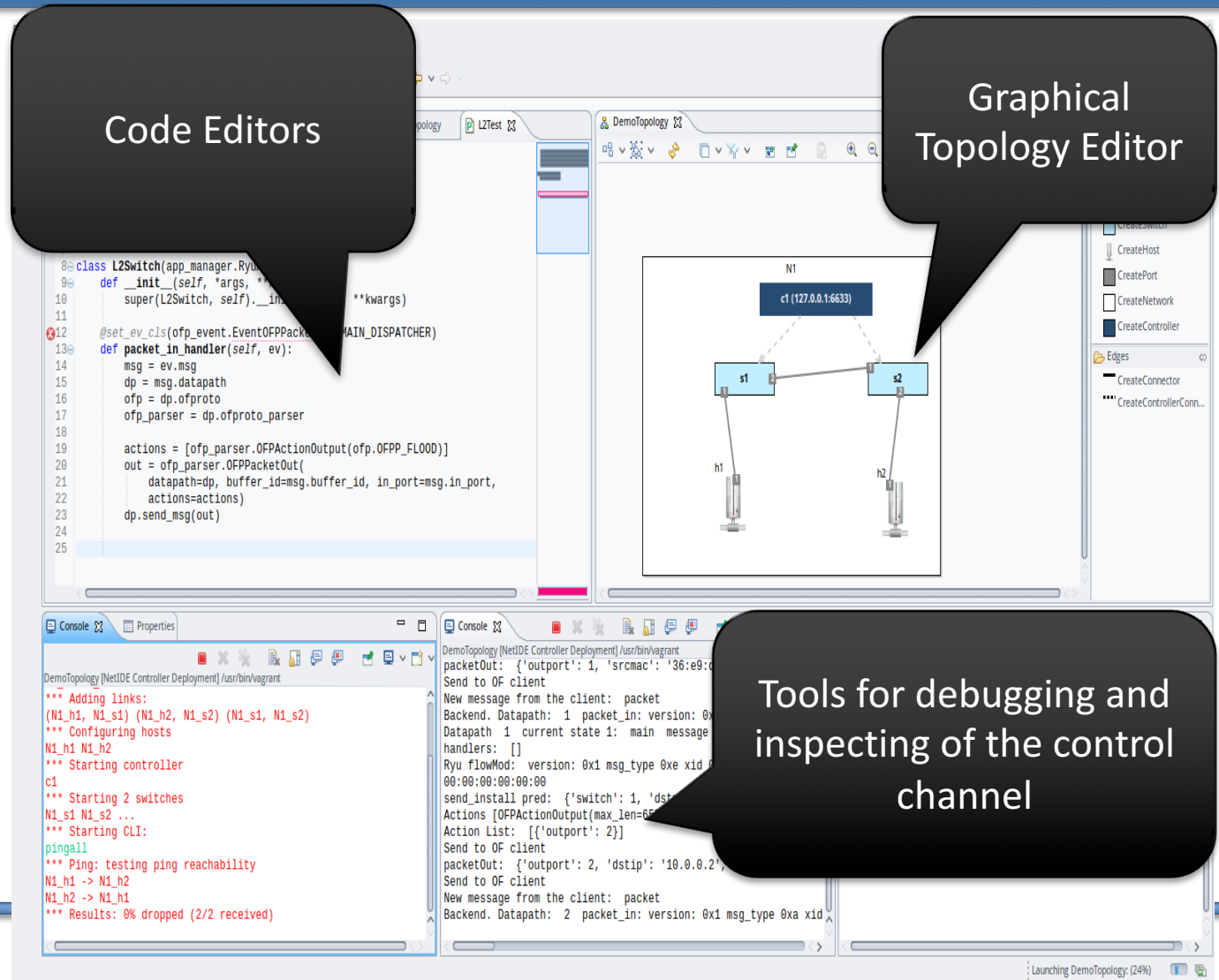
Automatically deploy the SDN applications

Test and debug the applications

Work on running Mininet via SSH

The Integrated Development Environment

- * Code Editors (PyDev, CDT, Java)
- * Topology editor
- * Interface with the Network Engine and tools
- * Access underlying network
- * Access to the Mininet CLI
- * Network Elements



The screenshot displays the NetIDE IDE interface. On the left, a code editor shows Python code for an L2Switch class. On the right, a graphical topology editor shows a network diagram with two switches (s1, s2) connected to hosts (h1, h2) and a controller (c1). At the bottom, a console window shows the execution of network setup commands and a ping test.

