CSEE 5690: Advanced Network Design and Optimization

Optimal Design of Virtual Networks for Resilient Cloud Services with and without MMF with fixed single path.

Amarnath Beedimane Hanumantharaya, MSEE.Swatesh Pakhare, MSEE.
School of computing and engineering
UMKC.

Abstract:

Resilience is the most important when you consider cloud infrastructure in this modern day and the network connecting the cloud service is the most important and we need to make sure that resilience of this network is reached faster. In this paper we are implementing the optimal design of virtual network resiliency and include Extension for that. We are going to implement this model using AMPLE.

A. INTRODUCTION :

Cloud customer mainly depended on performance and reliability, but reliability is the number one issue and performance is the number third issue according to customer.
In this paper we are going to implement a resilience model and implement a load balancing for the same. which increases reliability

B. RESILIENT VIRTUAL NETWORK DESIGN MODELS FOR CLOUD SERVICES

In this section we are going to implement our model using the below equations.

$$\sum_{c \in C} a_{s,c} = k \quad \forall s \in S$$

equation states that k belongs to server locations chosen for an anycast service with source s.
Link flow constraint is given by the equation

$$\sum_{l \in L: v \in E_l} \beta_{d,l} = \begin{cases} a_{s,c} & \text{if } v = s \text{ or } v = c \\ 2\delta_{d,v} & \text{otherwise} \end{cases}$$

$$\forall d = (s,c) \in D,\ v \in$$

$$\delta_{d,v} = a_{s,c} \quad \forall d = (s,c) \in D,\ v \in \{s,c\}$$

Above equation says that ensures that a node is flagged as ”used” for a service if it is the source or the target of that service and if it is chosen as a realization of the anycast service with source s.
Below equation states that if a virtual link, node or
virtual machine (VM) carries the traffic of any service, it
is part of the resulting VNet, otherwise not.

$$\gamma_l \geq \beta_{d,l} \quad \forall l \in L,\ d \in D$$

$$\alpha_v \geq \delta_{d,v} \quad \forall v \in V,\ d \in D$$

$$y_c \geq a_{s,c} \quad \forall c \in C,\ s \in S$$

Below equations provide upper bounds for γ l, α v and yc,
respectively, which ensures that a virtual link, node or VM
to be part of the resulting VNet only if it is actually used.
These bounds are only necessary for calculating the VNet cost
in delay optimization to obtain meaningful cost values but do
not restrict the optimality. And to calculate the required virtual link, node and VM capacities.

And VNO resilience realization for single service

$$\gamma_l \leq \sum_{d \in D} \beta_{d,l} \quad \forall l \in L$$

$$\alpha_v \leq \sum_{d \in D} \delta_{d,v} \quad \forall v \in V$$

$$y_c \leq \sum_{s \in S} a_{s,c} \quad \forall c \in C$$

$$u_l \geq \sum_{d \in D} \beta_{d,l}\, b_d \quad \forall l \in L$$

$$\omega_v \geq \sum_{d \in D} \delta_{d,v}\, n_d \quad \forall v \in V$$

$$z_c \geq \sum_{s \in S} a_{s,c}\, r_d \quad \forall c \in C \text{ with } d = (s, c$$

And finally we minimize the cost and delay using the equation

$$\min\ \varepsilon,\ \varepsilon = \sum_{l \in L} \varepsilon_l + \sum_{v \in V} \varepsilon_v + \sum_{c \in C} \varepsilon_c$$

$$\min \quad \sum_{d \in D} \sum_{l \in L} \beta_{d,l}\, t_l$$

C. VNO-RESILIENCE:

In vnet model we are considering the k=3 which means that we are routing in virtual layer in 3 different server location. Both the VMs and the paths leading to the VMs have to
be physically disjoint, such that in case of a failure at the
primary site, the DR site can be used by re-routing the
service inside the VNet. Hence, we need to add the diversity
constraints for these paths and VMs to the model. Below equations are the diversity constraints for link and node-diversity respectively for the connection paths.

