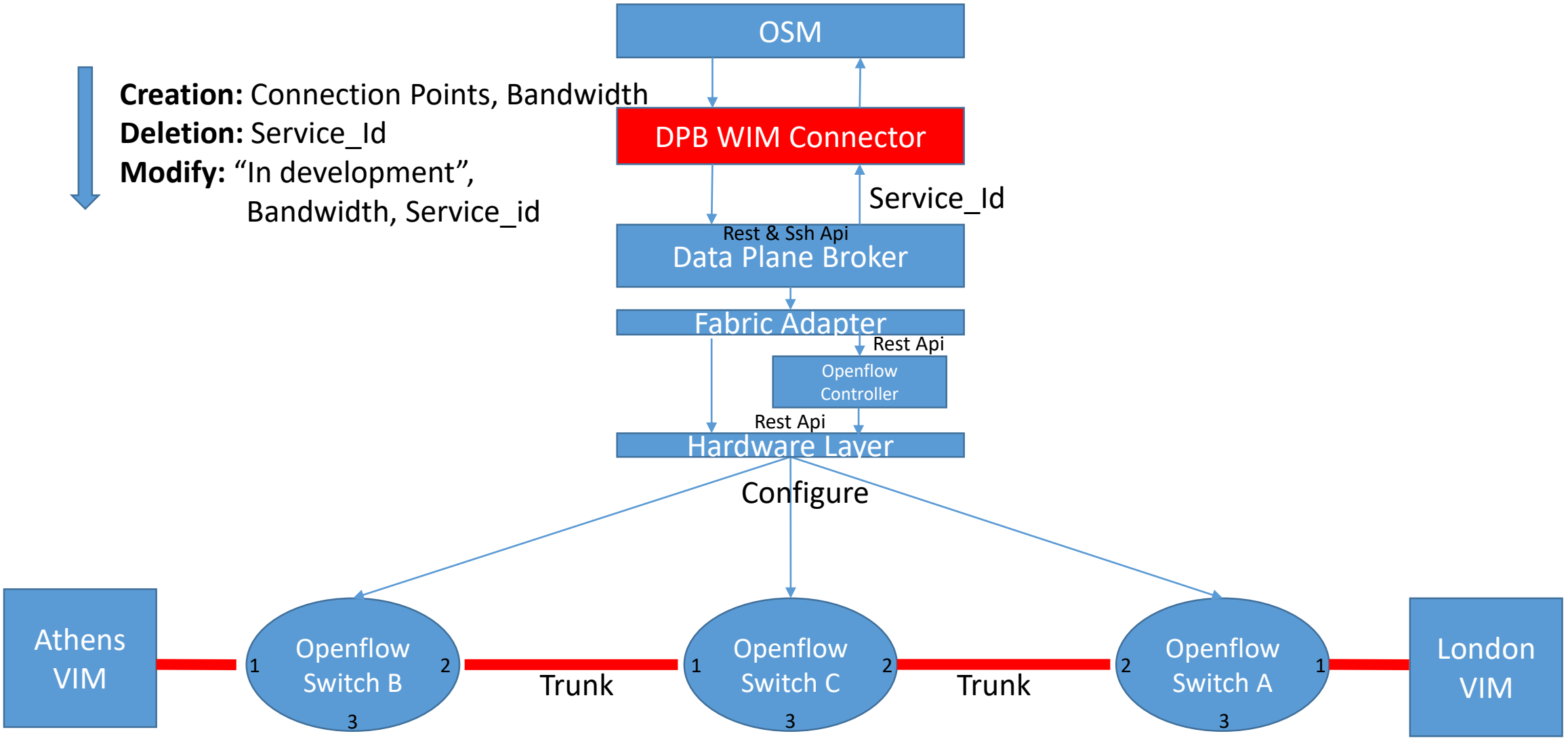
Data Plane Broker Integration, Configuration & Implementation.

