MDONS OOF Support - IDL/Path Optimization

Assumptions

1. Below is the list of solvers used in OOF and its usage. The Route Optimizer package will be used for the MDONS use case as well.
2. Policy is not going to be involved to list the constraints for now.
3. Once config policies are defined for MDONS use case , OOF will fetch these policies to get certain constraints and apply them while running the algorithm.
4. If two links between two controllers have the same specification then minizinc automatically chooses one. The implementation for Guilin release is based out of this logic.

Existing Solvers and Usage

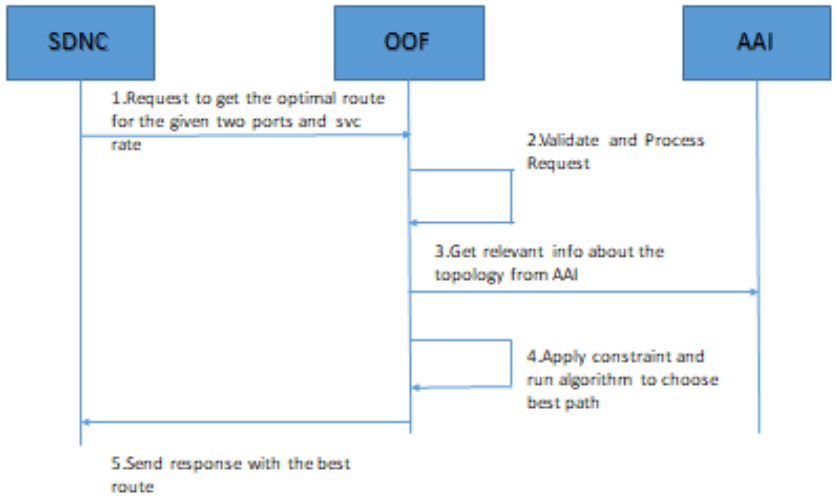
Solver	Usage
License Optimizer	VNF license optimization
PCI Optimizer	Pci optimization
Placement Optimizer	VNF placement optimization
Slice Optimizer	Slice selection and instantiation optimization
Route Optimizer	Perform the route calculations and return the vpn-bindings for CCVPN use case

OPTFRA-753 - Getting issue details... STATUS

OPTFRA-820 - Getting issue details... STATUS

Sequence Diagram

Flow Diagram



Request from SO or SDNC

IDL request

```
API: POST /api/oof/mdons/route/v1

BODY:
{
   "requestInfo":{
      "transactionId":"xxx-xxx-xxx",
      "requestId":"yyy-yyy-yyy",
      "callbackUrl":"",
      "callbackHeader": "",
      "sourceId":"SDNC",
      "requestType":"create",
      "numSolutions":1,
      "optimizers":[
         "route"
      ],
      "timeout":600
   },
   "routeInfo":{
      "routeRequest":{
         "srcDetails":{
            "interfaceId":"int19",
            "nodeId":"pnf1",
            "controllerId":"Controller1"
         },
         "dstDetails":{
            "interfaceId":"int20",
            "nodeId":"pnf4",
            "controllerId":"Controller3"
         },
         "serviceRate":"ODU2"
      }
   }
}
```

Note: "Callbackurl" field will be empty . Has been added to just follow the request format standard in OOF. When SDNC sends this request the field will be empty.