$$\beta_{d_1,l} + \beta_{d_2,k} \leq 1 \quad \forall s \in S,\ (d_1, d_2) \in D_s^2,\ (l, k) \in Z$$

$$\delta_{d_1,v_1} + \delta_{d_2,v_2} \leq 1 \quad \forall s \in S,\ (d_1, d_2) \in D_s^2,\ (v_1, v_2) \in (V \setminus \{s\})^2$$

Equation ensures that the primary
and backup sites are located in different availability regions.

is shown below

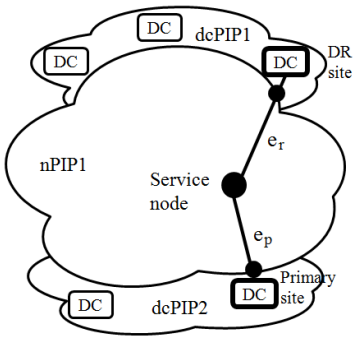

Figure 1

A. MODEL for VNET:

Below diagram is the model which we are considering for this paper. Here we are considering DC1 and DC2 as datacenter, C1,C2 and C3 as connection nodes and S1,S2 and S3 as service nodes. And each data center is connected to single connection nodes.

We made certain assumptions that For set C, we have assumed that two nodes connected to single DC are DC connection nodes. These DC connected nodes are further assumed as VMs in a virtual layer.

The dotted lines are virtual link in the vnet and hard lines are physical links. we have total of 12 virtual links.

As shown in the figure 1

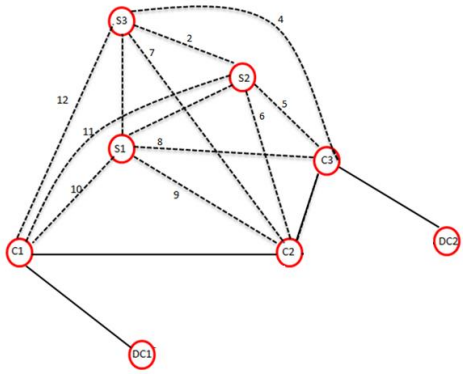

Figure 2

We implemented this vnet model and formulated

$$a_{s,c_1} + a_{s,c_2} \leq 1 \quad \forall s \in S,\ (c_1, c_2) \in R$$

using ample.

We found that with MMF objective function which was cost, according to equations we got total cost as 765.

For demand of 9. But when we take demand equal to 4 then the cost would be 405. And propagation delay found out to be zero in the both cases.

Since we didn't find any extension we removed some of the constraint's which is related to MMF and found out to be cost decreases drastically we found that for demand 9 cost is 45 and for demand 4 cost is 45.

And cost for demand 9 and 4 is same .

D. CHALLENGES FACED:

I. We had to struggle with constraint 2 in the formulation. I guess we did not know how to logically write the constraints for summation in constraint 2. We did not understand the presence of 'v' in condition (RHS) for this constraint. So, we decided to just follow the constraint as given.

Initially, we got this error.

```
presolve, constraint server_selection[1]:
        all variables eliminated, but lower bound = 3 > 0
presolve, constraint server_selection[2]:
        all variables eliminated, but lower bound = 3 > 0
presolve, constraint server_selection[3]:
        all variables eliminated, but lower bound = 3 > 0
ampl:
```

This error was resolved by making some syntactical changes for constraint 2. We spent like couple of days in debugging this error.

II. When we take k=3, we get the value for objective (minimizing the cost). But, for k=1 and k=2, we had to face this issue.

```
ampl: option solver cplex;
ampl: solve;

Sorry, the student edition of AMPL is limited to 300 variables
and 300 constraints and objectives (after presolve) for nonlinear
problems.  You have 213 variables, 1212 constraints, and 2 objectiv
ampl:
```

We tried to solve this problem in CPLEX. Apparently, . For one of the objective, minimizing the propagation delay, we did not get any solution for k=1, 2 and 3.

3.During VNO resilence, since we used k=3 we had to make certain changes to the resilence equation this also took like 1 day to figure out what the error was and equation was.