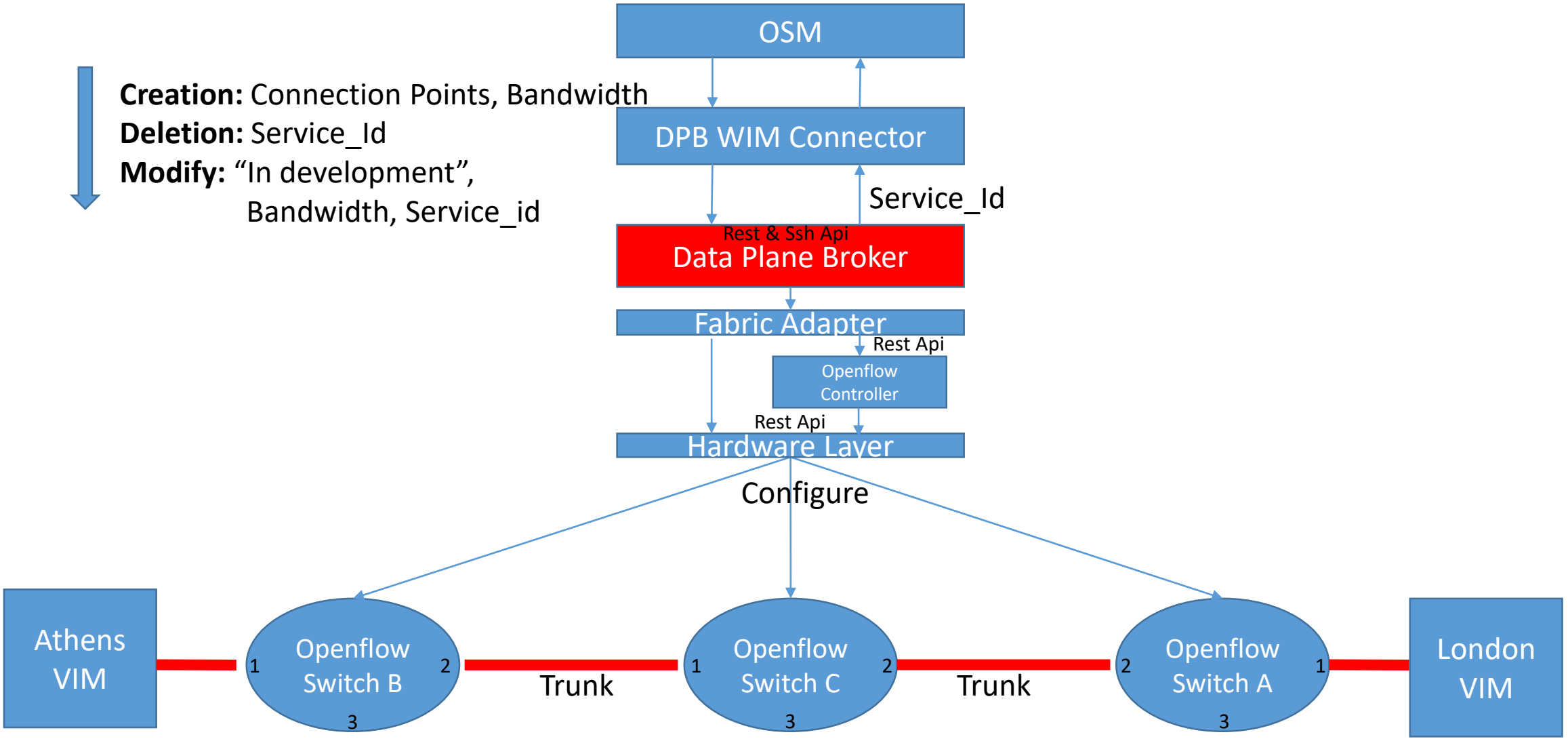
Paul McCherry

Lancaster University

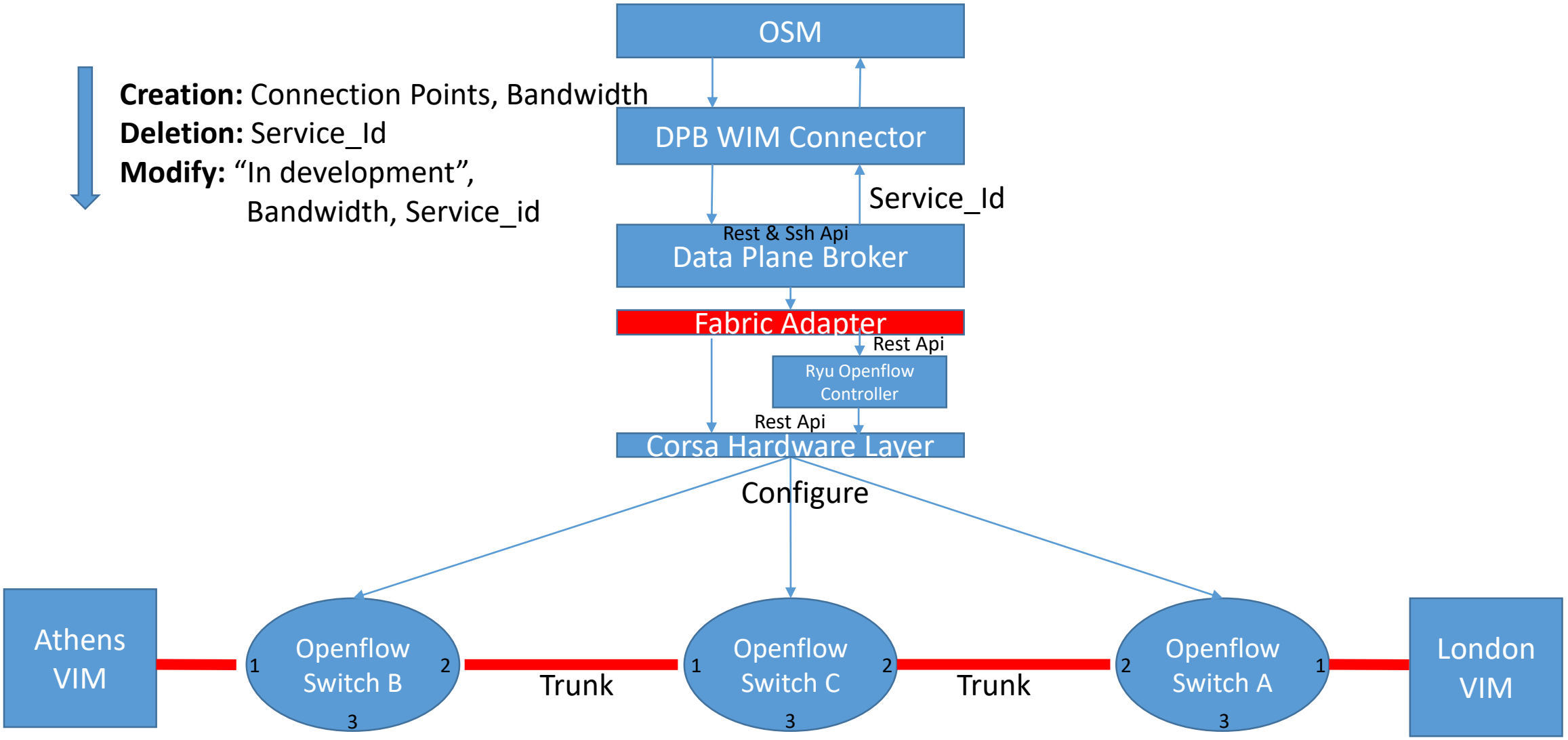
Data Plane Broker Integration



Data Plane Broker Integration



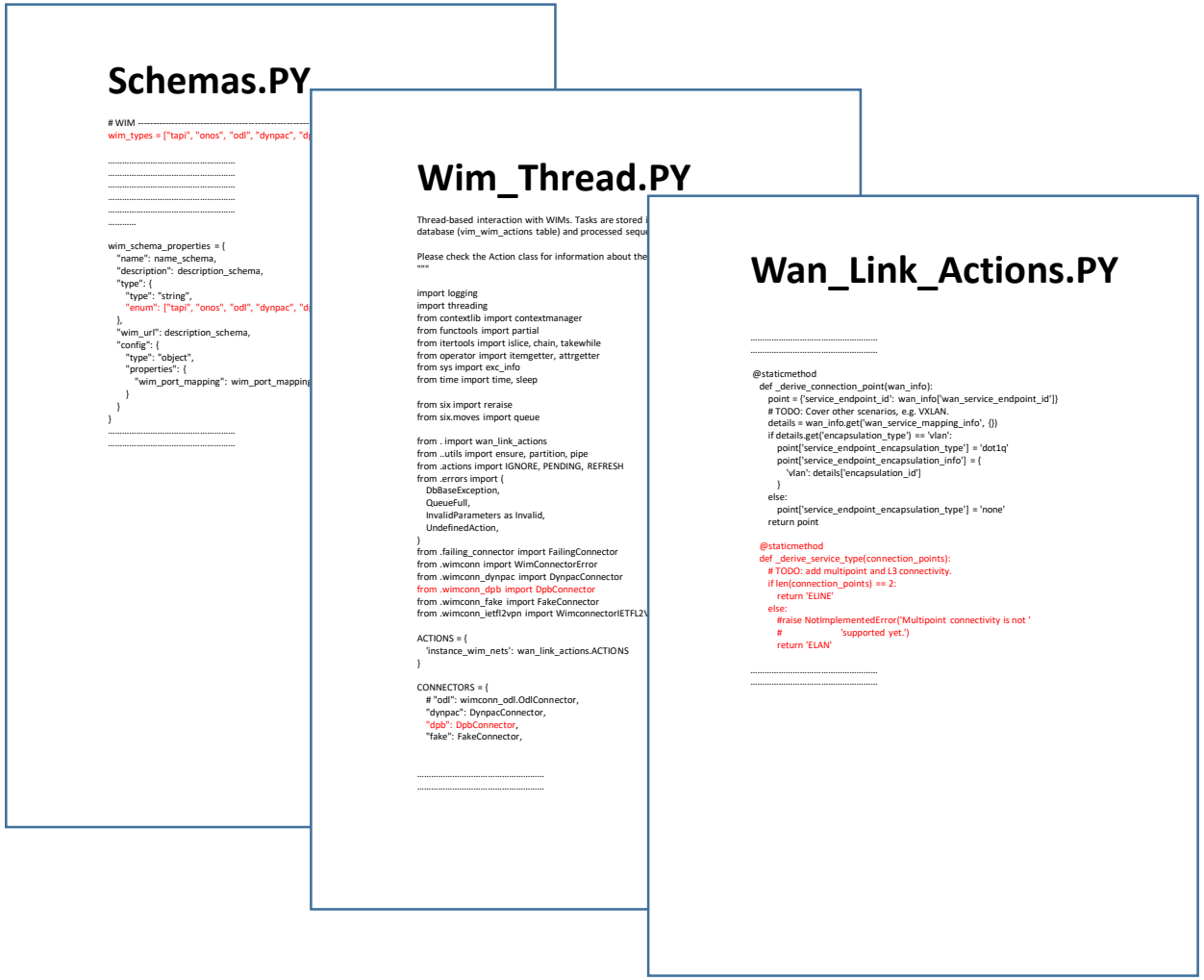
Data Plane Broker Integration



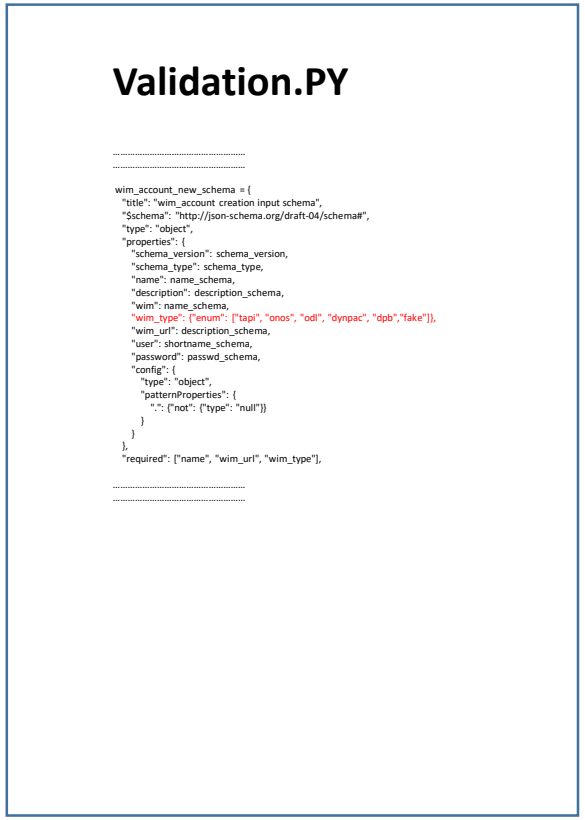
Data Plane Broker Integration

- Identify Modules which needed editing to add Dpb Wim connector

RO Container



NBI Container



Data Plane Broker Integration

RO Container

Wimconn_dpb.py

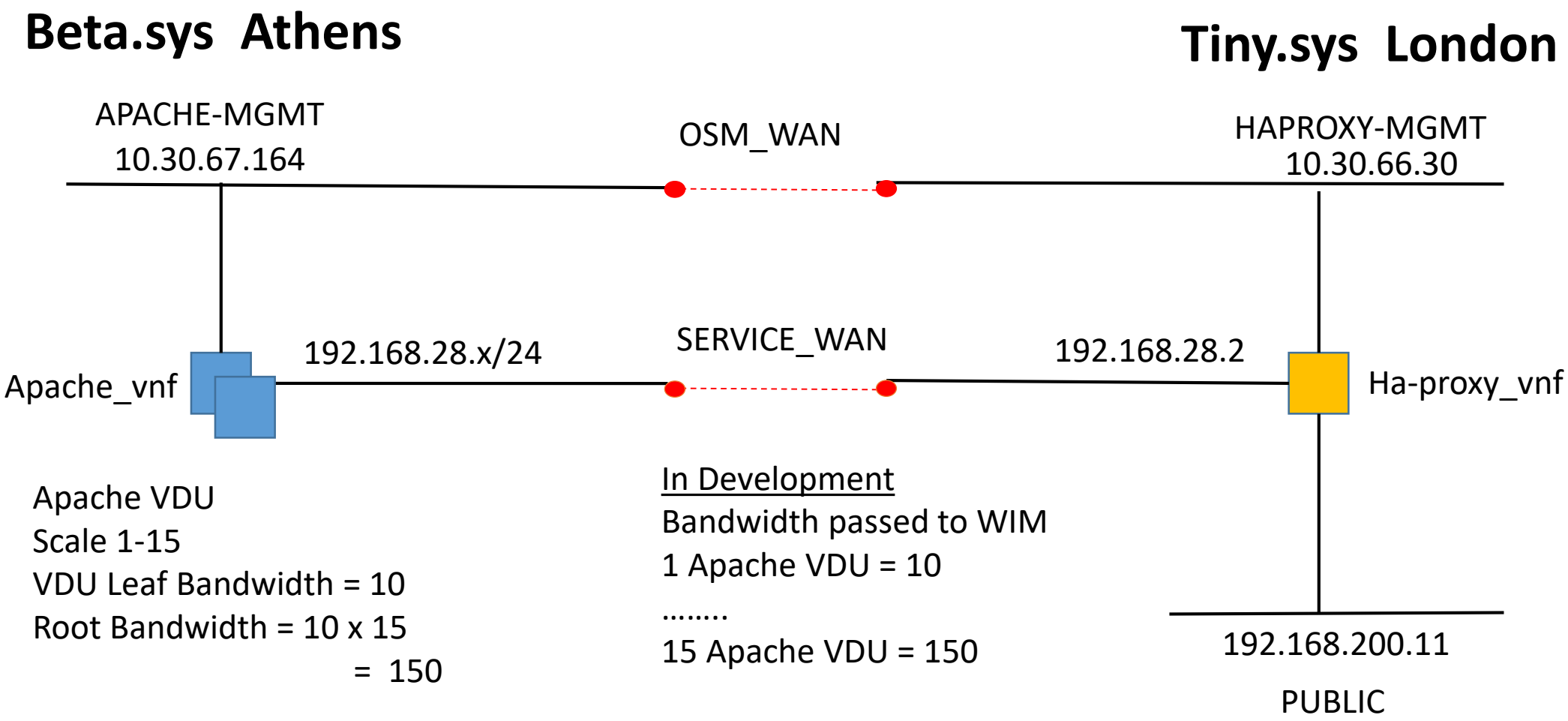
```
# TODO: List
#  - Add correct HTTP error codes
#  - Add some comments....
#  - PEP8 it

class DpbSshInterface():
   """ Communicate with the DPB via SSH """

   __LOGGER_NAME_EXT = ".ssh"
   __FUNCTION_MAP_POS = 1

   def __init__(self, wim_account, wim_url, wim_port, network, auth_data, logger_name):

      self.logger.info("SSH connection to DPB made OK")

class DpbRestInterface():
   """ Communicate with the DPB via the REST API """

   __LOGGER_NAME_EXT = ".rest"
   __FUNCTION_MAP_POS = 0

   def __init__(self, wim_account, wim_url, wim_port, network, logger_name):
.....

class DpbConnector(WimConnector):
   """ Use the DPB to establish multipoint connections """
