Deep Dive into Neutron

by Yong Sheng Gong

UnitedStack

caveats

- developers oriented
  - many codes and UML diagrams
- the snapshot of current neutron code
  - evolution of neutron codes will obsolete some contents of this presentation

Coming sessions about Neutron

- Load balancing in neutron

Thursday November 7, 2013 4:30pm - 5:10pm, SkyCity Grand Ballroom C (SkyCity Marriott Hotel)

- How to Write a Neutron Plugin, If You Really Need to

Thursday November 7, 2013 5:20pm - 6:00pm ,SkyCity Grand Ballroom C (SkyCity Marriott Hotel)

- OpenStack Neutron Modular Layer 2 Plugin Deep Dive

Friday November 8, 2013 11:00am - 11:40am, Expo Breakout Room 2 (AsiaWorld-Expo)

- Neutron Hybrid Deployment and Performance Analysis

Friday November 8, 2013 1:30pm - 2:10pm, Expo Breakout Room 2 (AsiaWorld-Expo)

- Neutron Network Namespaces and IPtables: Technical Deep Dive

Friday November 8, 2013 4:10pm - 4:50pm, Expo Breakout Room 2 (AsiaWorld-Expo)

Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

related skills

- WSGI

WSGI is the Web Server Gateway Interface. It is a specification for web servers and application servers to communicate with web applications.

- paste deploy

Paste Deployment is a system for finding and configuring WSGI applications and servers. The primary interaction with Paste Deploy is through its configuration files.

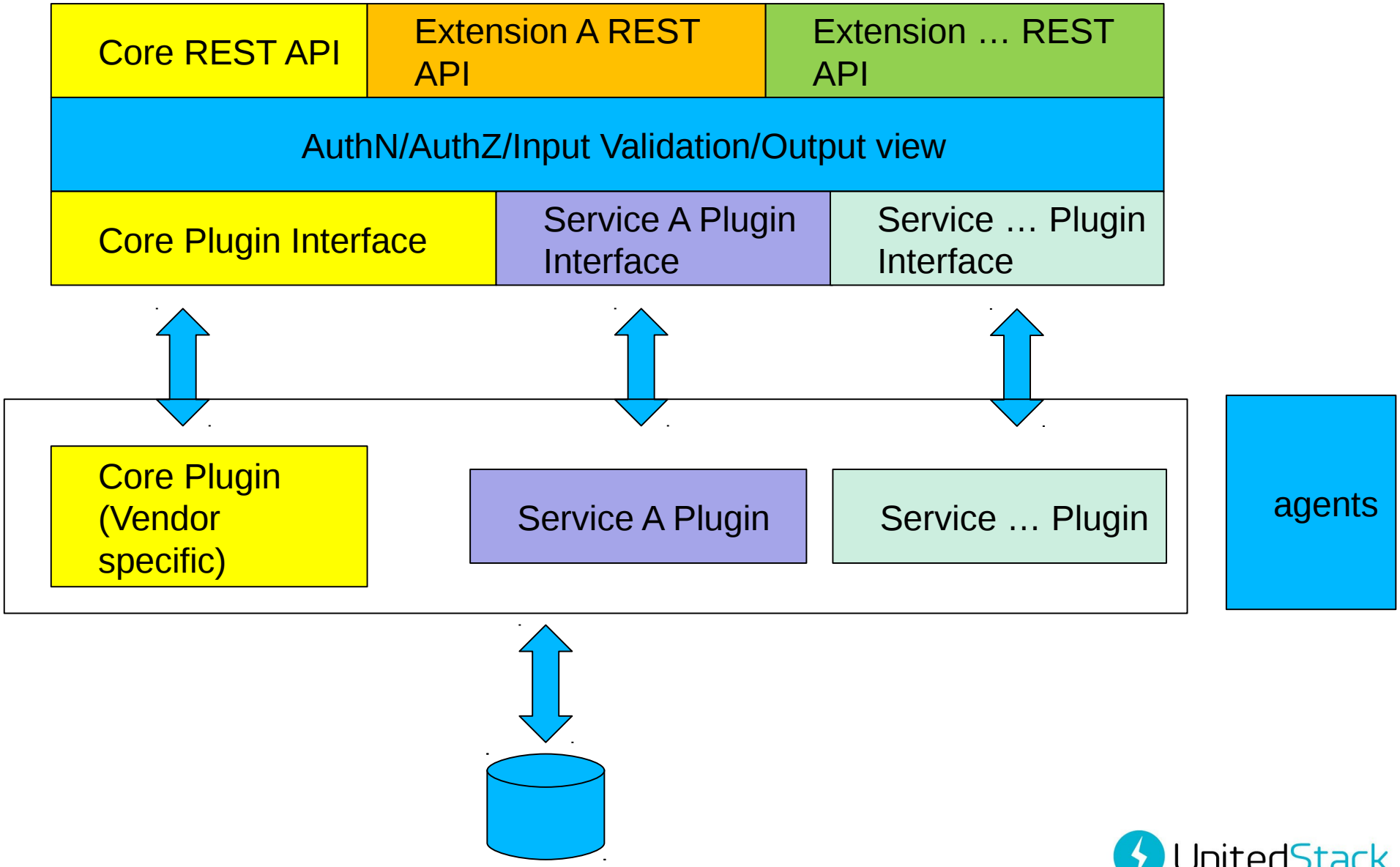
- Python Routes

Routes is a Python re-implementation of the Rails routes system for mapping URLs to application actions, and conversely to generate URLs. Routes makes it easy to create pretty and concise URLs that are RESTful with little effort.

- peCan

Will we change to pecan? see design summit session Neutron API Framework Replacement

Layer diagram of Neutron server



paste application and filters

```
[composite:neutron]
use = egg:Paste#urlmap
/: neutronversions
/v2.0: neutronapi_v2_0

[composite:neutronapi_v2_0]
use = call:neutron.auth:pipeline_factory
keystone = authtoken keystonecontext extensions neutronapiapp_v2_0

[filter:keystonecontext]
paste.filter_factory = neutron.auth:NeutronKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory

[filter:extensions]
paste.filter_factory =
neutron.api.extensions:plugin_aware_extension_middleware_factory

[app:neutronversions]
paste.app_factory = neutron.api.versions:Versions.factory

[app:neutronapiapp_v2_0]
paste.app_factory = neutron.api.v2.router:APIRouter.factory
```

main entry point

neutron/server/__init__.py: main()

1.config.parse(sys.argv[1:]) ← --config-file neutron.conf --config-file xxx.ini

2.neutron/common/config.py:load_paste_app("neutron")

2.1 neutron/auth.py:pipeline_factory()

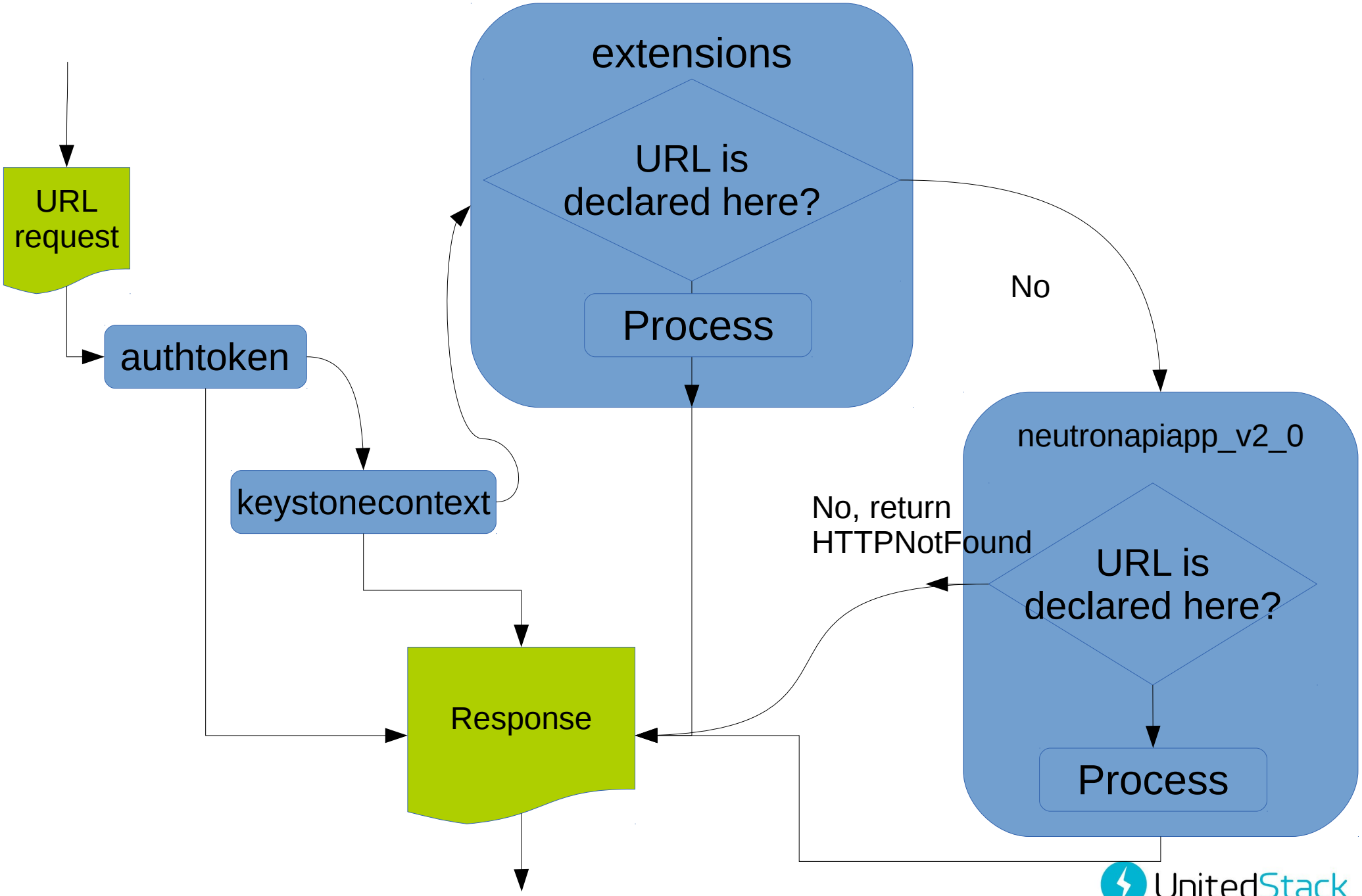
2.1.1 neutron/api/v2/router.py:APIRouter.factory()