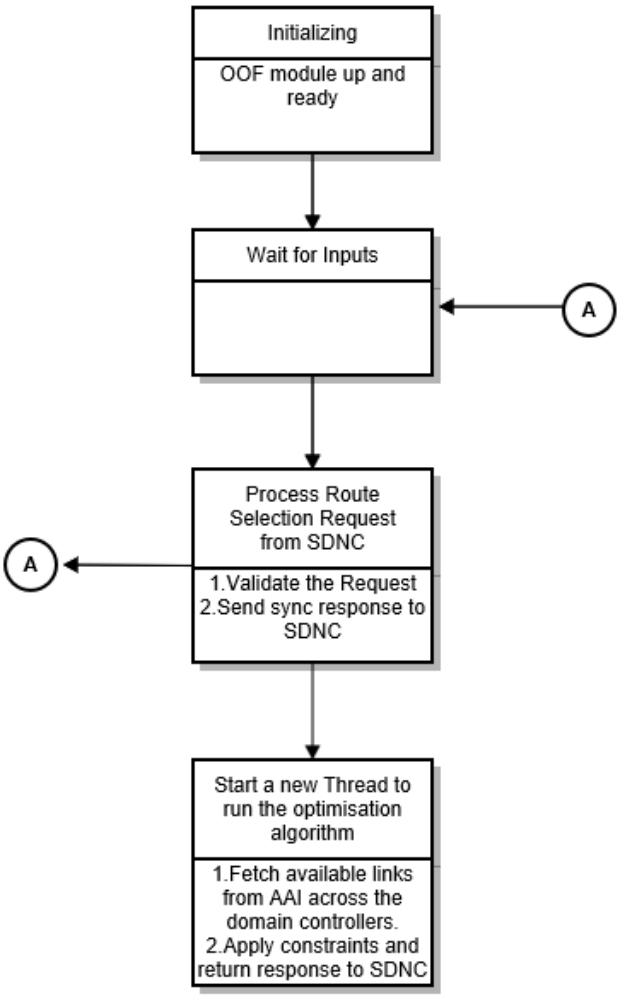
Response To SDNC

Response Example

Route Selection Response

```
{
            "requestId":"xxx-xxx-xxx,
            "transactionId":"yyy-yyy-yyy",
            "statusMessage":"SUCCESS",
            "requestStatus":"accepted",
            "solutions":{
                "routeInfo":{
                "serviceRoute":[
                    {
                     "srcInterfaceId":"int19",
                     "dstInterfaceId":"int1",
                     "controllerId":"Controller1"
                },
                    {
                     "srcInterfaceId":"int3",
                     "dstInterfaceId":"int4",
                     "controllerId":"Controller2"
                },
                    {
                     "srcInterfaceId":"int5",
                     "dstInterfaceId":"int20",
                     "controllerId":"Controller3"
                }
                ],
               "linkList":[
                    "link1",
                    "link2"
                ]
                }
           }
       }
```

State Diagram



OOF Impacts

1. A new api should be defined in the osdfapp.py class in the optf-osdf repository for the MDONS route selection.
2. Under the Route Optimizer section a new class has to be added to handle the MDONS route selection.
3. The standard minizinc template which gives the shortest path in a graph data structure will be used for this use case, same as the CCVPN use case.

Algorithm Details

1. Identify if the A and Z are from the same domain controller, if yes, linkName is set to null.
2. If not from the same domain, retrieve all the inter domain links across both the controllers from AAI.
3. The links will have information such as admin-state and rate. (Note: For now we are considering only these two fields to select the appropriate inter-domain-link).
4. Based on the bandwidth required and available bandwidth and the status of the links (of the end points, NNIs), the appropriate link is chosen.
5. The link name along with the names of NNI 1 and NN2 are sent to SDNC and SDNC/DG send the details request for service-creation to the domain controller(s).