Code Editors

```

class L2Switch(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(L2Switch, self).__init__(*args, **kwargs)

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def packet_in_handler(self, ev):
        msg = ev.msg
        dp = msg.datapath
        ofp = dp.ofproto
        ofp_parser = dp.ofproto_parser

        actions = [ofp_parser.OFPACTIONOutput(ofp.OFPP_FLOOD)]
        out = ofp_parser.OFPPacketOut(
            datapath=dp, buffer_id=msg.buffer_id, in_port=msg.in_port,
            actions=actions)
        dp.send_msg(out)
    
```

Graphical Topology Editor

The topology editor shows a network diagram with the following elements:

- Controller: c1 (127.0.0.1:6633)
- Switches: s1, s2
- Hosts: h1, h2

Tools for debugging and inspecting of the control channel

```

DemoTopology [NetIDE Controller Deployment] /usr/bin/vagrant
*** Adding links:
(N1_h1, N1_s1) (N1_h2, N1_s2) (N1_s1, N1_s2)
*** Configuring hosts
N1_h1 N1_h2
*** Starting controller
c1
*** Starting 2 switches
N1_s1 N1_s2 ...
*** Starting CLI:
pingall
*** Ping: testing ping reachability
N1_h1 -> N1_h2
N1_h2 -> N1_h1
*** Results: 0% dropped (2/2 received)
    
```

```

DemoTopology [NetIDE Controller Deployment] /usr/bin/vagrant
packetOut: {'outport': 1, 'srcmac': '36:e9:00:00:00:00'}
Send to OF client
New message from the client: packet
Backend. Datapath: 1 packet_in: version: 0x1 msg_type 0xa xid: 0x0
Datapath 1 current state 1: main message
handlers: []
Ryu flowMod: version: 0x1 msg_type 0xe xid: 0x0
send_install pred: {'switch': 1, 'dstip': '10.0.0.2'}
Actions [OFPACTIONOutput(max_len=65535)]
Action List: [{'outport': 2}]
Send to OF client
packetOut: {'outport': 2, 'dstip': '10.0.0.2'}
Send to OF client
New message from the client: packet
Backend. Datapath: 2 packet_in: version: 0x1 msg_type 0xa xid: 0x0
    
```

Developer tools

Enabling the developer to systematically test, profile, and tune their Network App

- * **Logger:** tracing capabilities to judge the performance of the deployed Network App
- * **Garbage Collector:** Cleans the switches' memory from unused flow rules
- * **Model Checker:** systematically exercises app behaviour and flag actions that lead to violations of the desired safety properties
- * **Profiler:** judging the impact of network failures on the Network App behaviour
- * **Debugger:** supports debug of packet processing (OFReplay, packet inspection and flow table checking)