2.1.2 neutron/api/extensions.py:
plugin_aware_extension_middleware_factory()

2.1.3 neutron.auth:NeutronKeystoneContext.factory()

2.1.4 keystoneclient.middleware.auth_token:filter_factory()

filters and application pipeline



neutronapiapp_v2_0: load plugins

neutron/api/v2/router.py:APIRouter.factory()

1. __init__()

   1.1 plugin = manager.NeutronManager.get_plugin()

   1.1.1 neutron/manager.py:__init__()

   A `1.1.1.1 create core plugin instance`

   B `1.1.1.2 neutron/manager.py:_load_service_plugins()`

neutron.conf:
service_plugins = ...
core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin

```
NeutronManager
:service_plugins =
{“CORE”: ml2_plugin,
 "LOADBALANCER":xxx,
 ...}
```

UnitedStack

what are plugins and extensions

- extensions are about resources and the actions on them

```
@classmethod
def get_resources(cls):
    for resource_name in ['router', 'floatingip']:
        ...
        controller = base.create_resource(
            collection_name, resource_name, plugin...)

        ex = ResourceExtension(collection_name, controller,
member_actions...)
```

- plugins are used to support the resources

```
supported_extension_aliases = ["router", "ext-gw-mode",
                               "extraroute",
"l3_agent_scheduler"]
def update_router(self, context, id, router):
def get_router(self, context, id, fields=None):
```

neutronapiapp_v2_0: load extensions

```
neutron/api/v2/router.py:APIRouter.factory()
  1. __init__()
      1.1 plugin = manager.NeutronManager.get_plugin()
      1.2 extensions.PluginAwareExtensionManager.get_instance()

        1.2.1 extensions.py:get_extensions_path()

        1.2.2 PluginAwareExtensionManager.__init__(paths, plugins)
          1.2.2.1 _load_all_extensions()
          for each path in paths
              _load_all_extensions_from_path(path
          )    add_extension(ext)
                   _check_extension(ext)
```

A

B

neutron standard extension plus ones specified by api_extensions_path= in neutron.conf

check each python module name under the path, and capitalize the first letter of the module name to find the class in it, excluding the modules starting with "_".

1. check if the potential extension has implemented the needed functions
2. check if one of plugins supports it. plugin's supported_extension_aliases attribute defines what extensions it supports.

neutronapiapp_v2_0: install core resources

```
neutron/api/v2/router.py:APIRouter.factory()
  1. __init__()
      1.1 plugin = manager.NeutronManager.get_plugin()
      1.2 PluginAwareExtensionManager.get_instance()
      1.3 install core resources
```



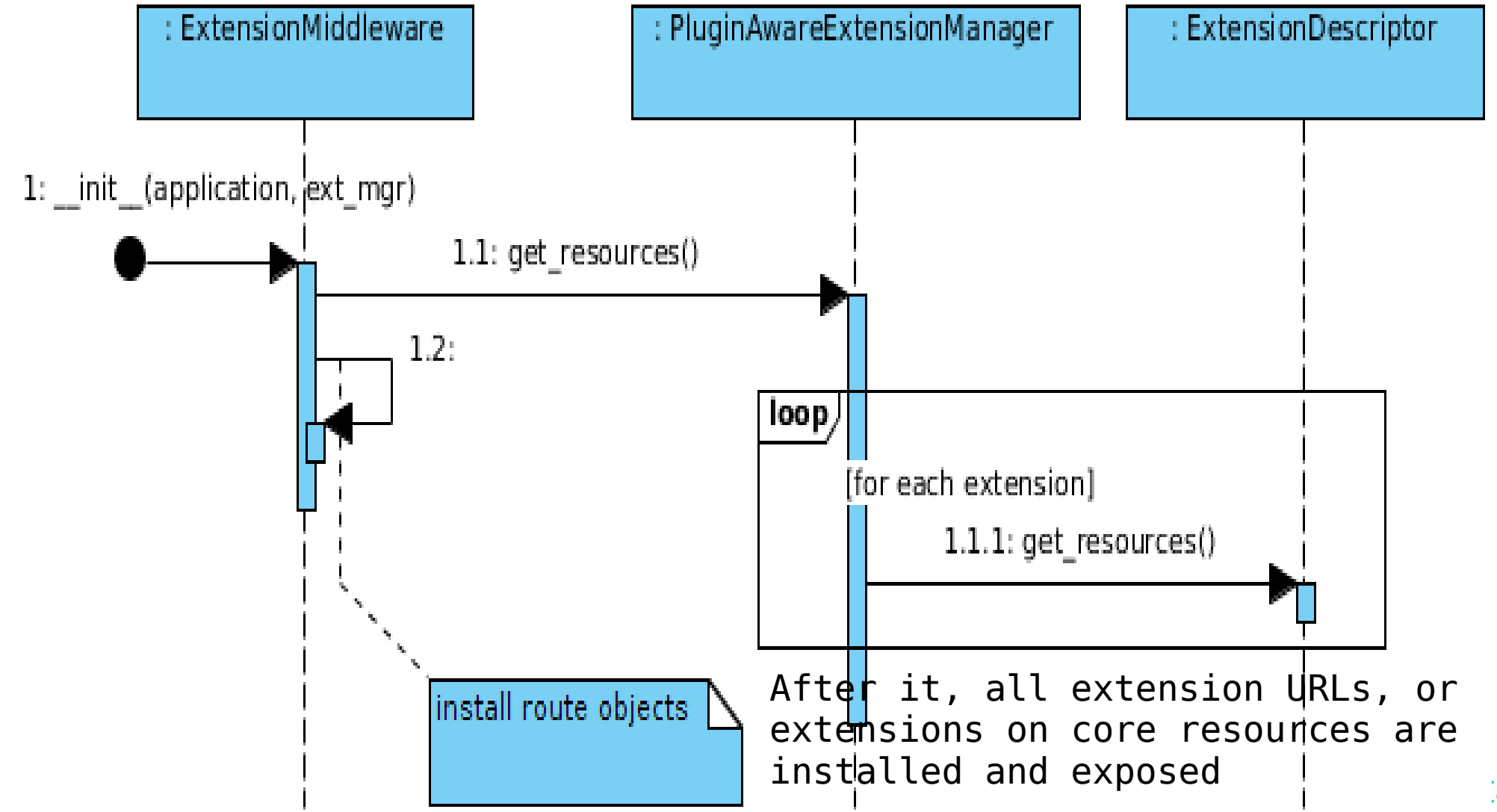
```
neutron/api/v2/router.py:
RESOURCES = {'network': 'networks',
             'subnet': 'subnets',
             'port': 'ports'}
```

After it, core resources URLs, i.e. Core Resource API, are installed and exposed.

extension filter: assemble extensions

2.1.2 neutron/api/extensions.py:plugin_aware_extension_middleware_factory()