Minizinc Template

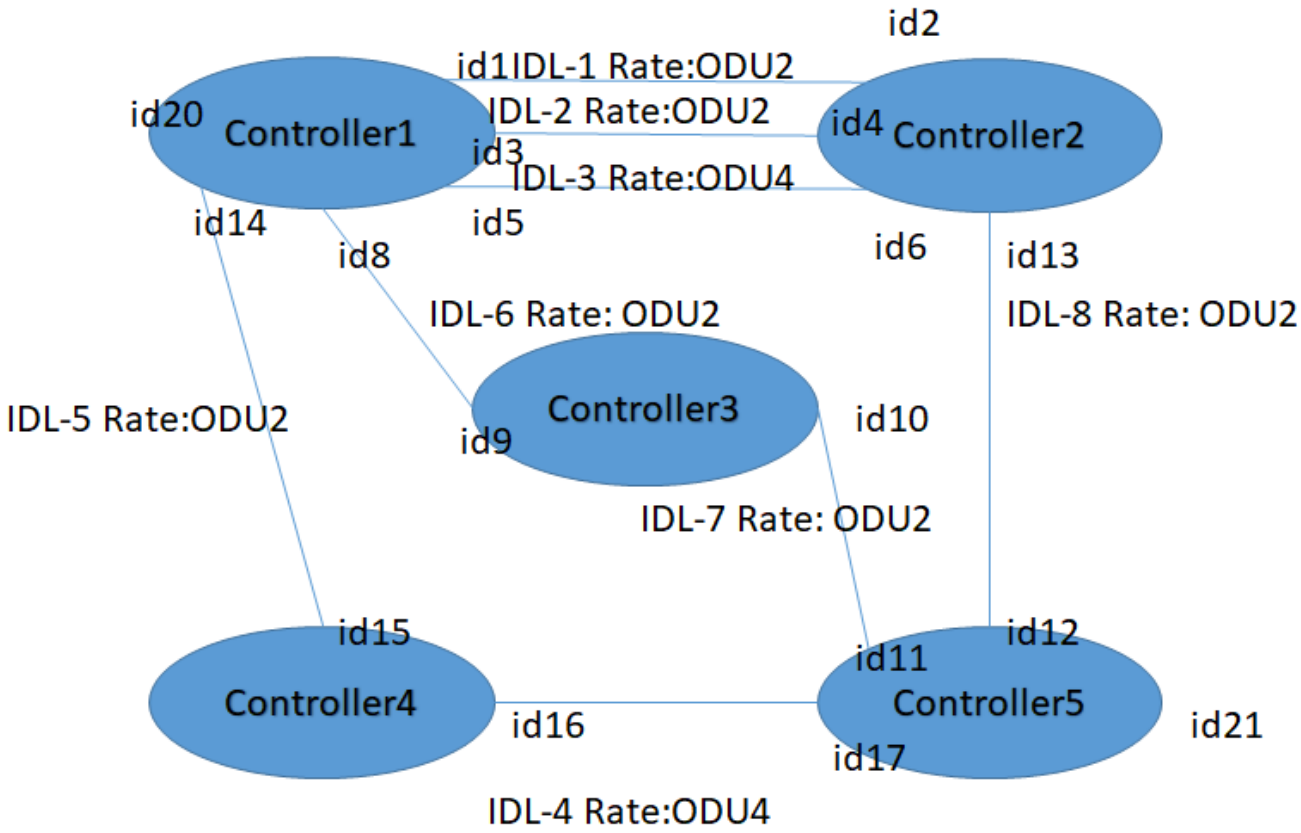
Here is the Minizinc Module used in IDL Path Optimizer.

Finding the shortest path - Minizinc

```
% Number of nodes
int: N;
    % Start node
1..N: Start;
    % End node
1..N: End;
    % Number of edges (directed arcs)
int: M;
    %  The actual edges
set of int: Edges = 1..M;
    % Edge lengths
array[Edges] of int: L;
    % Edge start node
array[Edges] of 1..N: Edge_Start;
array[Edges] of 1..N: Edge_End;

    % Variable indicating if edge is used
array[Edges] of var 0..1: x;

constraint
    forall( i in 1..N ) (
        if i = Start then
                % outgoing flow
            sum(e in Edges where Edge_Start[e] = i)(x[e]) -
                % incoming flow
            sum(e in Edges where Edge_End[e] = i)(x[e])
            = 1
        elseif i = End then
            sum(e in Edges where Edge_Start[e] = i)(x[e]) -
            sum(e in Edges where Edge_End[e] = i)(x[e])
            = -1
        else
            sum(e in Edges where Edge_Start[e] = i)(x[e]) -
            sum(e in Edges where Edge_End[e] = i)(x[e])
            = 0
        endif
    );


solve minimize sum(e in Edges)( L[e] * x[e] );

output ["Length: ", show(sum(e in Edges)(L[e] * x[e])), "\n"] ++
       ["Start : ", show(Start), "\n"] ++
       ["End   : ", show(End), "\n\n"] ++
       ["Edges in shortest path:\n"] ++
       [ if   fix(x[e]) = 1
         then show(Edge_Start[e]) ++ " -> " ++ show(Edge_End[e]) ++ "\n"
         else ""
         endif | e in Edges
       ];
```