__SUPPORTED_SERV_TYPES = ["ELAN (L2)", "ELINE (L2)"]
   __SUPPORTED_CONNECTION_TYPES = ["REST", "SSH"]
   __SUPPORTED_SSH_AUTH_TYPES = ["KEY", "PASS"]
   __SUPPORTED_SSH_KEY_TYPES = ["ECDSA", "RSA"]
```

```
   def create_connectivity_service(self, service_type, connection_points, **kwargs):
      self.logger.info("CREATING CONNECTIVITY SERVICE")
      #self.__check_service(service_type, connection_points, kwargs)
      response = self.__post(self.__ACTIONS_MAP.get("CREATE"))
.....
            "ingress-bw": 10.0,
             "egress-bw": 10.0})
             #"ingress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("ingress"),
             #"egress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("egress")}
.....
```

```
def delete_connectivity_service(self, service_uuid, conn_info=None):
#       self.__post(self.__ACTIONS_MAP.get("DEACTIVATE"), "/service/"+service_id)
      self.logger.debug("deletion infO - suuId= "+str(service_uuid)+" conn_info= "+str(conn_info))
.....
```

```
 def edit_connectivity_service(self, service_uuid, conn_info=None,
                               connection_points=None, **kwargs):
      """Change an existing connectivity service.
This method's arguments and return value follow the same convention as
      :meth:`~.create_connectivity_service`.
```