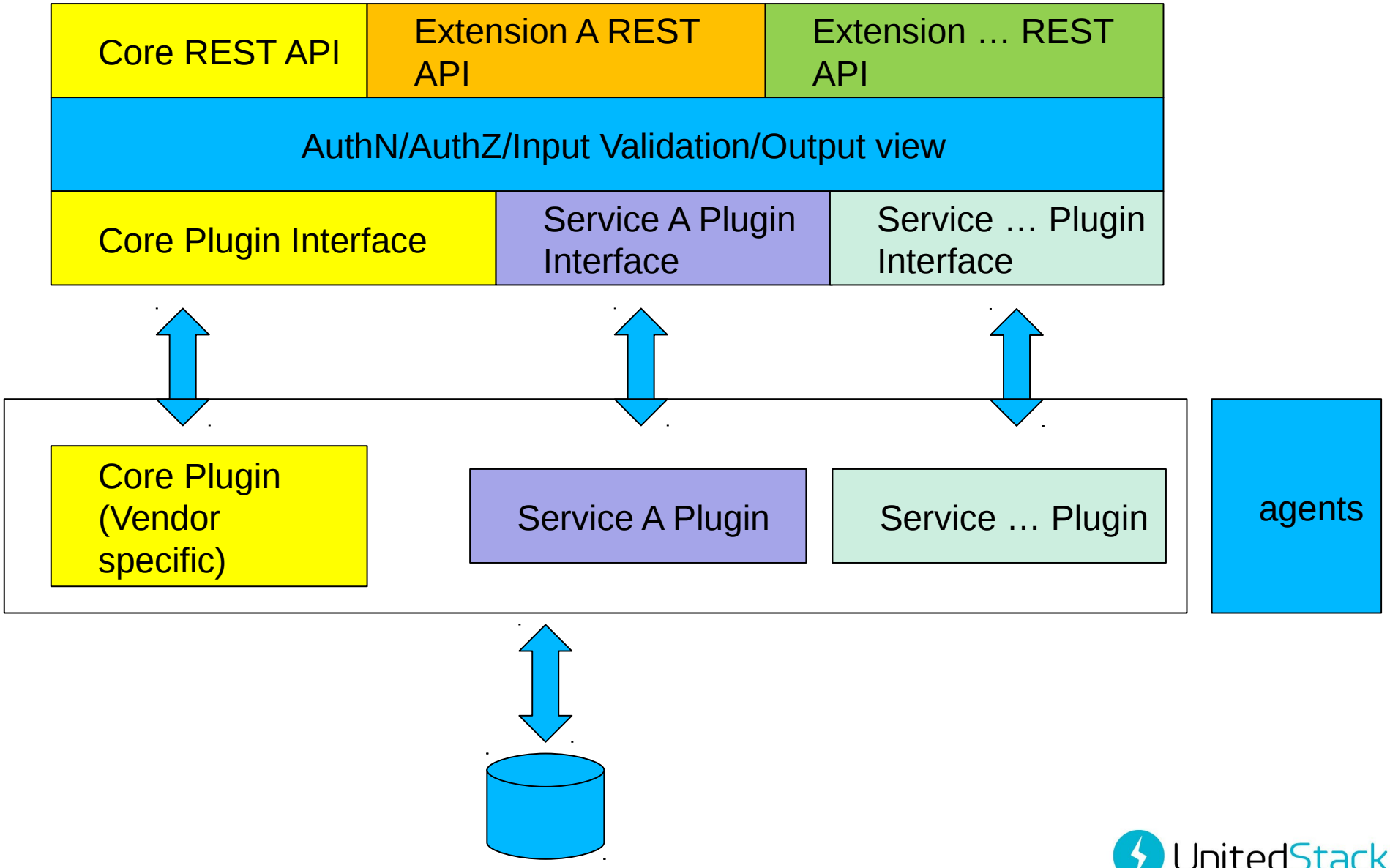
```
ext_mgr = PluginAwareExtensionManager.get_instance()
return ExtensionMiddleware(app, ext_mgr=ext_mgr)
```



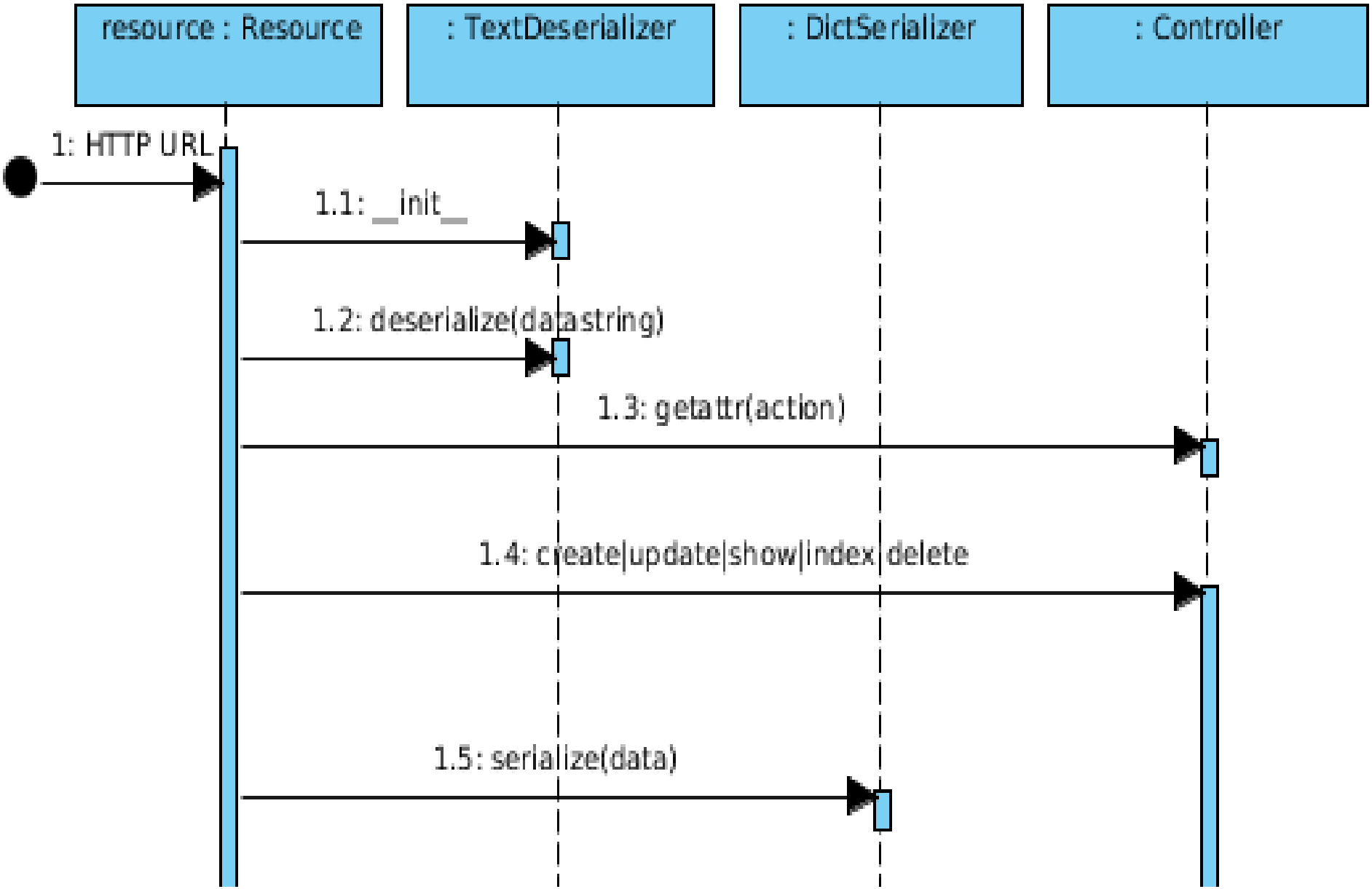
Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

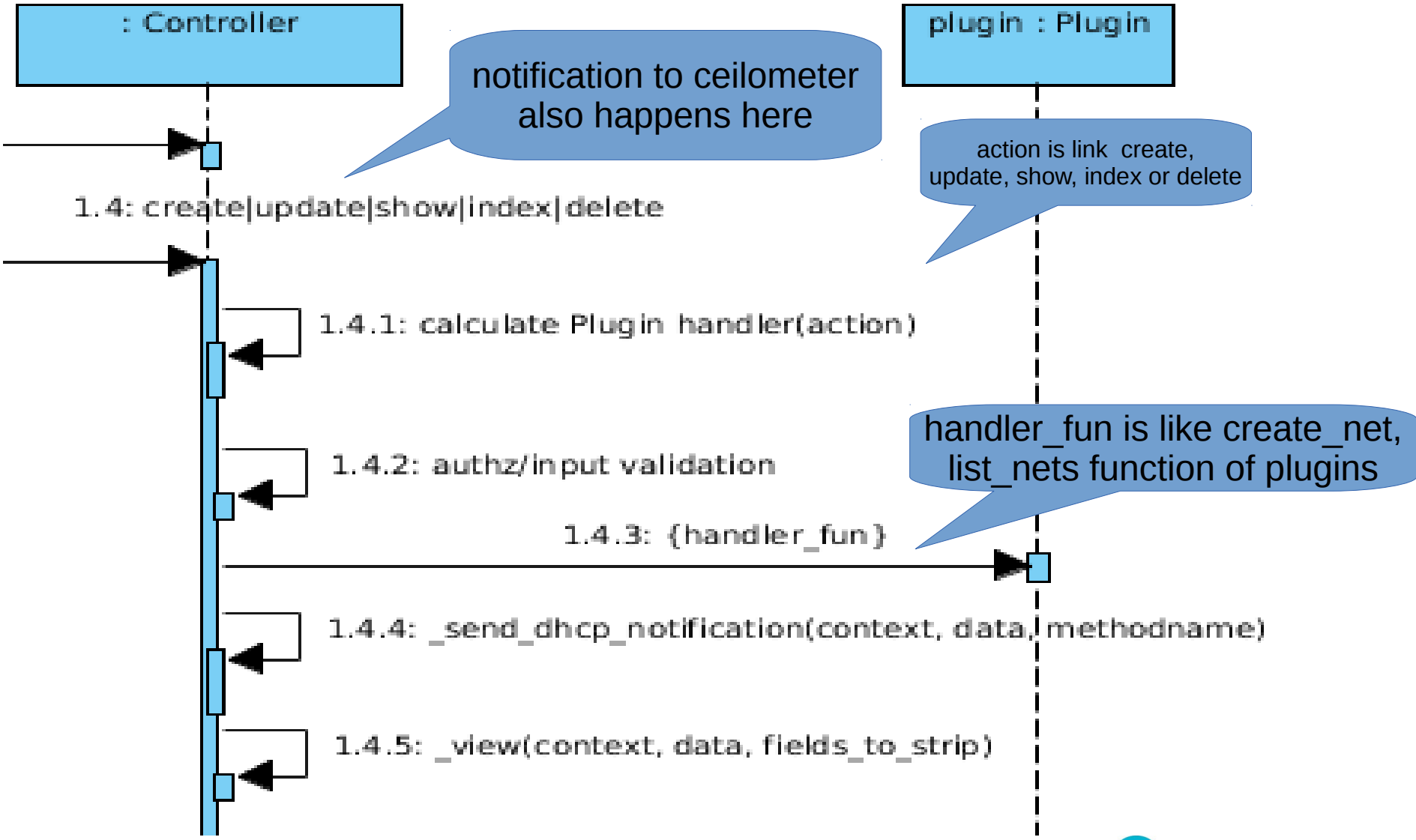
Layer diagram



URL processing (major steps)