Example Inter Domain Paths

Non Multiplexing Between the Domains



1. Route optimizer receives the service route request from SDNC with the source and destination interface ids.
2. The controllers of both the points are found using the AAI queries.
3. Then optimizer queries AAI to fetch all the possible Controllers in the inventory to send it to minizinc. It fetches all the inter-domain links from the inventory and filters it based on the "rate" mentioned in the request and the operational status "up".
4. Since minizinc expects the data in a certain way, optimizer wil encode all the data retrieved using the sckit module in python and send it to minizinc.
5. The pymzn module is used to interface python with the minizinc language.
6. Once minizinc chooses the optimized path, the optimizer will again decode the data and find the chosen links from the source to the destination.
7. The optimizer then would fetch the interface details from the link and form a proper service route list that would be sent back to SDNC along with the list of logical links.
8. SDNC in turn will fetch the end points from the service routes list object from the response and create as many domain services as present in the service route list.
9. The logical links in the response will be used to form the relationship with the access service which will be done by SDNC.
10. For the above example if a route has to be found between the "id20" of Controller1 "id21" of Controller5 following request will be sent to OOF by SDNC defined above.
11. After running the algorithm OOF wil return the following response defined above.

Multiplexing Between the Domains

TBD

AAI API Dependency

Query to retrieve the interdomain links for a given Controller:

Controller Inter-Domain-Link Query

```
API: PUT /aai/v19/query?format=resource

BODY:

{
"start" : ["external-system"],
"query" : "query/getInterDomainLink?linktype=inter-domain&controller=Virtuora-TAPI1"
}

RESPONSE EXAMPLE:

{
  "results": [{
    "logical-link": {
      "link-name": "IDLLink1",
      "in-maint": false,
      "link-type": "inter-domain",
      "resource-version": "1588952379221",
      "operational-status": "available",
      "relationship-list": {
        "relationship": [{
          "related-to": "p-interface",
          "relationship-label": "tosca.relationships.network.LinksTo",
          "related-link": "/aai/v16/network/pnfs/pnf/f17ae566-6cb9-3907-9bb1-00d04ca5e9d9/p-interfaces/p-
interface/641bb3d1-0817-3783-aba2-aa2129dfe1b1",
          "relationship-data": [{
            "relationship-key": "pnf.pnf-name",
            "relationship-value": "f17ae566-6cb9-3907-9bb1-00d04ca5e9d9"
          }, {
            "relationship-key": "p-interface.interface-name",
            "relationship-value": "641bb3d1-0817-3783-aba2-aa2129dfe1b1"
          }],
          "related-to-property": [{
            "property-key": "p-interface.prov-status"
          }]
        }, {
          "related-to": "p-interface",
          "relationship-label": "tosca.relationships.network.LinksTo",
          "related-link": "/aai/v16/network/pnfs/pnf/36bfbd31-e715-3312-9dd1-5a9e46d4b4d5/p-interfaces/p-
interface/3cfaf6b4-7923-3258-bb72-e2a62c40fe5e",
          "relationship-data": [{
            "relationship-key": "pnf.pnf-name",
            "relationship-value": "36bfbd31-e715-3312-9dd1-5a9e46d4b4d5"
          }, {
            "relationship-key": "p-interface.interface-name",
            "relationship-value": "3cfaf6b4-7923-3258-bb72-e2a62c40fe5e"
          }],
          "related-to-property": [{
            "property-key": "p-interface.prov-status"
          }]
        }]
      }
    }
  }]
}
```

Query to find the esr controller given a p-interface id:

P-interface Query

```
API: /aai/v19/query?format=resource

BODY:

{
"start" : ["external-system"],
"query" : "query/getDomainController?portid=49b3fb2a-6868-3a23-a833-79a3aa1dd24f"
}

RESPONSE EXAMPLE:

{
  "results": [{
    "esr-thirdparty-sdnc": {
      "thirdparty-sdnc-id": "Virtuora-TAPI2",
      "location": "Core",
      "product-name": "VirtuoraNetworkController",
      "resource-version": "1588951277460",
      "esr-system-info-list": {
        "esr-system-info": [{
          "esr-system-info-id": "Virtuora-TAPI2",
          "system-name": "Virtuora-TAPI2",
          "type": "TAPI",
          "vendor": "Fujitsu",
          "version": "V2",
          "service-url": "https://167.254.204.66:9443",
          "user-name": "admin",
          "password": "admin",
          "system-type": "Controller",
          "protocol": "RESTAPI",
          "ssl-cacert": "example-ssl-cacert-val-20589",
          "ssl-insecure": true,
          "ip-address": "167.254.204.66",
          "port": "9443",
          "cloud-domain": "example-cloud-domain-val-76077",
          "default-tenant": "example-default-tenant-val-71148",
          "passive": true,
          "remote-path": "example-remotepath-val-5833",
          "system-status": "example-system-status-val-23435",
          "resource-version": "1588951277460"
        }]
      },
      "relationship-list": {
        "relationship": [{
          "related-to": "network-resource",
          "relationship-label": "org.onap.relationships.inventory.BelongsTo",
          "related-link": "/aai/v16/network/network-resources/network-resource/Virtuora-TAPI2-TAPI-754a91dc-
dcd1-3530-8e95-a4880c298a1f",
          "relationship-data": [{
            "relationship-key": "network-resource.network-id",
            "relationship-value": "Virtuora-TAPI2-TAPI-754a91dc-dcd1-3530-8e95-a4880c298a1f"
          }],
          "related-to-property": [{
            "property-key": "network-resource.network-id",
            "property-value": "Virtuora-TAPI2-TAPI-754a91dc-dcd1-3530-8e95-a4880c298a1f"
          }]
        }]
      }
    }
  }]
}
```

API to retrieve all the inter-domain links:

Inter-domain Links retrieval

```
API: GET /aai/v19/logical-links?link-type=inter-domain

RESPONSE EXAMPLE:

{
  "logical-link": [{
    "link-name": "IDLLink1",
    "in-maint": false,
    "link-type": "inter-domain",
    "resource-version": "1588952379221",
    "operational-status": "available",
    "relationship-list": {
      "relationship": [{
        "related-to": "p-interface",
        "relationship-label": "tosca.relationships.network.LinksTo",
        "related-link": "/aai/v16/network/pnfs/pnf/f17ae566-6cb9-3907-9bb1-00d04ca5e9d9/p-interfaces/p-interface
/641bb3d1-0817-3783-aba2-aa2129dfe1b1",
        "relationship-data": [{
          "relationship-key": "pnf.pnf-name",
          "relationship-value": "f17ae566-6cb9-3907-9bb1-00d04ca5e9d9"
        }, {
          "relationship-key": "p-interface.interface-name",
          "relationship-value": "641bb3d1-0817-3783-aba2-aa2129dfe1b1"
        }],
        "related-to-property": [{
          "property-key": "p-interface.prov-status"
        }]
      }, {
        "related-to": "p-interface",
        "relationship-label": "tosca.relationships.network.LinksTo",
        "related-link": "/aai/v16/network/pnfs/pnf/36bfbd31-e715-3312-9dd1-5a9e46d4b4d5/p-interfaces/p-interface
/3cfaf6b4-7923-3258-bb72-e2a62c40fe5e",
        "relationship-data": [{
          "relationship-key": "pnf.pnf-name",
          "relationship-value": "36bfbd31-e715-3312-9dd1-5a9e46d4b4d5"
        }, {
          "relationship-key": "p-interface.interface-name",
          "relationship-value": "3cfaf6b4-7923-3258-bb72-e2a62c40fe5e"
        }],
        "related-to-property": [{
          "property-key": "p-interface.prov-status"
        }]
      }]
    }
  }]
}
```