Usage Case for a Network Service across a WIM

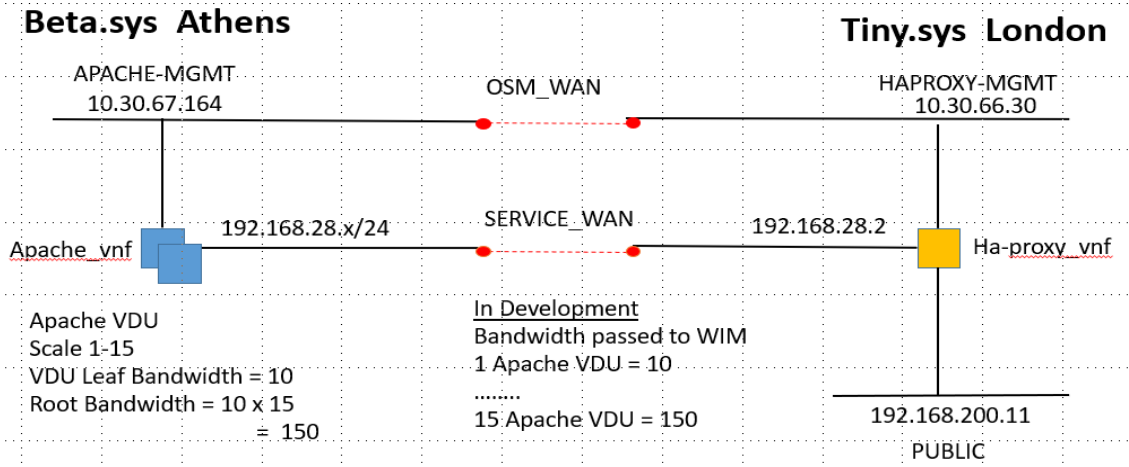


Configuration of the WIM

OSM Vim-Create

1

```
#create 2 datacentres, example file vimcreate2.sh
# create Athens on beta.sys Vim
osm vim-create --name athens-datacentre --auth_url http://beta.sys:5000/v3 --tenant admin --
user demo --password demo --account_type openstack --description “ATHENS Project demo on
Beta.sys" --config '{"external_connections": [ {"condition": {
"provider:physical_network":"provider", "provider:network_type": "vlan"},
"vim_external_port":{"switch":"openflow:1","port":"1"}}]}'

# create London on tiny.sys Vim
osm vim-create --name London-datacentre --auth_url http://tiny.sys:5000/v3 --tenant admin --
user demo --password demo --account_type openstack --description “London Project admin on
tiny.sys" --config '{"external_connections": [ {"condition": {
"provider:physical_network":"provider", "provider:network_type": "vlan"},
"vim_external_port":{"switch":"openflow:1","port":“2"}}]}'
```



OSM Wim-Create

2

```
CREATION OF A WiM USING REST Interface
#
# example file wimcreate2.sh
# uncomment either rest config or ssh config
#create a 2 vim wim
osm wim-create \
--name London-Athens \
--url beta.sys \
--user initiate \
--password admin \
--wim_type dpb \
--description "Demo 2 endpoint dpb WIM" \
--wim_port_mapping ./wim_ports2.yaml \
--config '{"port":4753,"network":"aggr","connection_type":"REST"}'
#--config '{"port":22,"network":"aggr","connection_type":"SSH","ssh_auth": \
# {"auth_type":"KEY","key_file":"/home/paul/.ssh/id_ecdsa","key_type":"ECDSA"}}'
```

```
# wim_ports2.yaml
---
- datacenter_name: “athens-datacentre"
  pop_wan_mappings:
  - pop_switch_dpid: "openflow:1"
    pop_switch_port: 1
    wan_service_endpoint_id: “athens"
    wan_service_mapping_info:
      mapping_type: direct-connect

- datacenter_name: “London-datacentre"
  pop_wan_mappings:
  - pop_switch_dpid: "openflow:1"
    pop_switch_port: 2
    wan_service_endpoint_id: “London"
    wan_service_mapping_info:
      mapping_type: direct-connect
```

Configuration of the WIM

CREATION OF A WiM USING REST or SSH Interface

```
#
# example file wimcreate2.sh
# uncomment either rest config or ssh config
#create a 2 vim wim
osm wim-create \
--name London-Athens \
--url beta.sys \
--user demo \
--password demo \
--wim_type dpb \            # NOTE USE OF NEW DPB WIM TYPE
--description "Demo 2 endpoint dpb WIM" \
--wim_port_mapping ./wim_ports2.yaml \
#--config '{"port":22,"network":"aggr","connection_type":"SSH","ssh_auth": \
# {"auth_type":"KEY","key_file":"/home/paul/.ssh/id_ecdsa","key_type":"ECDSA"}}'

--config '{"port":4753,"network":"aggr","connection_type":"REST"}'
```

RO Container

Wimconn_dpb.py

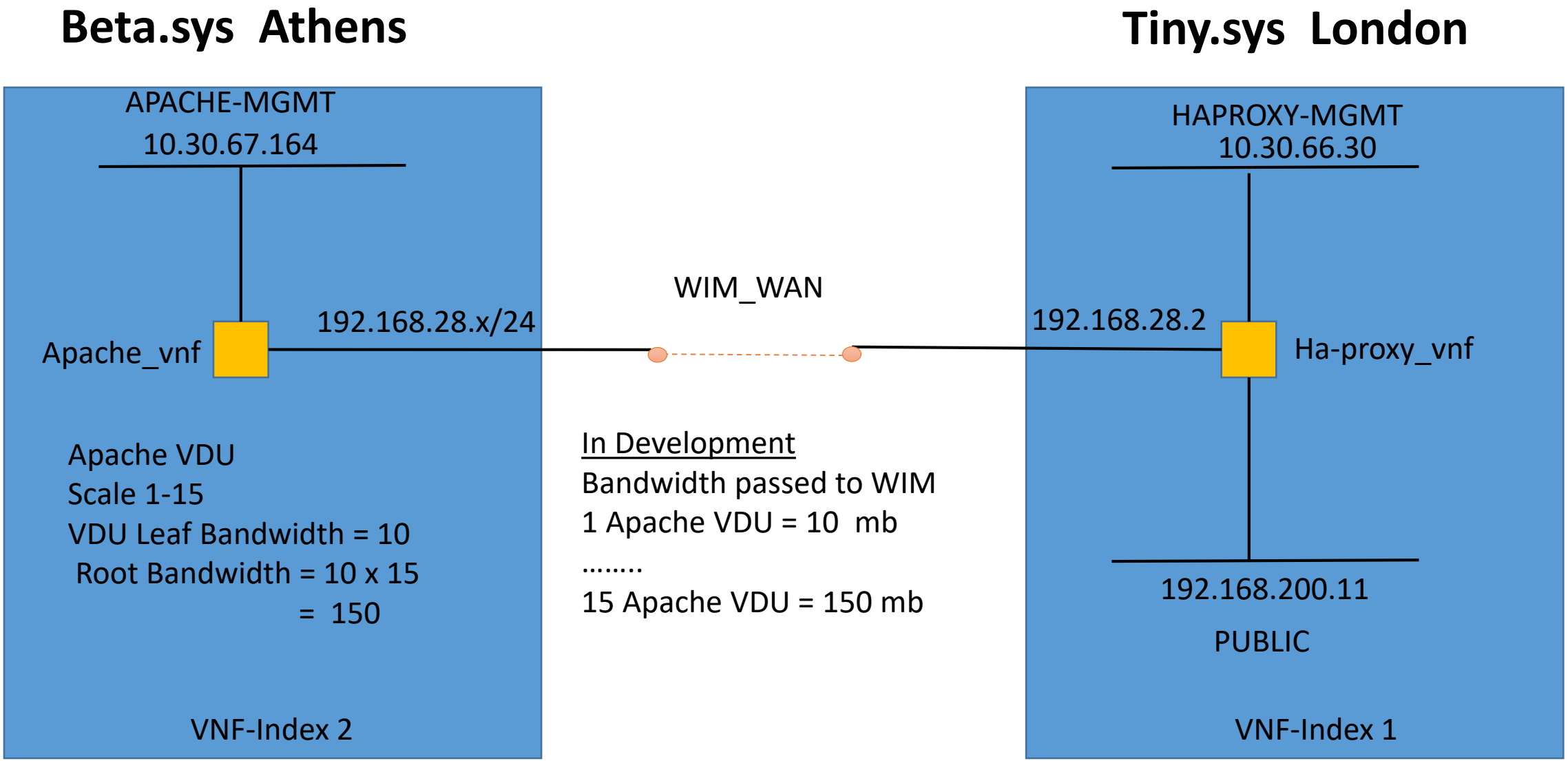
```
# TODO: List
#  - Add correct HTTP error codes
#  - Add some comments....
#  - PEP8 it

class DpbSshInterface():
    """ Communicate with the DPB via SSH """

    __LOGGER_NAME_EXT = ".ssh"
    __FUNCTION_MAP_POS = 1

    def __init__(self, wim_account, wim_url, wim_port, network, auth_data, logger_name):

        self.logger.info("SSH connection to DPB made OK")

class DpbRestInterface():
    """ Communicate with the DPB via the REST API """

    __LOGGER_NAME_EXT = ".rest"
    __FUNCTION_MAP_POS = 0

    def __init__(self, wim_account, wim_url, wim_port, network, logger_name):
.....

class DpbConnector(WimConnector):
    """ Use the DPB to establish multipoint connections """
__SUPPORTED_SERV_TYPES = ["ELAN (L2)", "ELINE (L2)"]
    __SUPPORTED_CONNECTION_TYPES = ["REST", "SSH"]
    __SUPPORTED_SSH_AUTH_TYPES = ["KEY", "PASS"]
    __SUPPORTED_SSH_KEY_TYPES = ["ECDSA", "RSA"]
```

```
    def create_connectivity_service(self, service_type, connection_points, **kwargs):
        self.logger.info("CREATING CONNECTIVITY SERVICE")
        #self.__check_service(service_type, connection_points, kwargs)
        response = self.__post(self.__ACTIONS_MAP.get("CREATE"))
.....
                "ingress-bw": 10.0,
                "egress-bw": 10.0}
                #"ingress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("ingress"),
                #"egress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("egress")}
```

```
def delete_connectivity_service(self, service_uuid, conn_info=None):
#       self.__post(self.__ACTIONS_MAP.get("DEACTIVATE"), "/service/"+service_id)
        self.logger.debug("deletion infO - suuId= "+str(service_uuid)+" conn_info= "+str(conn_info))
.....
```

```
    def edit_connectivity_service(self, service_uuid, conn_info=None,
                                  connection_points=None, **kwargs):
        """Change an existing connectivity service.
    This method's arguments and return value follow the same convention as
        :meth:`~.create_connectivity_service`.
```