URL processing continued



Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

ML2 Plugin

- simultaneously utilize the variety of layer 2 networking technologies found in complex real-world data centers
- It currently works with the existing openvswitch, linuxbridge, and hyperv L2 agents
- The ml2 framework is also intended to greatly simplify adding support for new L2 networking technologies
- consists of network types and mechanisms

https://wiki.openstack.org/wiki/Neutron/ML2#ML2_Drivers

Type and mechanism drivers in setup.cfg

```
neutron.ml2.type_drivers =
    flat = neutron.plugins.ml2.drivers.type_flat:FlatTypeDriver
    local = neutron.plugins.ml2.drivers.type_local:LocalTypeDriver
    vlan = neutron.plugins.ml2.drivers.type_vlan:VlanTypeDriver
    gre = neutron.plugins.ml2.drivers.type_gre:GreTypeDriver
    vxlan = neutron.plugins.ml2.drivers.type_vxlan:VxlanTypeDriver
neutron.ml2.mechanism_drivers =
    linuxbridge = neutron.plugins.ml2.drivers.mech_linuxbridge:LinuxbridgeMechanismDriver
    openvswitch = neutron.plugins.ml2.drivers.mech_openvswitch:OpenvswitchMechanismDriver
    hyperv = neutron.plugins.ml2.drivers.mech_hyperv:HypervMechanismDriver
    ncs = neutron.plugins.ml2.drivers.mechanism_ncs:NCSMechanismDriver
    arista = neutron.plugins.ml2.drivers.mech_arista.mechanism_arista:AristaDriver
    cisco_nexus = neutron.plugins.ml2.drivers.cisco.mech_cisco_nexus:CiscoNexusMechanismDriver
    l2population = neutron.plugins.ml2.drivers.l2pop.mech_driver:L2populationMechanismDriver
```

Configuration for types in ml2.ini

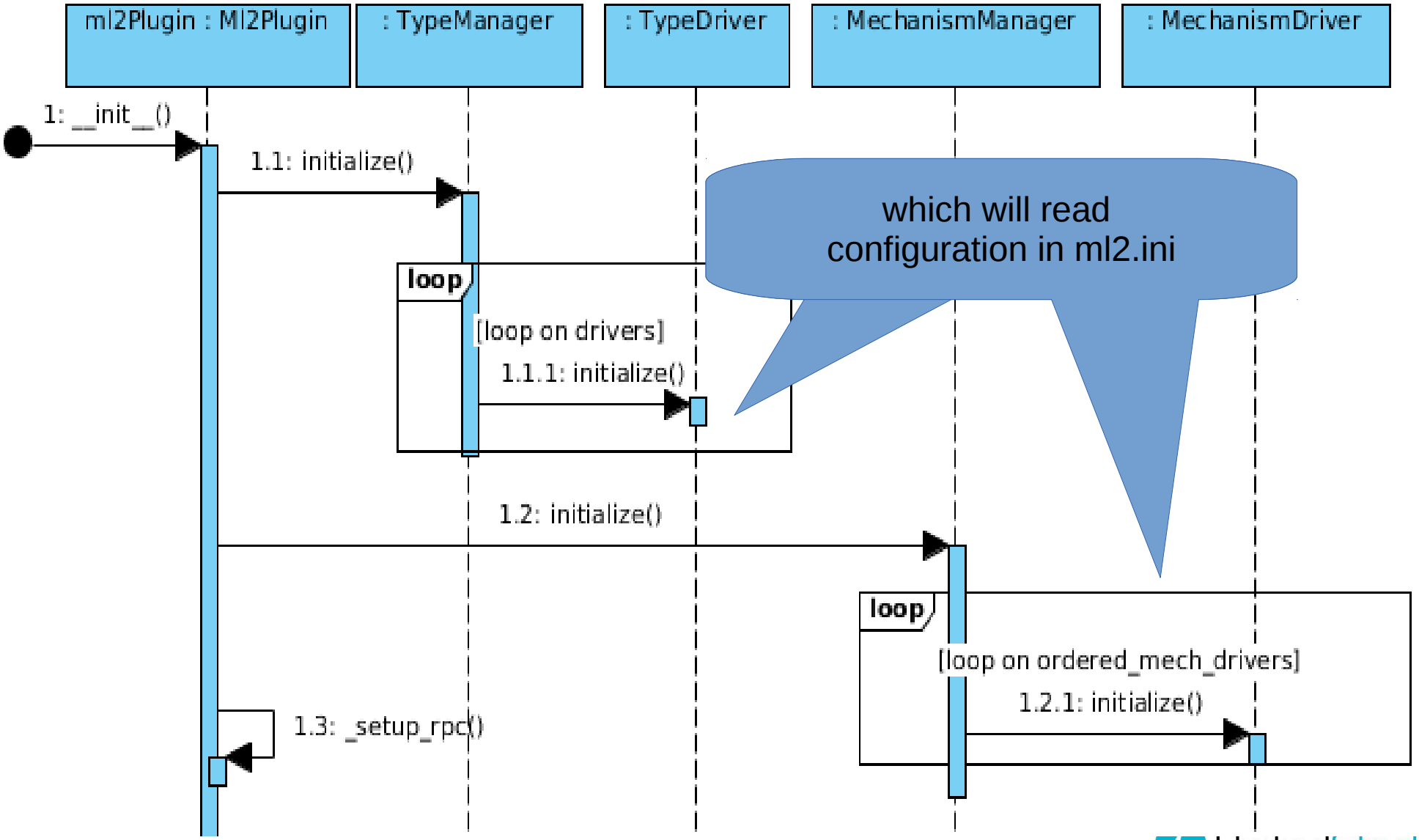
neutron-server --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/ml2.ini

```
[ml2]
type_drivers = local,flat,vlan,gre,vxlan
mechanism_drivers = openvswitch,linuxbridge
tenant_network_types = vlan,gre,vxlan

[ml2_type_flat]
flat_networks = physnet1,physnet2

[ml2_type_vlan]
network_vlan_ranges = physnet1:1000:2999,physnet2

[ml2_type_gre]
tunnel_id_ranges = 1:1000

[ml2_type_vxlan]
vni_ranges = 1001:2000
```

__init__ of ML2

neutron/manager.py:__init__()

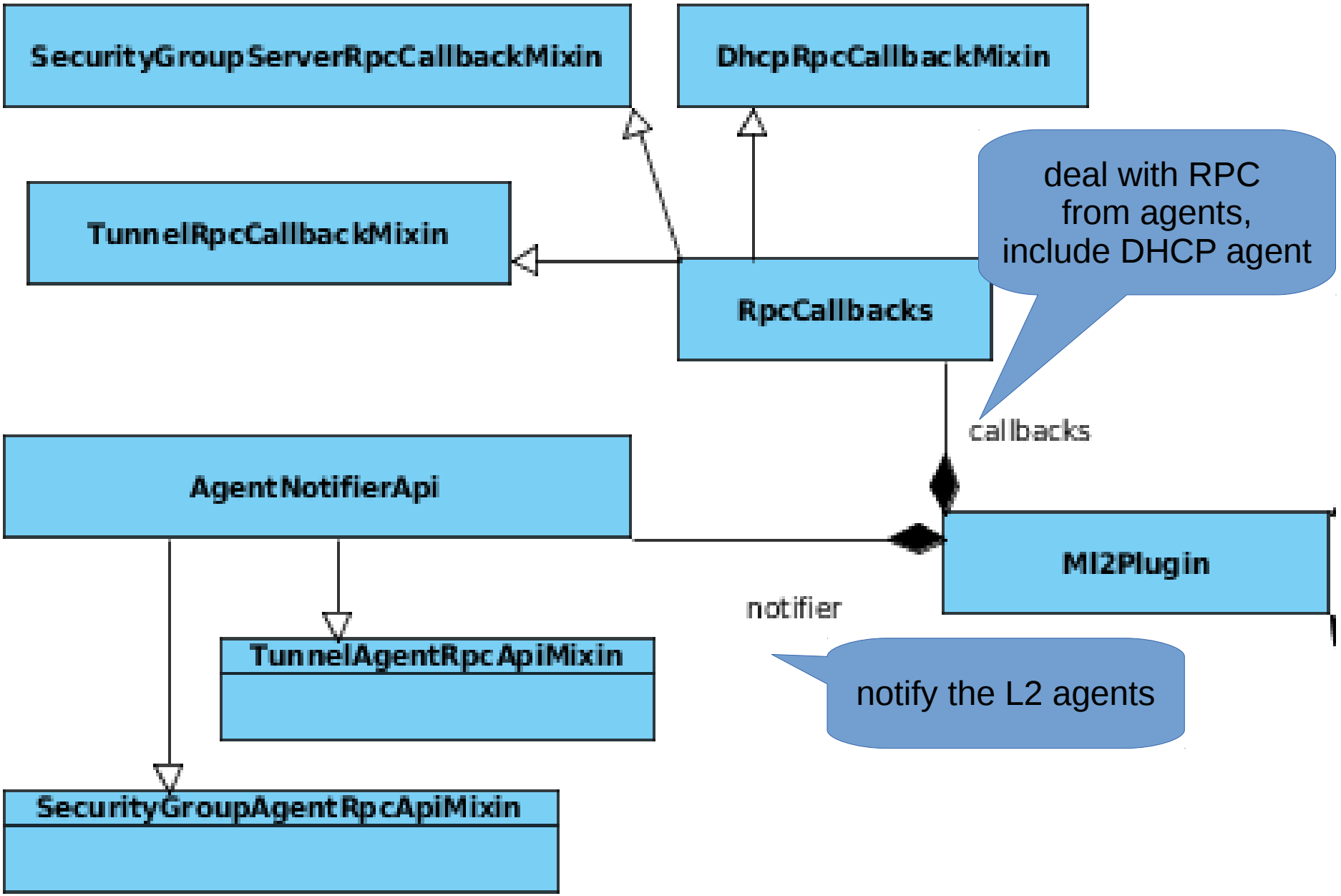
```
create core plugin instance [core_plugin=]
```