E. INDEX:

Ample formulation for VNET:

```
param S > 0 integer; #servive nodes
param C > 0 integer; # DC connection
nodes
param V > 0 integer; # virtual nodes
param L > 0 integer; #Virtual links
param D > 0 integer; #demand
param El > 0 integer; # Endpoints of
virtual link
param Z > 0 integer; # set of virtual
link pairs
param E > 0 integer; # Edges in the
pgysical network
param N > 0 integer; #Nodes in the
physical network
param Pl > 0 integer; #set of physical
edges on which virtual link is mapped
param R > 0; # DC connection node pair

set service_nodes := 1..S;
set dc_connect_nodes := 1..C;
set virtual_nodes := 1..V;
set virtual_links := 1..L;
set demand_nos := 1..D;
set virtual_links_end := 1..El;
set virtual_links_pair := 1..Z;
set link_nos := 1..E;
set Nodes := 1..N;
set vir_phy_link := 1..Pl;
set dc_connect_nodes_pair := 1..R;


#Generation of virtual links
param v_src {virtual_links} within
Nodes;
param v_dest {virtual_links} within
Nodes;


#Generation of demand
param src  {demand_nos} within Nodes;
```

we did not get any solution.

```
param dest {demand_nos} within Nodes;
#Generation of Routes

#Parameters

param k=3;
param b {d in demand_nos} >= 0 integer;
param n {d in demand_nos}>= 0 integer;
param r {d in demand_nos}>= 0 integer;
param t {l in virtual_links} >= 0 integer;
param lambda {l in virtual_links}>= 0
integer;
param theta {l in virtual_links}>= 0
integer;
param mu {v in virtual_nodes}>= 0 integer;
param eta {v in virtual_nodes}>= 0 integer;
param phi {c in dc_connect_nodes}>= 0
integer;
param psi {c in dc_connect_nodes}>= 0
integer;

var a {s in service_nodes, c in
dc_connect_nodes} >=0 binary;


var temp {s in service_nodes, c in
dc_connect_nodes, d in demand_nos, l in
virtual_links} >=0 ;

var beta { d in demand_nos, l in
virtual_links}>=0 binary;
var delta { d in demand_nos, v in
virtual_nodes} >=0 binary;
var gamma {l in virtual_links} >=0 binary;
var alpha {v in virtual_nodes} >=0 binary;
var y {c in dc_connect_nodes} >=0 binary;

var u {l in virtual_links};
var w {v in virtual_nodes};
var z {c in dc_connect_nodes};

#Objective

minimize Cost: sum{l in
virtual_links}((lambda[l]*gamma[l])+
(theta[l]*u[l]))
   + sum{v in
virtual_nodes}((mu[v]*alpha[v])+
(eta[v]*w[v]))
   + sum{c in
dc_connect_nodes}((phi[c]*y[c])+
(psi[c]*z[c]));

(sum {l in virtual_links}
beta[d,l]*t[l]);

#constraints

subj to server_selection {s in
service_nodes} :
      sum {c in dc_connect_nodes}
a[s,c] =k;



subj to link_flow {d in demand_nos,v in
virtual_nodes,s in service_nodes,c in
dc_connect_nodes}:
      sum { l in virtual_links : v in
virtual_nodes} beta[d,l]
          = if (v=s ) then a[s,c]  else
2*delta[d,v]
      or
       if (v=c) then a[s,c]  else
2*delta[d,v];
subj to node_is_flagged_as_used {d in
demand_nos, v in virtual_nodes,c in
dc_connect_nodes,s in service_nodes} :
      delta[d,v]=a[s,c];

subj to virtual_link_carrying_traffic {
      l in virtual_links, d in
demand_nos}:
      gamma[l] >= beta[d,l];

subj to virtual_node_carrying_traffic
{v in virtual_nodes,d in demand_nos}:
      alpha[v] >=delta[d,v];

subj to
virtual_machine_carrying_traffic { c in
dc_connect_nodes,s in service_nodes}:
      y[c] >= a[s,c];

subj to upper_bound_virtual_links {l in
virtual_links}:
      sum {d in demand_nos} beta[d,l]
>= gamma[l];

subj to upper_bound_virtual_nodes {v in
virtual_nodes}:
      sum {d in demand_nos} delta[d,v]
>= alpha[v];

subj to upper_bound_virtual_machine {c
in dc_connect_nodes}:
      sum {s in service_nodes} a[s,c]
>= y[c];
```

```
minimize Prop_Delay : sum {d in demand_nos}
subj to virtual_link_capacity {l in
virtual_links}:

      sum {d in demand_nos} beta[d,l]*b[d]
<=u[l];

subj to virtual_node_capacity {v in
virtual_nodes}:
      sum {d in demand_nos}
delta[d,v]*n[d] <=w[v];

subj to virtual_machine_capacity {c in
dc_connect_nodes, d in demand_nos}:
      sum {s in service_nodes} a[s,c]*r[d]
<=z[c];


                  # VNO resilience

subj to diversity_link { d in demand_nos,l
in virtual_links}:

      beta[1,l] + beta[1,k] - 1 <= 0;
subj to diversity_node {d in demand_nos, v
in virtual_nodes}:

      delta[1,1] + delta[1,2] -2 <= 0;

subj to diversity_loc {s in service_nodes,c
in dc_connect_nodes}:
    a[s,1] +a[s,2] + a[s,3] -3 <= 0;
```