Implementation of the Network Service across the WIM

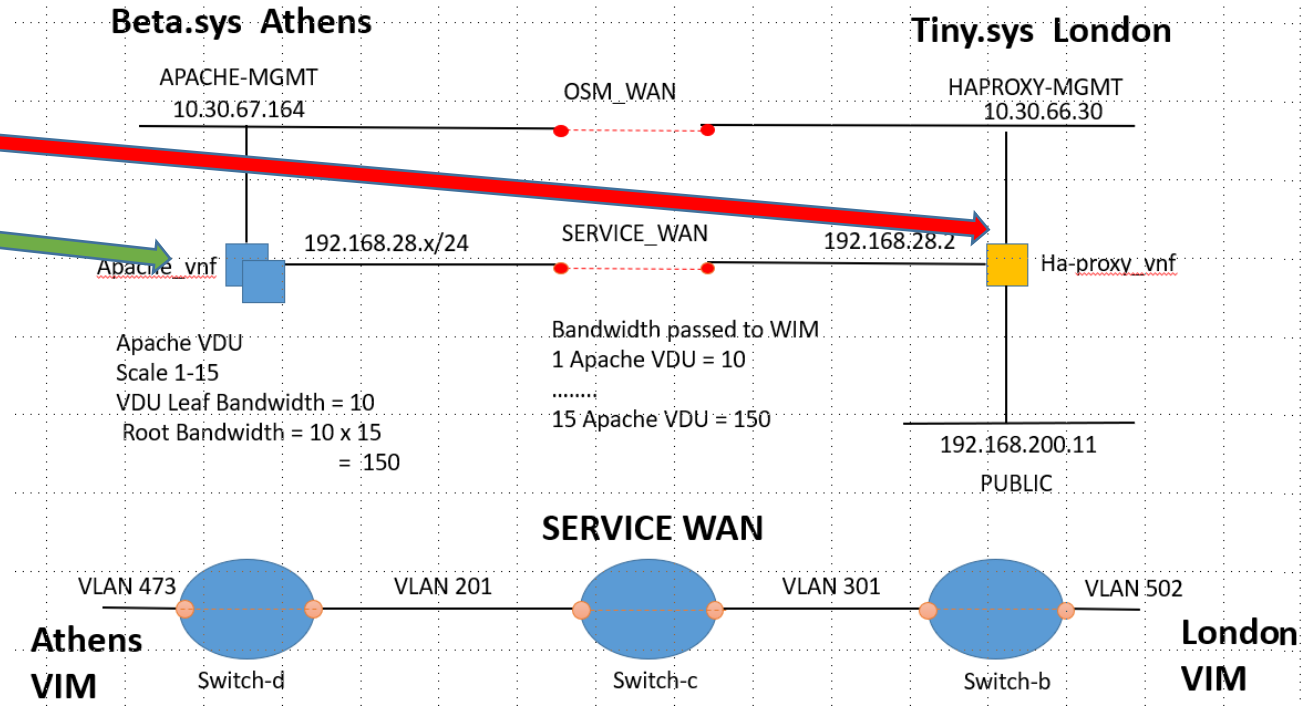


Implementation of the Network Service across the WIM

```
# example file nscreate2.sh

# create 1 ns, 2 vnfs and 1 network across 2 vims (WIM)
osm ns-create \
--ns_name WIM \
--nsd_name webserver_wimmetric_autoscale_ns \
--vim_account athens-datacentre \
--config '{vnf:[{member-vnf-index: "1", vim_account: London-datacentre}, \
{member-vnf-index: "2", vim_account: athens-datacentre}]}'
```