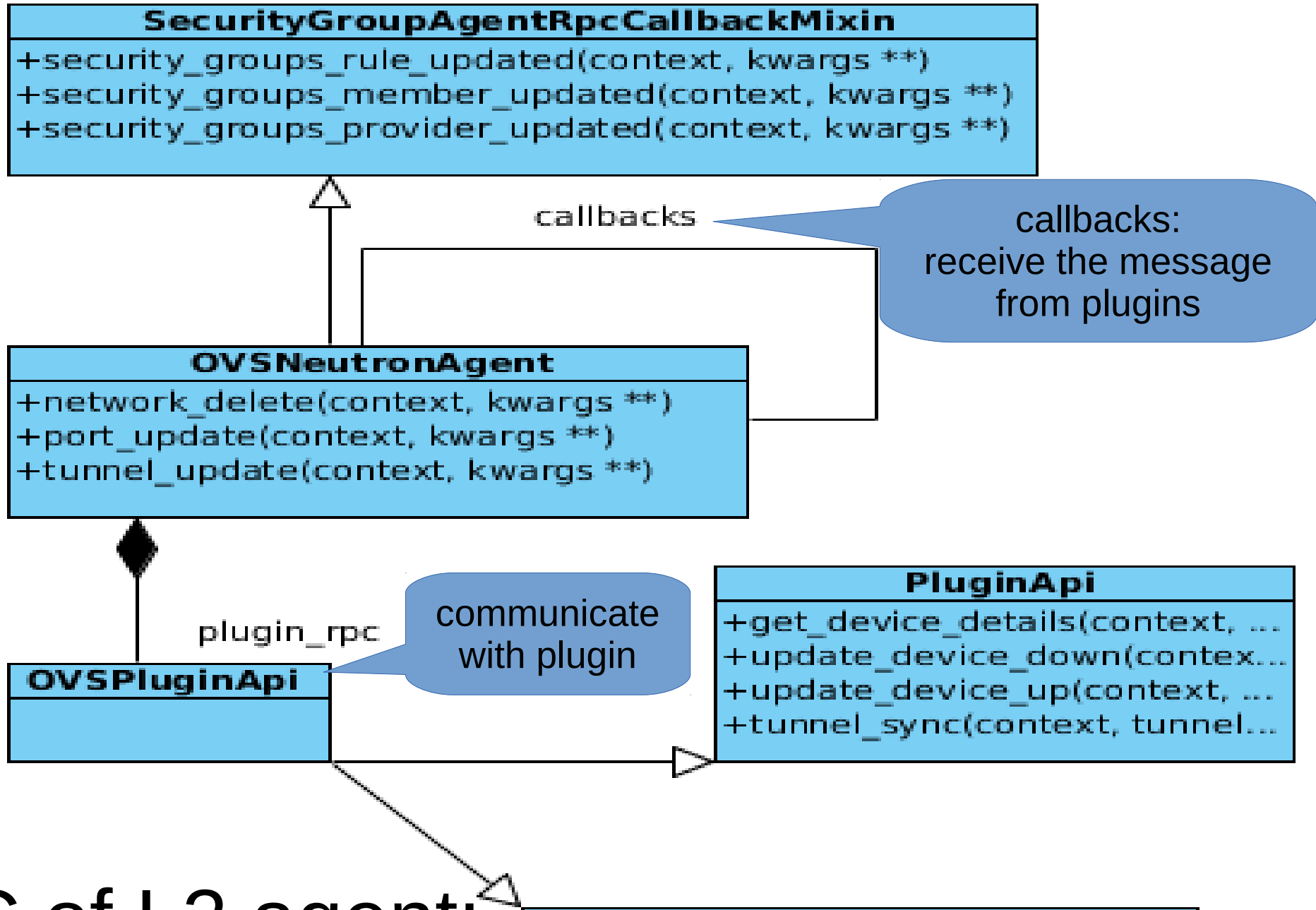


Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

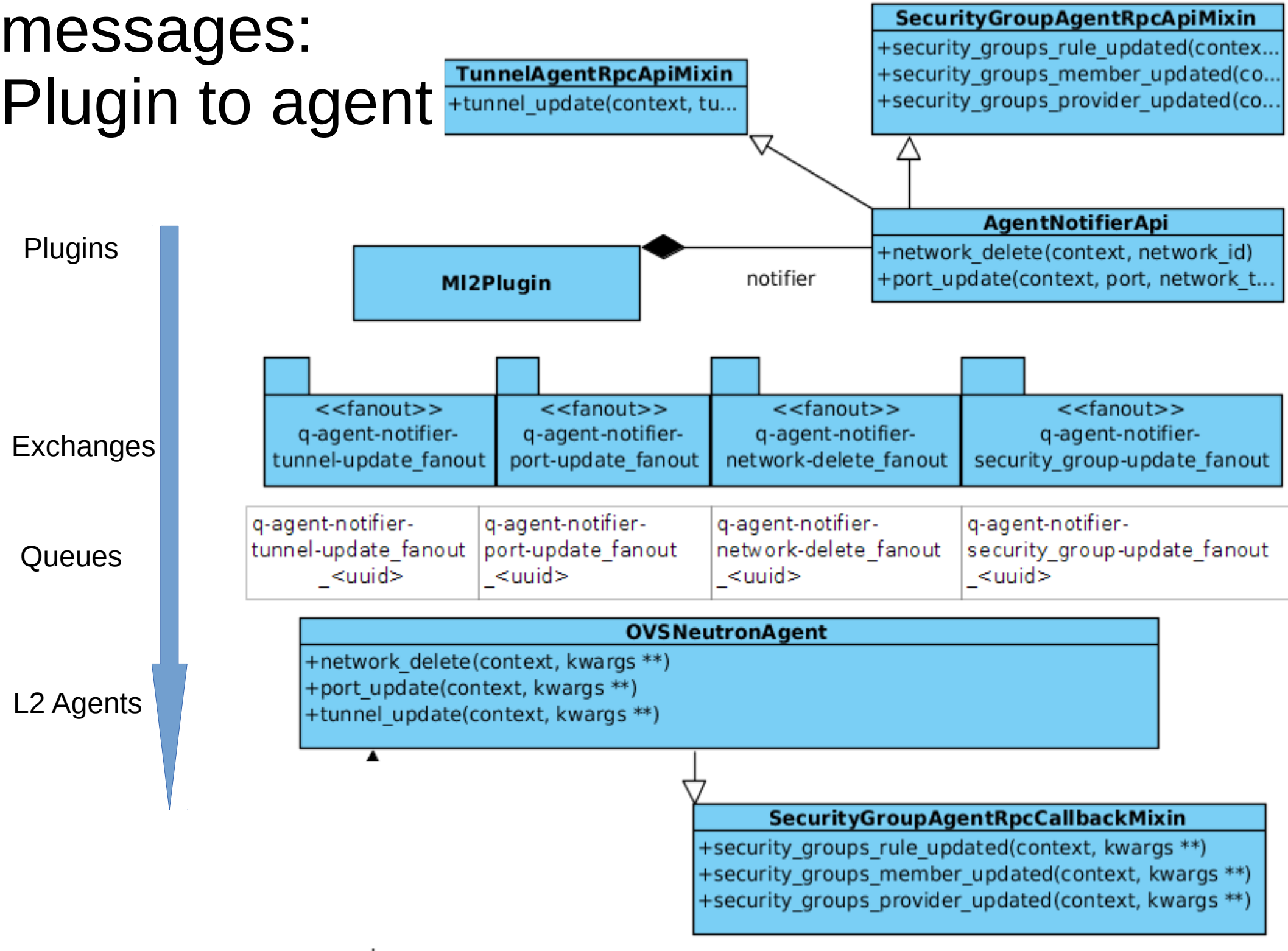
RPC structure of ML2

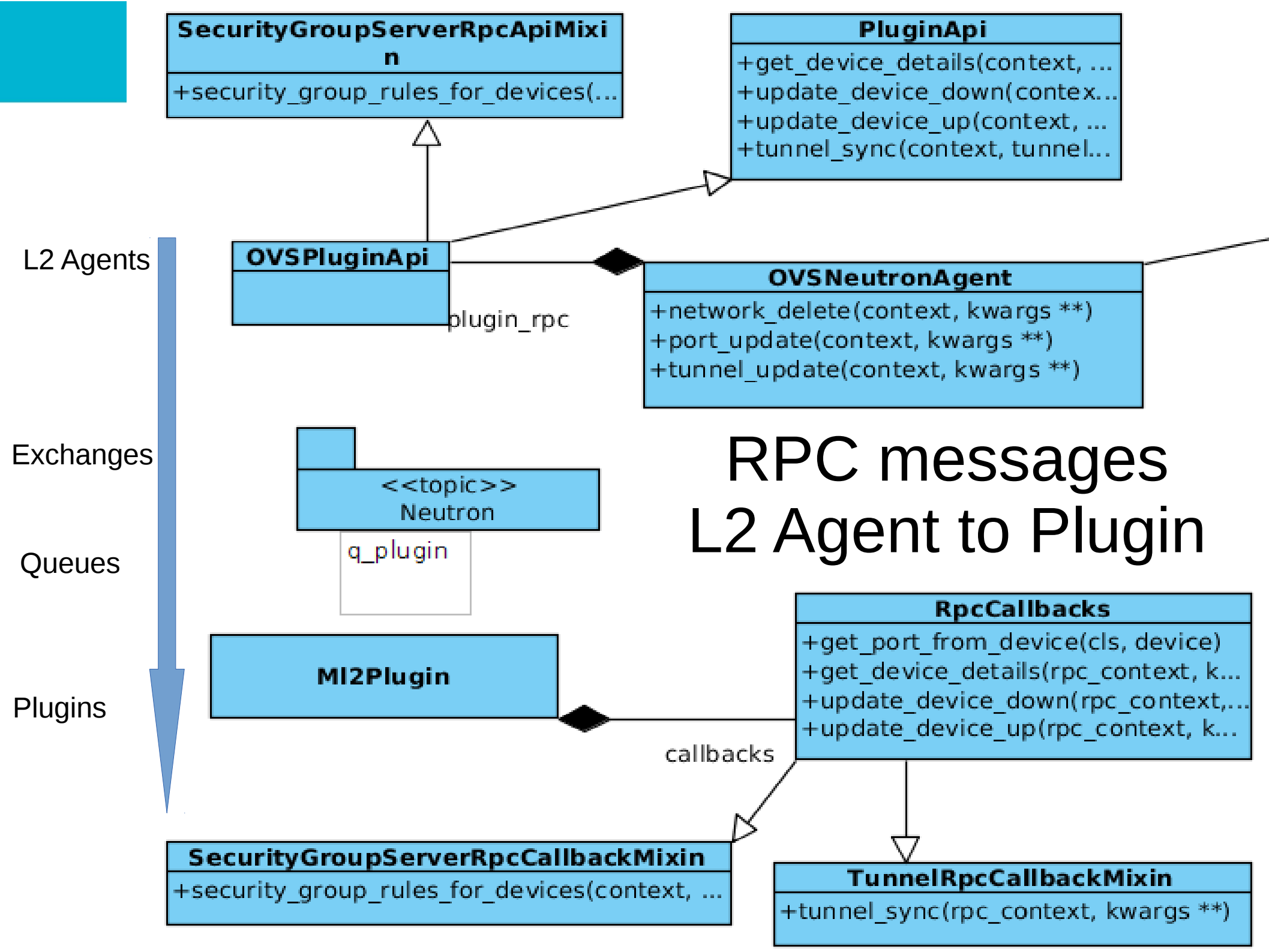




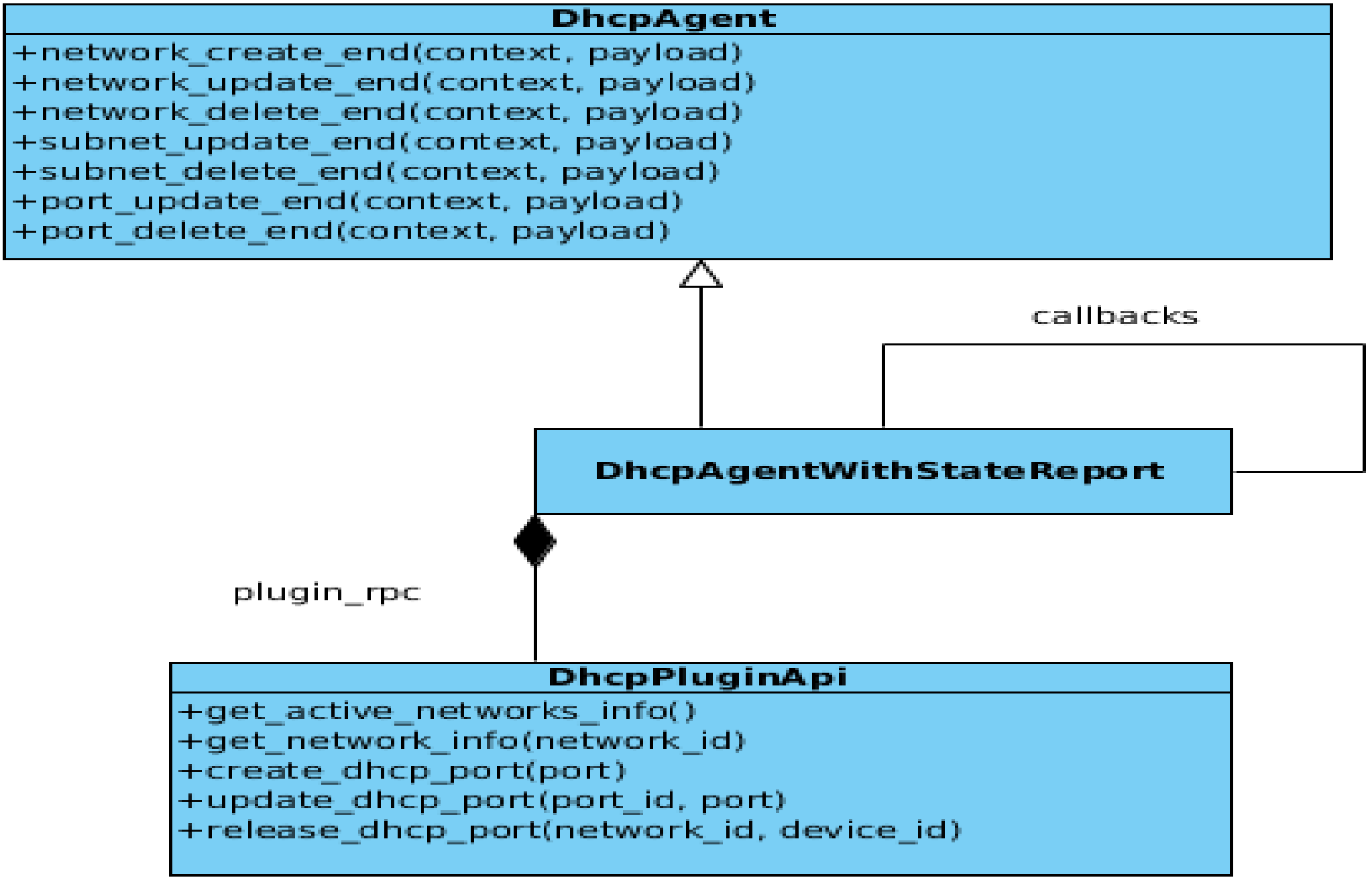
RPC of L2 agent: ovs neutron agent

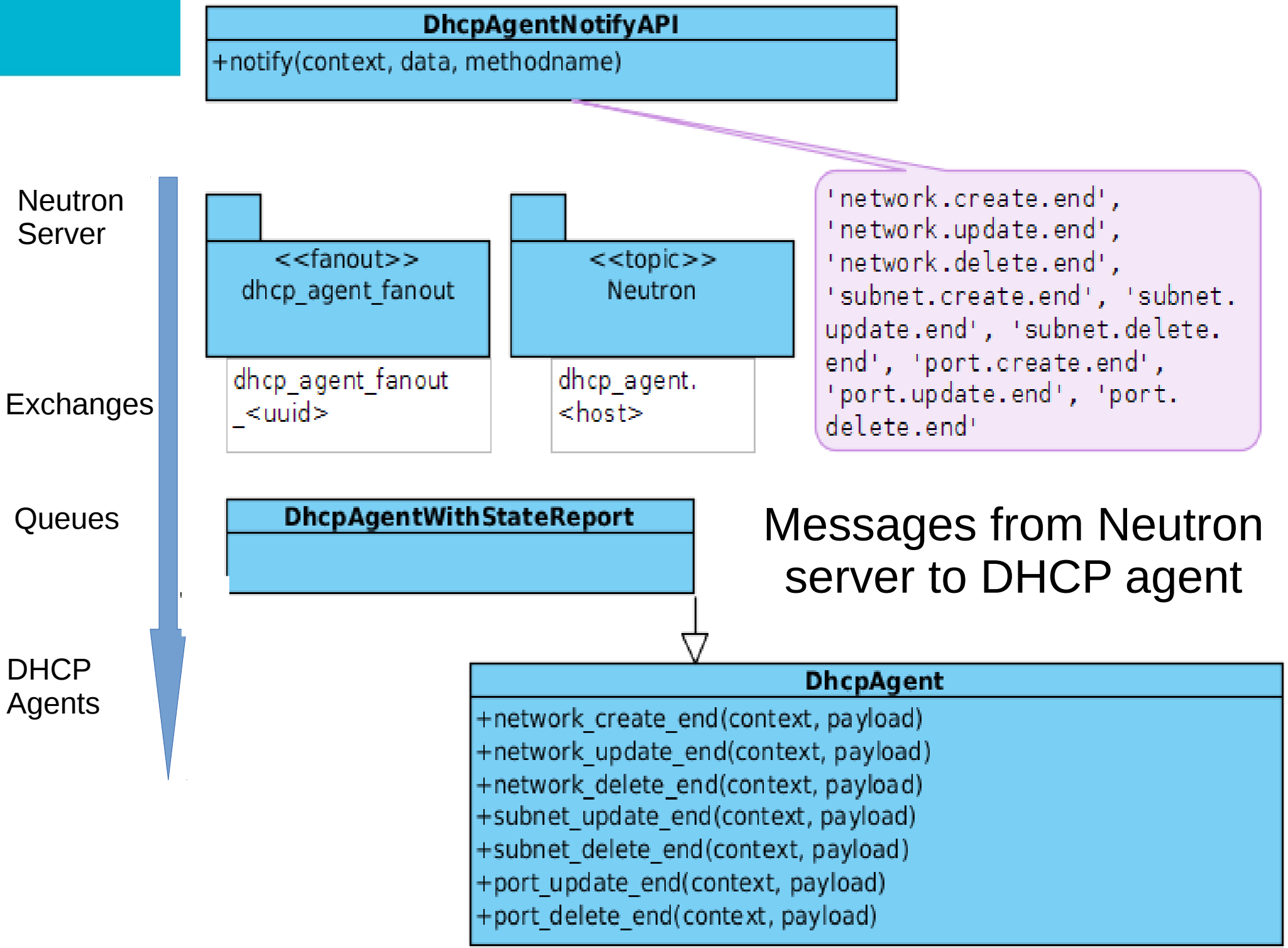
messages: Plugin to agent



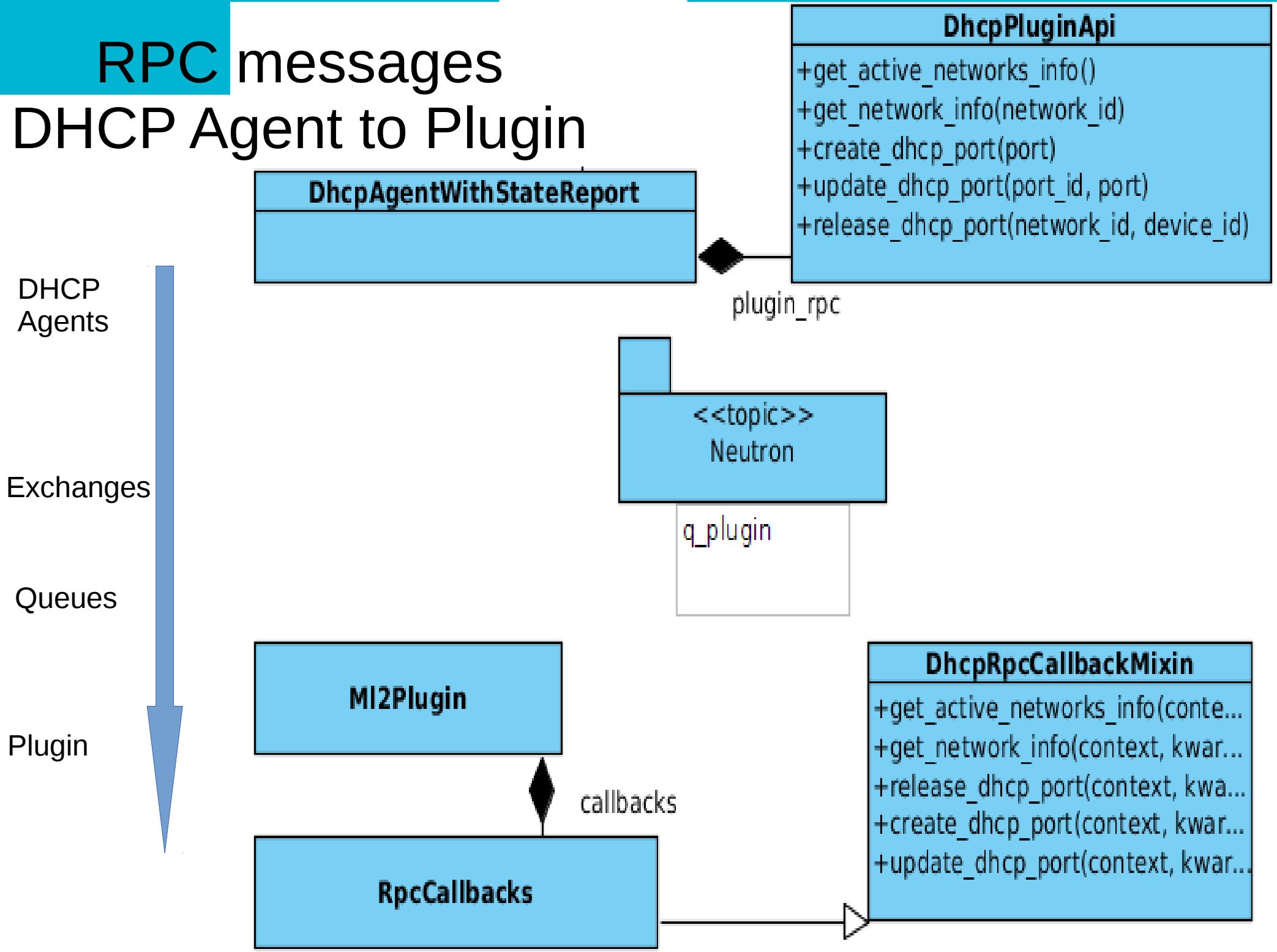

SecurityGroupServerRpcApiMixin
+security_group_rules_for_devices(...
PluginApi
+get_device_details(context, ...
+update_device_down(contex...
+update_device_up(context, ...
+tunnel_sync(context, tunnel...
L2 Agents
OVSPluginApi
plugin_rpc