Ample formulation for VNET without MMF:

```
param S > 0 integer; #servive nodes
param C > 0 integer; # DC connection nodes
param V > 0 integer; # virtual nodes
param L > 0 integer; #Virtual links
param D > 0 integer; #demand
param El > 0 integer; # Endpoints of
virtual link
param Z > 0 integer; # set of virtual link
pairs
param E > 0 integer; # Edges in the
pgysical network
param N > 0 integer; #Nodes in the physical
network
param Pl > 0 integer; #set of physical
edges on which virtual link is mapped
param R > 0; # DC connection node pair


set service_nodes := 1..S;
set virtual_nodes := 1..V;
set virtual_links := 1..L;
set demand_nos := 1..D;
set virtual_links_end := 1..El;
set virtual_links_pair := 1..Z;
set link_nos := 1..E;
set Nodes := 1..N;
set vir_phy_link := 1..Pl;
set dc_connect_nodes_pair := 1..R;


#Generation of virtual links
param v_src {virtual_links} within
Nodes;
param v_dest {virtual_links} within
Nodes;


#Generation of demand


param src  {demand_nos} within Nodes;
param dest {demand_nos} within Nodes;


#Generation of Routes

#Parameters

param k=3;
param b {d in demand_nos} >= 0 integer;
param n {d in demand_nos}>= 0 integer;
param r {d in demand_nos}>= 0 integer;
param t {l in virtual_links} >= 0
integer;
param lambda {l in virtual_links}>= 0
integer;
param theta {l in virtual_links}>= 0
integer;
param mu {v in virtual_nodes}>= 0
integer;
param eta {v in virtual_nodes}>= 0
integer;
param phi {c in dc_connect_nodes}>= 0
integer;
param psi {c in dc_connect_nodes}>= 0
integer;

var a {s in service_nodes, c in
dc_connect_nodes} >=0 binary;


var temp {s in service_nodes, c in
dc_connect_nodes, d in demand_nos, l in
virtual_links} >=0 ;
```

```
set dc_connect_nodes := 1..C;
var beta { d in demand_nos, l in
virtual_links}>=0 binary;
var delta { d in demand_nos, v in
virtual_nodes} >=0 binary;
var gamma {l in virtual_links} >=0 binary;
var alpha {v in virtual_nodes} >=0 binary;
var y {c in dc_connect_nodes} >=0 binary;


var u {l in virtual_links};
var w {v in virtual_nodes};
var z {c in dc_connect_nodes};




#Objective

minimize Cost: sum{l in
virtual_links}((lambda[l]*gamma[l])+
(theta[l]*u[l]))
   + sum{v in
virtual_nodes}((mu[v]*alpha[v])+
(eta[v]*w[v]))
   + sum{c in
dc_connect_nodes}((phi[c]*y[c])+
(psi[c]*z[c]));


minimize Prop_Delay : sum {d in demand_nos}
(sum {l in virtual_links} beta[d,l]*t[l]);

#constraints

subj to server_selection {s in
service_nodes} :
      sum {c in dc_connect_nodes} a[s,c]
=k;




subj to link_flow {d in demand_nos,v in
virtual_nodes,s in service_nodes,c in
dc_connect_nodes}:
      sum { l in virtual_links : v in
virtual_nodes} beta[d,l]
          = if (v=s ) then a[s,c]  else
2*delta[d,v]
      or
       if (v=c) then a[s,c]  else
2*delta[d,v];
subj to node_is_flagged_as_used {d in
dc_connect_nodes,s in service_nodes} :
      delta[d,v]=a[s,c];

subj to virtual_link_carrying_traffic {
      l in virtual_links, d in
demand_nos}:
      gamma[l] >= beta[d,l];

subj to virtual_node_carrying_traffic
{v in virtual_nodes,d in demand_nos}:
      alpha[v] >=delta[d,v];

subj to
virtual_machine_carrying_traffic { c in
dc_connect_nodes,s in service_nodes}:
      y[c] >= a[s,c];

subj to upper_bound_virtual_links {l in
virtual_links}:
      sum {d in demand_nos} beta[d,l]
>= gamma[l];

subj to upper_bound_virtual_nodes {v in
virtual_nodes}:
      sum {d in demand_nos} delta[d,v]
>= alpha[v];

subj to upper_bound_virtual_machine {c
in dc_connect_nodes}:
      sum {s in service_nodes} a[s,c]
>= y[c];

#subj to virtual_link_capacity {l in
virtual_links}:
#      sum {d in demand_nos}
beta[d,l]*b[d] <=u[l];

subj to virtual_node_capacity {v in
virtual_nodes}:
      sum {d in demand_nos}
delta[d,v]*n[d] <=w[v];
subj to virtual_machine_capacity {c in
dc_connect_nodes, d in demand_nos}:
      sum {s in service_nodes}
a[s,c]*r[d] <=z[c];
                    # VNO resilience

subj to diversity_link { d in
demand_nos,l in virtual_links}:

      beta[1,l] + beta[1,k] - 1 <= 0;
subj to diversity_node {d in
demand_nos, v in virtual_nodes}:

      delta[1,1] + delta[1,2] -2 <= 0;
```

```
demand_nos, v in virtual_nodes,c in

subj to diversity_loc {s in service_nodes,c
in dc_connect_nodes}:
    a[s,1] +a[s,2] + a[s,3] -3 <= 0;
```