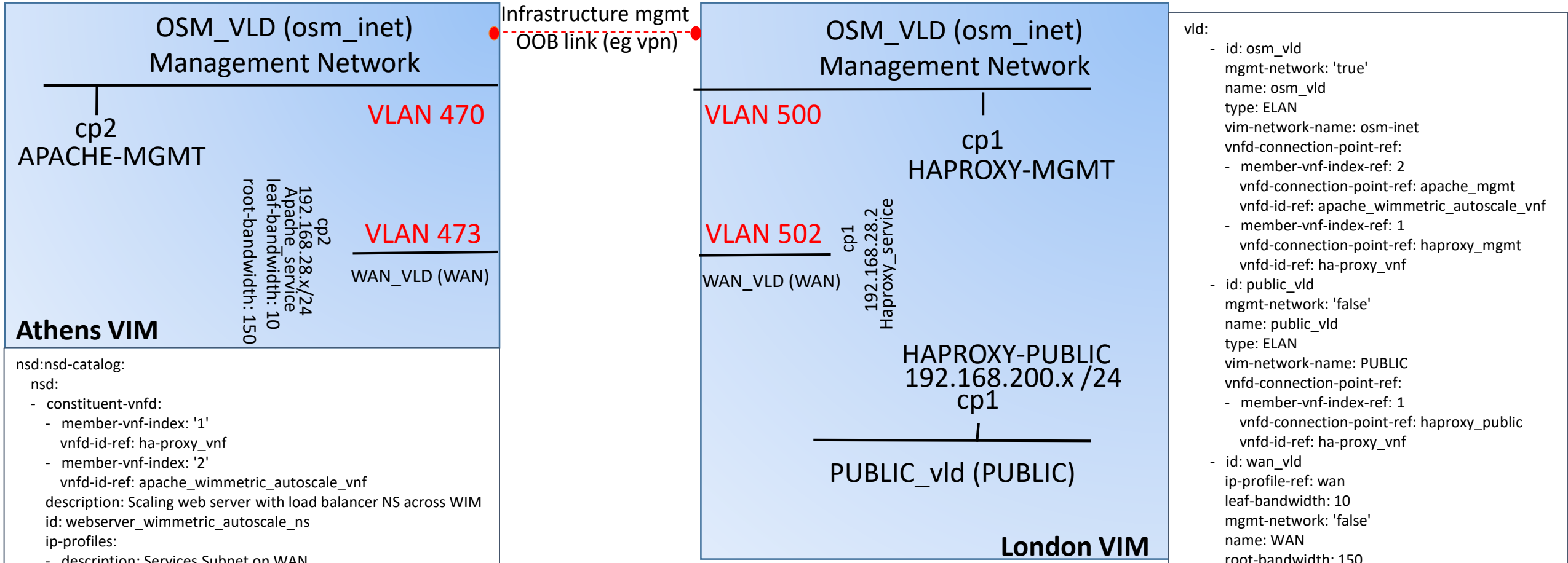
Network Service Descriptor

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: '1'
      vnfd-id-ref: ha-proxy_vnf
    - member-vnf-index: '2'
      vnfd-id-ref: apache_wimmetric_autoscale_vnf
    description: Scaling web server with load balancer NS across
    id: webserver_wimmetric_autoscale_ns
    ip-profiles:
    - description: Services Subnet on WAN
      ip-profile-params:
        dhcp-params:
          enabled: true
        gateway-address: 0.0.0.0
        ip-version: ipv4
        subnet-address: 192.168.28.0/24
      name: wan
    name: webserver_wimmetric_autoscale_ns
    short-name: webserver_wimmetric_autoscale_ns
    vendor: p.mccherry@lancaster.ac.uk
    version: '1.0'
```



Network Service Descriptor

Implementation of the Network Service across the WIM

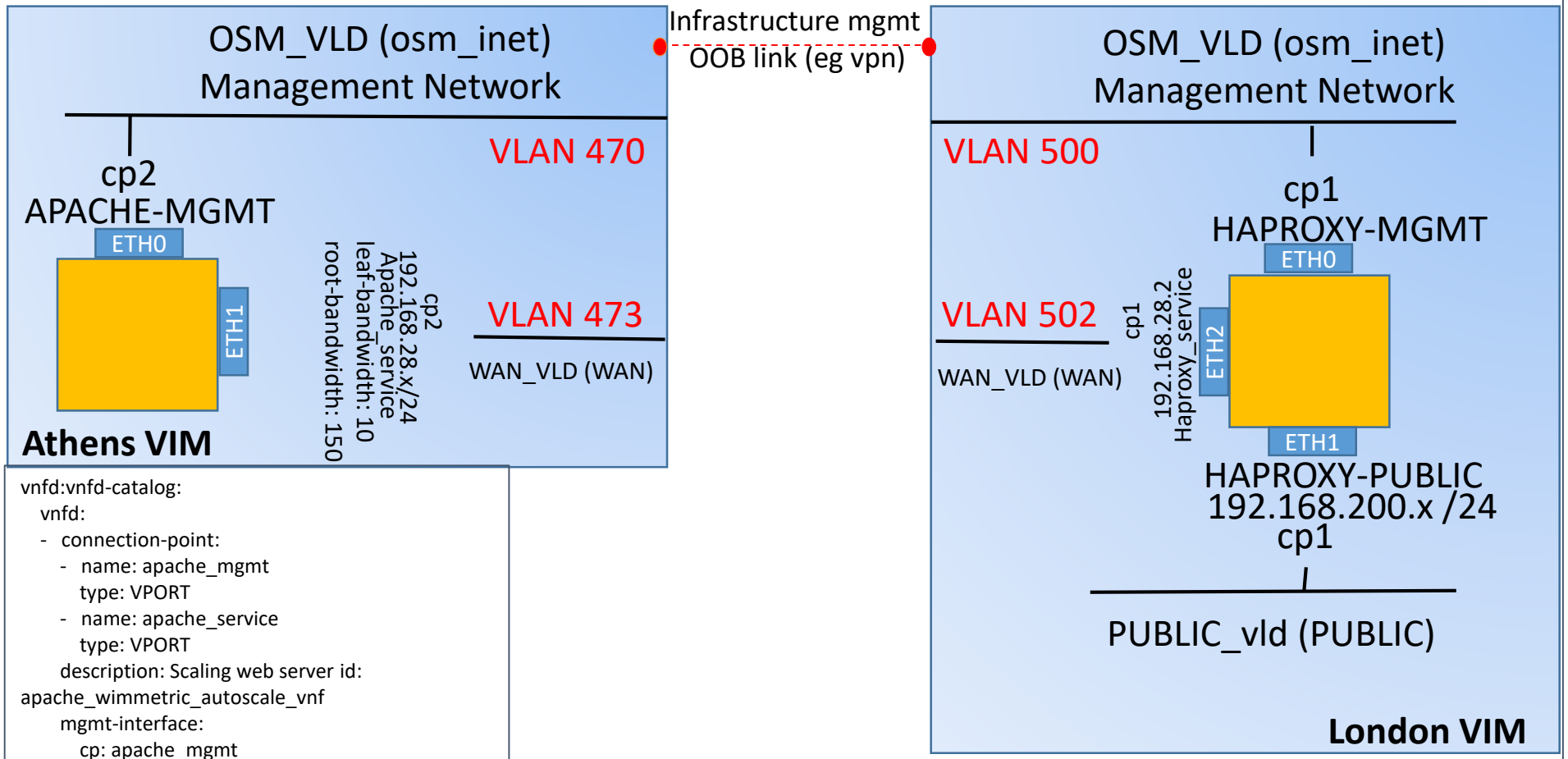



```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: '1'
      vnfd-id-ref: ha-proxy_vnf
    - member-vnf-index: '2'
      vnfd-id-ref: apache_wimmetric_autoscale_vnf
    description: Scaling web server with load balancer NS across WIM
    id: webserver_wimmetric_autoscale_ns
    ip-profiles:
    - description: Services Subnet on WAN
      ip-profile-params:
        dhcp-params:
          enabled: true
        gateway-address: 0.0.0.0
        ip-version: ipv4
        subnet-address: 192.168.28.0/24
      name: wan
    name: webserver_wimmetric_autoscale_ns
    short-name: webserver_wimmetric_autoscale_ns
    vendor: p.mccherry@lancaster.ac.uk
    version: '1.0'
```

```
vld:
  - id: osm_vld
    mgmt-network: 'true'
    name: osm_vld
    type: ELAN
    vim-network-name: osm-inet
    vnfd-connection-point-ref:
    - member-vnf-index-ref: 2
      vnfd-connection-point-ref: apache_mgmt
      vnfd-id-ref: apache_wimmetric_autoscale_vnf
    - member-vnf-index-ref: 1
      vnfd-connection-point-ref: haproxy_mgmt
      vnfd-id-ref: ha-proxy_vnf
  - id: public_vld
    mgmt-network: 'false'
    name: public_vld
    type: ELAN
    vim-network-name: PUBLIC
    vnfd-connection-point-ref:
    - member-vnf-index-ref: 1
      vnfd-connection-point-ref: haproxy_public
      vnfd-id-ref: ha-proxy_vnf
  - id: wan_vld
    ip-profile-ref: wan
    leaf-bandwidth: 10
    mgmt-network: 'false'
    name: WAN
    root-bandwidth: 150
    type: ELAN
    vnfd-connection-point-ref:
    - ip-address: 192.168.28.2
      member-vnf-index-ref: 1
      vnfd-connection-point-ref: haproxy_service
      vnfd-id-ref: ha-proxy_vnf
    - member-vnf-index-ref: 2
      vnfd-connection-point-ref: apache_service
      vnfd-id-ref: apache_wimmetric_autoscale_vnf
```

VNF Descriptors

Implementation of the Network Service across the WIM

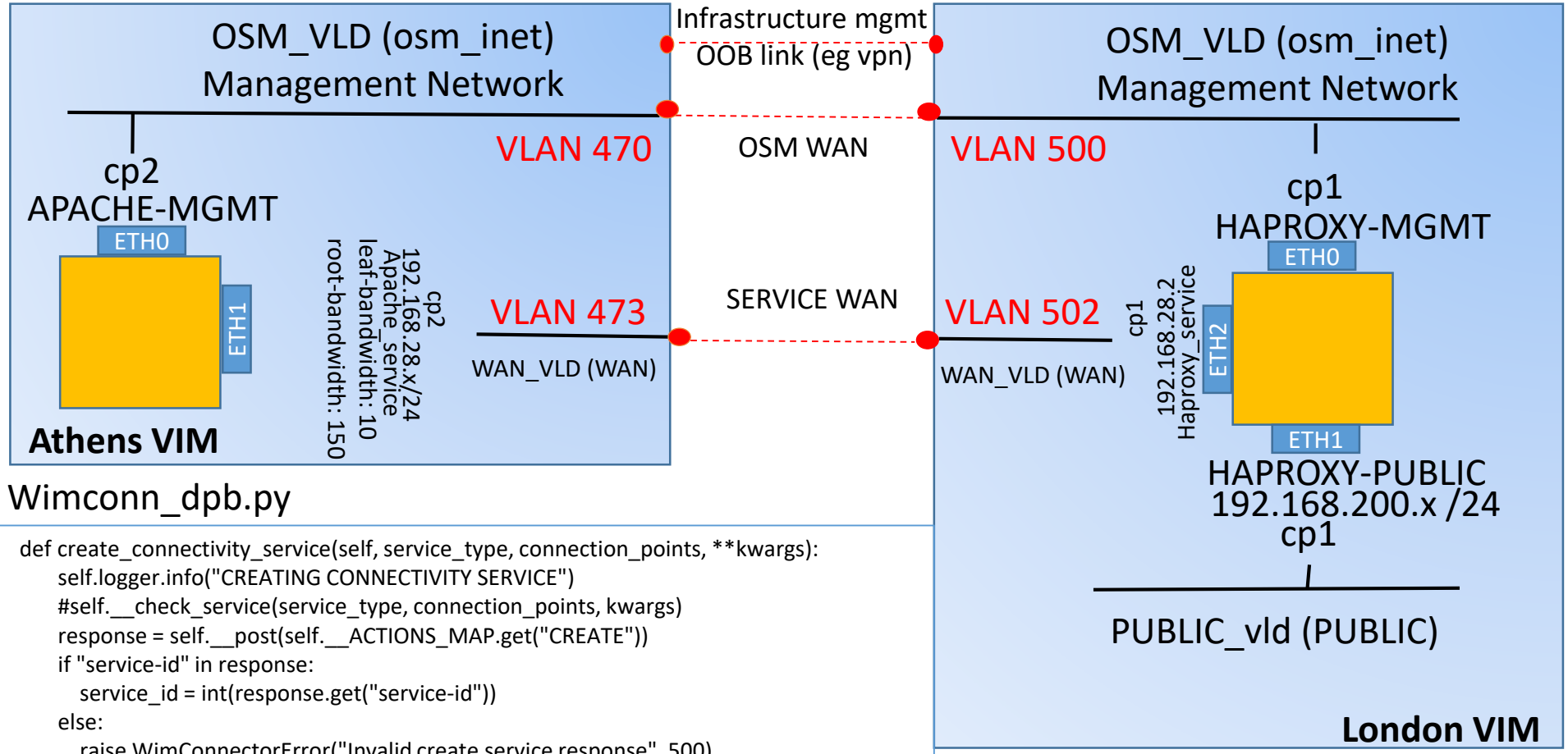