OVSNeutronAgent
+network_delete(context, kwargs **)
+port_update(context, kwargs **)
+tunnel_update(context, kwargs **)
Exchanges
<<topic>>
Neutron
Queues
q_plugin
RPC messages
L2 Agent to Plugin
RpcCallbacks
+get_port_from_device(cls, device)
+get_device_details(rpc_context, k...
+update_device_down(rpc_context,...
+update_device_up(rpc_context, k...
Plugins
Ml2Plugin
callbacks
SecurityGroupServerRpcCallbackMixin
+security_group_rules_for_devices(context, ...
TunnelRpcCallbackMixin
+tunnel_sync(rpc_context, kwargs **)

RPC structure of DHCP agent




DhcpAgentNotifyAPI
+notify(context, data, methodname)
Neutron Server
<<fanout>>
dhcp_agent_fanout
<<topic>>
Neutron
'network.create.end',
'network.update.end',
'network.delete.end',
'subnet.create.end', 'subnet.update.end', 'subnet.delete.end', 'port.create.end', 'port.update.end', 'port.delete.end'
Exchanges
dhcp_agent_fanout_<uuid>
dhcp_agent.<host>
Queues
DhcpAgentWithStateReport
Messages from Neutron server to DHCP agent
DHCP Agents
DhcpAgent
+network_create_end(context, payload)
+network_update_end(context, payload)
+network_delete_end(context, payload)
+subnet_update_end(context, payload)
+subnet_delete_end(context, payload)
+port_update_end(context, payload)
+port_delete_end(context, payload)