and for datafile is to be:

```
data;
param S=3;
param C=3 ;
param V=6;  # virtual nodes
param L=12; #Virtual links
param D = 4; #demand
param El=2; # Endpoints of virtual link
#param Z = 0; # set of virtual link pairs
param E=8 ; # Edges in the pgysical network
param N=6 ; #Nodes in the physical network
#param Pl > 0 integer; #set of physical
edges on which virtual link is mapped
param R=2; # DC connection node pair
#param k=1;

 param:             v_src   v_dest  lambda
theta t  :=   #for virtual link
     1                4      6       2
3   1
     2                5      6       2
3   1
     3                4      5       2
3   1
     4                3      6       2
3   1
     5                3      5       2
3   1
     6                2      5       2
3   1
     7                2      6       2
3   1
     8                3      4       2
3   1
     9                2      4       2
3   1
     10               1      4       2
3   1
     11               1      5       2
3   1
     12               1      6       2
3   1   ;



param:     src  dest   b    n     r
:=  #for demand d(s,c)
  1         1    4     5   2    1
  3         3    4     5   2    1
  4         1    5     5   2    1
  5         2    5     5   2    1
  6         3    5     5   2    1
  7         1    6     5   2    1
  8         2    6     5   2    1
  9         3    6     5   2    1
;


  param: mu   eta   := #for virtual
nodes
     1   4    6
     2   4    6
     3   4    6
     4   4    6
     5   4    6
     6   4    6    ;


     param:  phi  psi  :=
       1      7   8
       2      7   8
       3      7   8;

 end;
```

2 2 4 5 2 1