```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: apache_mgmt
      type: VPORT
    - name: apache_service
      type: VPORT
    description: Scaling web server id:
apache_wimmetric_autoscale_vnf
    mgmt-interface:
      cp: apache_mgmt
    monitoring-param:
    - aggregation-type: AVERAGE
      id: apache_vnf_cpu_util
      name: apache_vnf_cpu_util
      vdu-monitoring-param:
        vdu-monitoring-param-ref: apache_cpu_util
        vdu-ref: apache_vdu
    name: apache_wimmetric_autoscale_vnf
```

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: haproxy_mgmt
      type: VPORT
    - name: haproxy_public
      type: VPORT
    - name: haproxy_service
      type: VPORT
    description: Scaling web server with load balancer
id: pr_vnf
    mgmt-interface:
      cp: haproxy_mgmt
    name: ha-proxy_vnf
    short-name: ha-proxy_vnf
    vdu:
    - cloud-init-file: cloud_init_haproxy
      count: '1'
      description: haproxy_vdu
      id: haproxy_vdu
      image: haproxy_ubuntu
      interface:
      - external-connection-point-ref: haproxy_public
        name: haproxy_vdu_eth1
        position: '1'
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: haproxy_mgmt
        mgmt-interface: true
        name: haproxy_vdu_eth0
        position: '2'
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: haproxy_service
        name: haproxy_vdu_eth2
        position: '3'
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      name: haproxy_vdu
      vm-flavor:
        memory-mb: '4096'
        storage-gb: '10' vcpu-count: '4'
```

Wim Creation

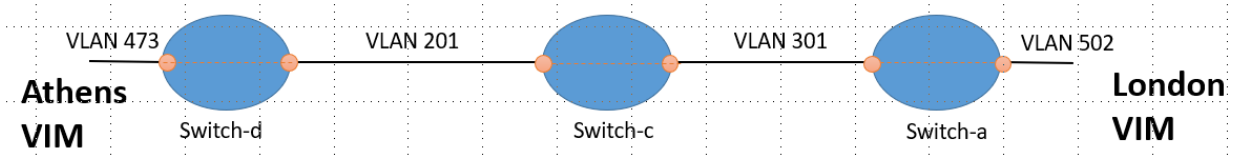
Implementation of the Network Service across the WIM



Wimconn_dpb.py

```
def create_connectivity_service(self, service_type, connection_points, **kwargs):
    self.logger.info("CREATING CONNECTIVITY SERVICE")
    #self.__check_service(service_type, connection_points, kwargs)
    response = self.__post(self.__ACTIONS_MAP.get("CREATE"))
    if "service-id" in response:
        service_id = int(response.get("service-id"))
    else:
        raise WimConnectorError("Invalid create service response", 500)
    data = {"segment": []}
    for point in connection_points:
        data["segment"].append({
            "terminal-name": point.get("service_endpoint_id"),
            "label": int((point.get("service_endpoint_encapsulation_info")).get("vlan")),
            "ingress-bw": 10.0,
            "egress-bw": 10.0})
            #"ingress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("ingress"),
            #"egress-bw": (bandwidth.get(point.get("service_endpoint_id"))).get("egress")}
    .........
    return (str(service_id), None)
```