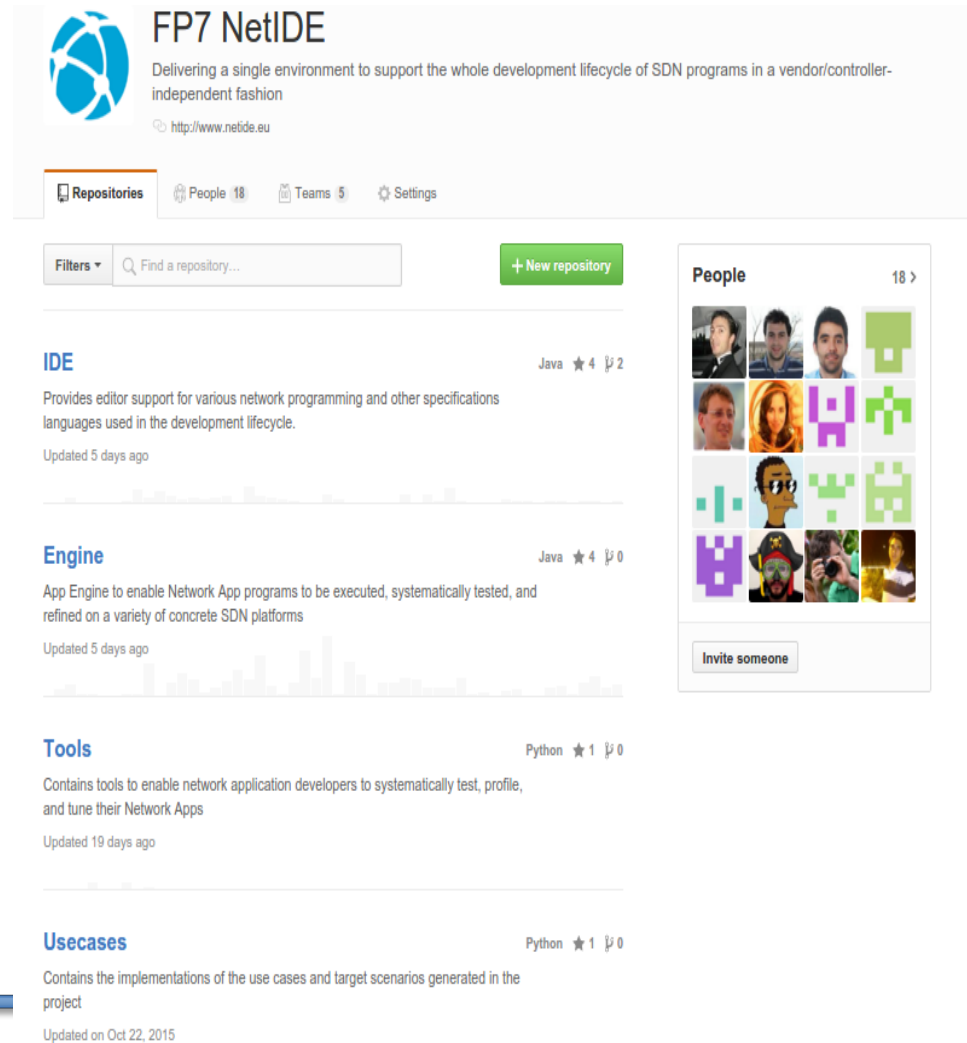
Try our code?

We assure survival of NetIDE results by contributing them to different FOSS projects

Source code of IDE, Network Engine and Tools are publicly available on Github under **Eclipse Public License v1.0**

Usecases contains implementations of target scenarios that validate the NetIDE framework.

<https://github.com/fp7-netide>



The screenshot shows the GitHub repository page for FP7 NetIDE. At the top, the repository name "FP7 NetIDE" is displayed with a description: "Delivering a single environment to support the whole development lifecycle of SDN programs in a vendor/controller-independent fashion" and the website "http://www.netide.eu". Below this, there are navigation tabs for "Repositories", "People 18", "Teams 5", and "Settings". A search bar with "Find a repository..." and a "+ New repository" button are visible. The main content area lists three repositories:

- IDE** (Java): "Provides editor support for various network programming and other specifications languages used in the development lifecycle." Updated 5 days ago. 4 stars, 2 forks.
- Engine** (Java): "App Engine to enable Network App programs to be executed, systematically tested, and refined on a variety of concrete SDN platforms." Updated 5 days ago. 4 stars, 0 forks.
- Tools** (Python): "Contains tools to enable network application developers to systematically test, profile, and tune their Network Apps." Updated 19 days ago. 1 star, 0 forks.
- Usecases** (Python): "Contains the implementations of the use cases and target scenarios generated in the project." Updated on Oct 22, 2015. 1 star, 0 forks.

On the right side, there is a "People" section showing 18 contributors with their profile pictures and a grid of avatars. An "Invite someone" button is located below the grid.

Thank You!