RPC messages
DHCP Agent to Plugin



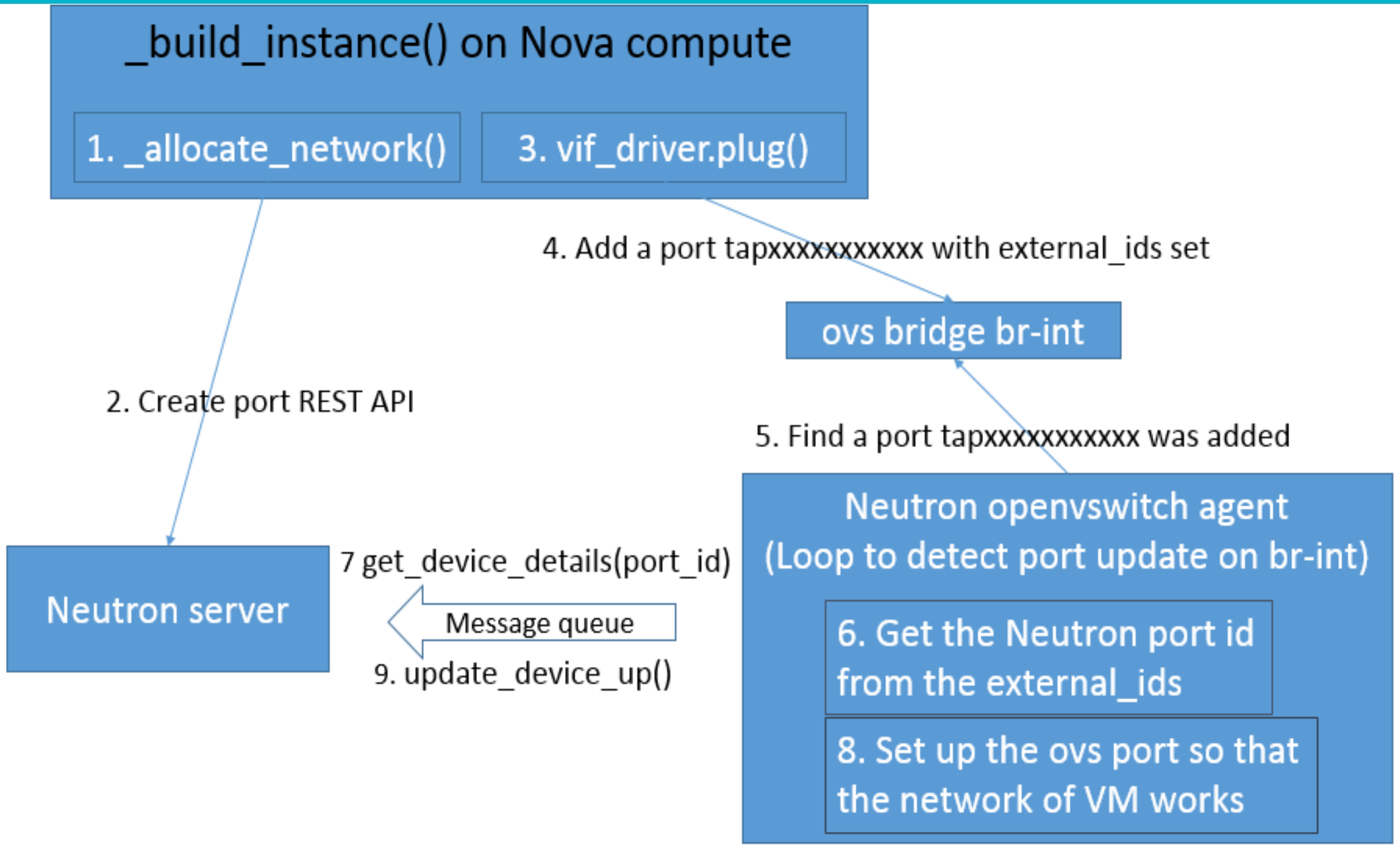
Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

Some Neutron options in Nova.conf

- network_api_class = nova.network.neutronv2.api.API
- neutron_url = http://172.16.108.1:9696
- neutron_region_name = RegionOne
- neutron_admin_tenant_name = service
- neutron_auth_strategy = keystone
- neutron_admin_auth_url = http://172.16.108.1:35357/v2.0
- neutron_admin_password = password
- neutron_admin_username = neutron
- libvirt_vif_driver = nova.virt.libvirt.vif.LibvirtGenericVIFDriver

interaction to boot VM (OVS bridge)

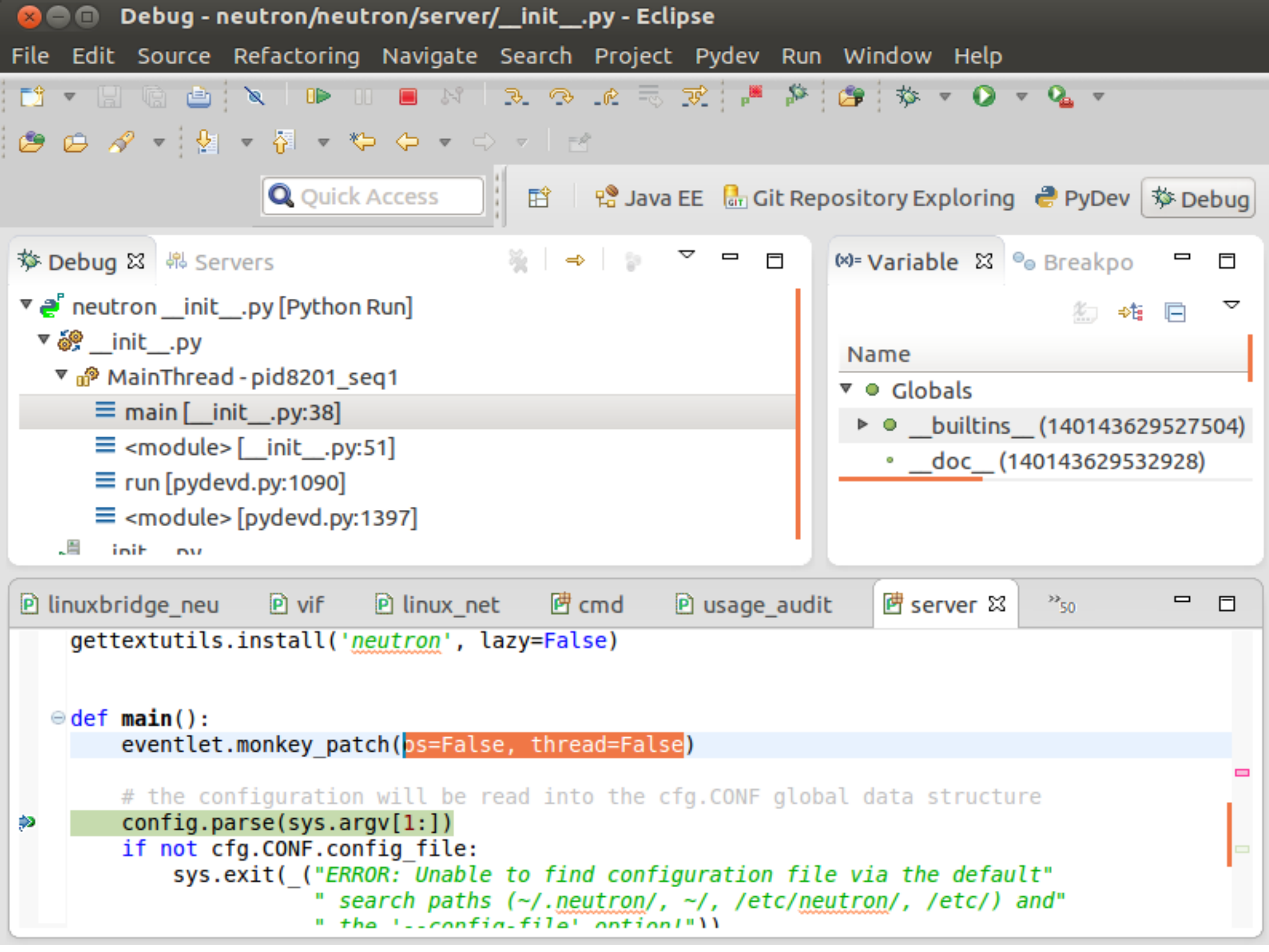


Contents

- the process of neutron start
- the normal steps to process a request
- Start ML2 plugin
- message queues in Neutron
- interaction with nova compute
- To debug the Neutron

debug Neutron

- https://wiki.openstack.org/wiki/NeutronDevelopment
- Eclipse pydev to debug neutron server
  - neutron/server/__init__.py:
    - change eventlet.monkey_patch() To: eventlet.monkey_patch(os=False, thread=False)
  - and then create a python run/debug configuration with the correct parameter such as "--config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini"


Debug - neutron/neutron/server/__init__.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
Quick Access
Java EE
Git Repository Exploring
PyDev
Debug
Debug
Servers
neutron __init__.py [Python Run]
__init__.py
MainThread - pid8201_seq1
main [__init__.py:38]
<module> [__init__.py:51]
run [pydevd.py:1090]
<module> [pydevd.py:1397]
Variable
Breakpo
Name
Globals
__builtins__ (140143629527504)
__doc__ (140143629532928)
linuxbridge_neu
vif
linux_net
cmd
usage_audit
server
gettextutils.install('neutron', lazy=False)
def main():
eventlet.monkey_patch(os=False, thread=False)
# the configuration will be read into the cfg.CONF global data structure
config.parse(sys.argv[1:])
if not cfg.CONF.config_file:
sys.exit(_("ERROR: Unable to find configuration file via the default"
" search paths (~/.neutron/, ~/, /etc/neutron/, /etc/) and"

ipdb

- add the following line to the neutron/server/__init__.py:

```
import ipdb; ipdb.set_trace()
```

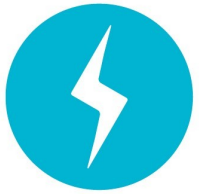
- start the neutron server

ipdb debug

gongysh@gongysh-ThinkPad-T530: ~

```
> /mnt/data/opt/stack/neutron/neutron/server/__init__.py(36)main()
     35     import ipdb; ipdb.set_trace()
---> 36     eventlet.monkey_patch()
     37

ipdb> n
> /mnt/data/opt/stack/neutron/neutron/server/__init__.py(39)main()
     38     # the configuration will be read into the cfg.CONF global
e
---> 39     config.parse(sys.argv[1:])
     40     if not cfg.CONF.config_file:

ipdb> n
> /mnt/data/opt/stack/neutron/neutron/server/__init__.py(40)main()
     39     config.parse(sys.argv[1:])
---> 40     if not cfg.CONF.config_file:
     41         sys.exit(_("ERROR: Unable to find configuration file v
t"

ipdb> p cfg.CONF.config_file
['/etc/neutron/neutron.conf', '/etc/neutron/plugins/ml2/ml2_conf.ini']
ipdb>
```

Thanks



UnitedStack