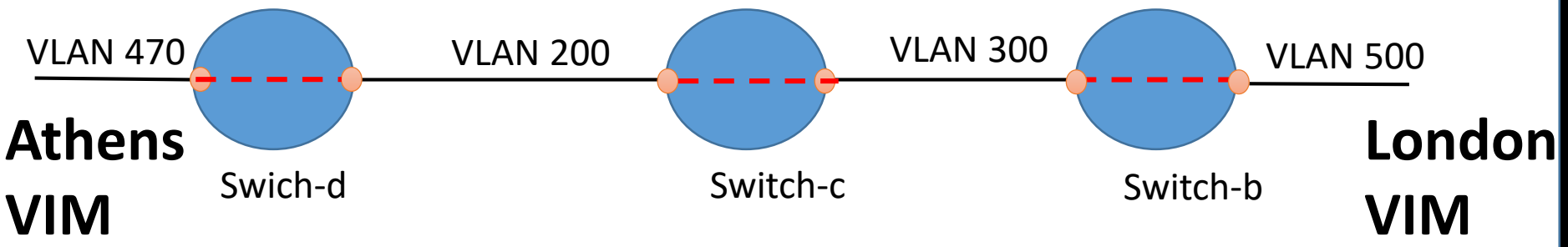
WAN Links



Implementation of the Network Service across the WIM

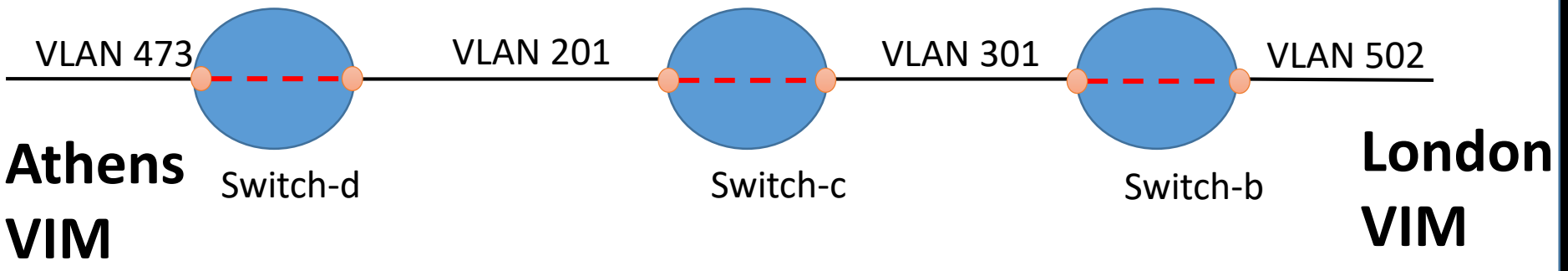
Service_id 1

OSM-INET



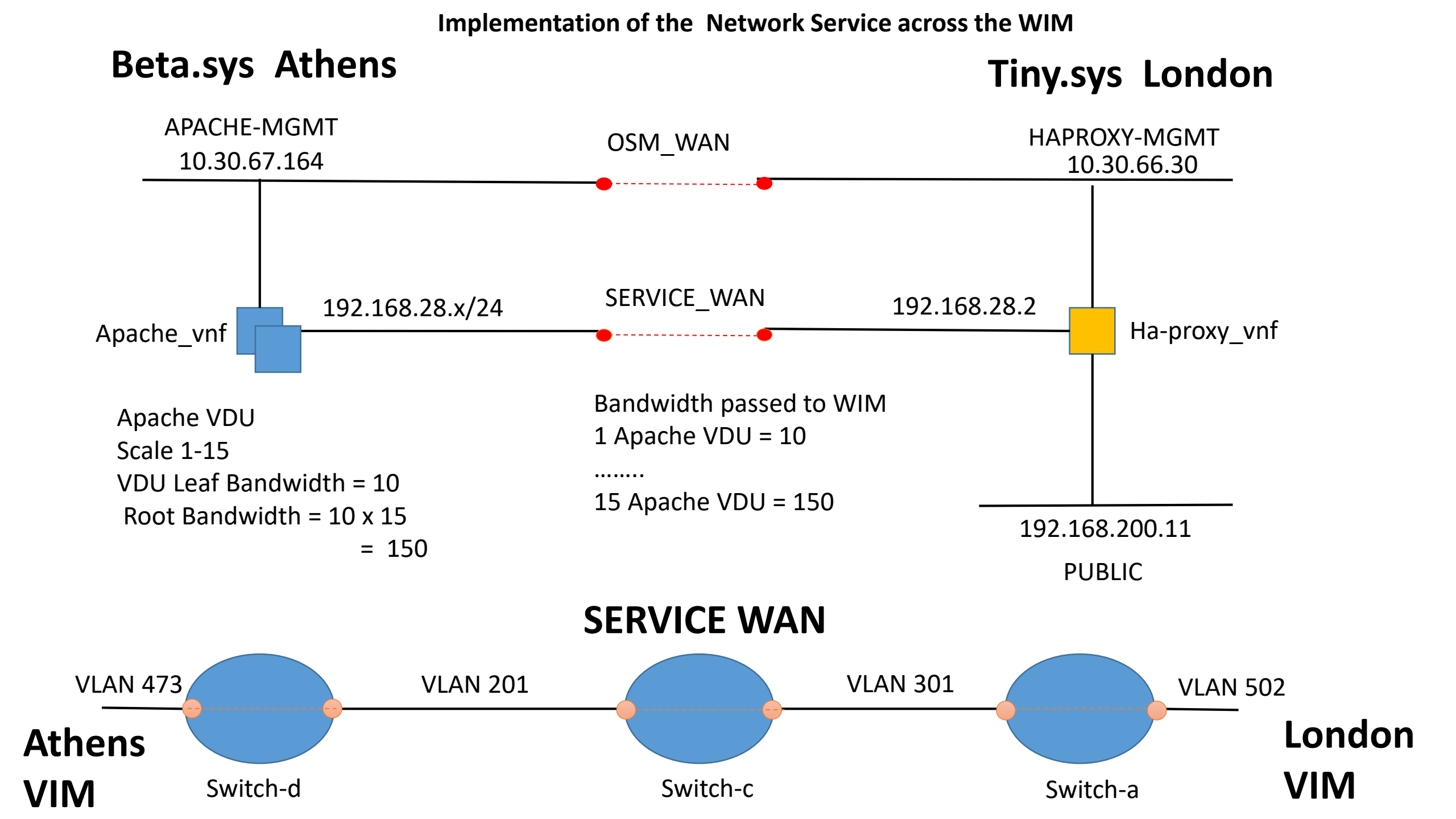
Service_id 2

SERVICE WAN

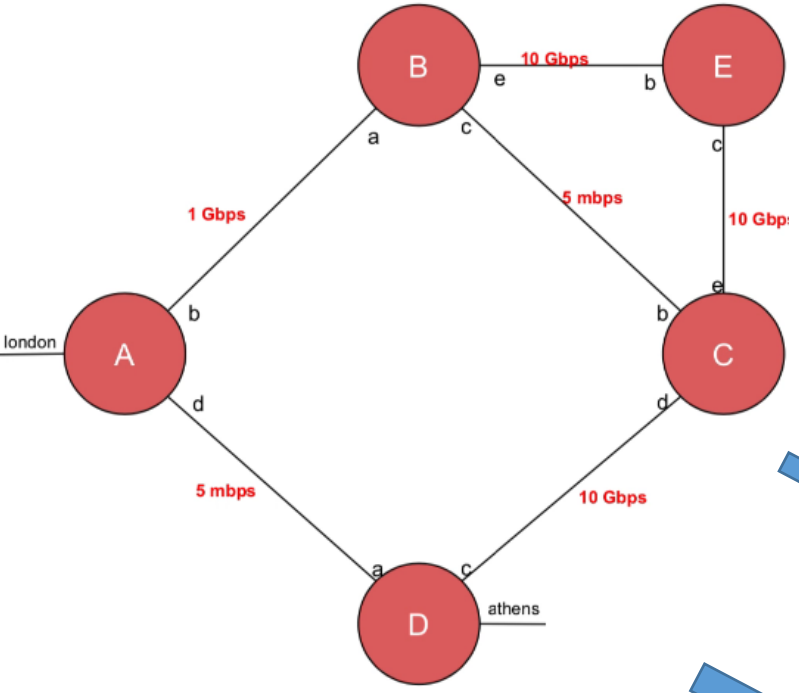



```
initiate@beta:~$ dpb -n aggr dump
Creating agents...
Obtaining networks...
Getting new multiplexer...
aggregate aggr:
   1 ACTIVE
         switch-c:d:200=switch-d:c:200
         switch-b:c:300=switch-c:b:300
     inferior ACTIVE:
       switch-d:athens:470 10.0000 10.0000
       switch-d:c:200 10.0000 10.0000
     inferior ACTIVE:
       switch-b:c:300 10.0000 10.0000
       switch-b:London:500 10.0000 10.0000
     inferior ACTIVE:
       switch-c:d:200 10.0000 10.0000
       switch-c:b:300 10.0000 10.0000
   2 ACTIVE
         switch-c:d:201=switch-d:ca:201
         switch-b:c:301=switch-c:b:301
     inferior ACTIVE:
       switch-d:athens:473 10.0000 10.0000
       switch-d:c:201 10.0000 10.0000
     inferior ACTIVE:
       switch-b:c:301 10.0000 10.0000
       switch-b:London:502 10.0000 10.0000
     inferior ACTIVE:
       switch-c:d:201 10.0000 10.0000
       switch-c:b:301 10.0000 10.0000
```

Implementation of the Network Service across the WIM



Implementation of the Multipoint Service



Berlin VIM

Berlin endpoint

London VIM

London endpoint

Openflow Switch

Paris endpoint

Paris VIM

Athens endpoint

Athens VIM

Challenges

- Instance dictionary passed from LCM into RO (nfvo.py) when undergoing vdu scaling does not include any banwidth information
- However, A separate link scaling action with its own instance dictionary would be a better option offering more flexibility.
- No Edit Function in Wim actions.py,

```
'action', # MD - CREATE, DELETE, FIND).
```

- The comment in the create connectivity service function, Keyword Arguments: bandwidth (int): value in kilobytes , does suggest that bandwidth can be passed into the create and edit methods, however, for the current setup, it only comes as a None type.
- Action needs to pass Service id, Authentications of Vims, bandwidth info

Questions ?

Paul McCherry

Lancaster University

Thankyou for your attention

Paul McCherry

Lancaster University