Performance Analysis of Data Centre Network (DCN) Architectures using Virtual Network Embedded Simulator (VNE-Sim)

ENSC894: Communication Networks

Spring 2020

Project Group: Team 1

Project Website: https://tehreemf.wixsite.com/vne-sim

Team Members:

Lucy Malsawmtluangi
lmalsawm@sfu.ca

Tehreem Naeem
tehreemf@sfu.ca

Amandeep Singh Bhogal
abhogal@sfu.ca

Afolabi David Abioye
afolabi_abioye@sfu.ca

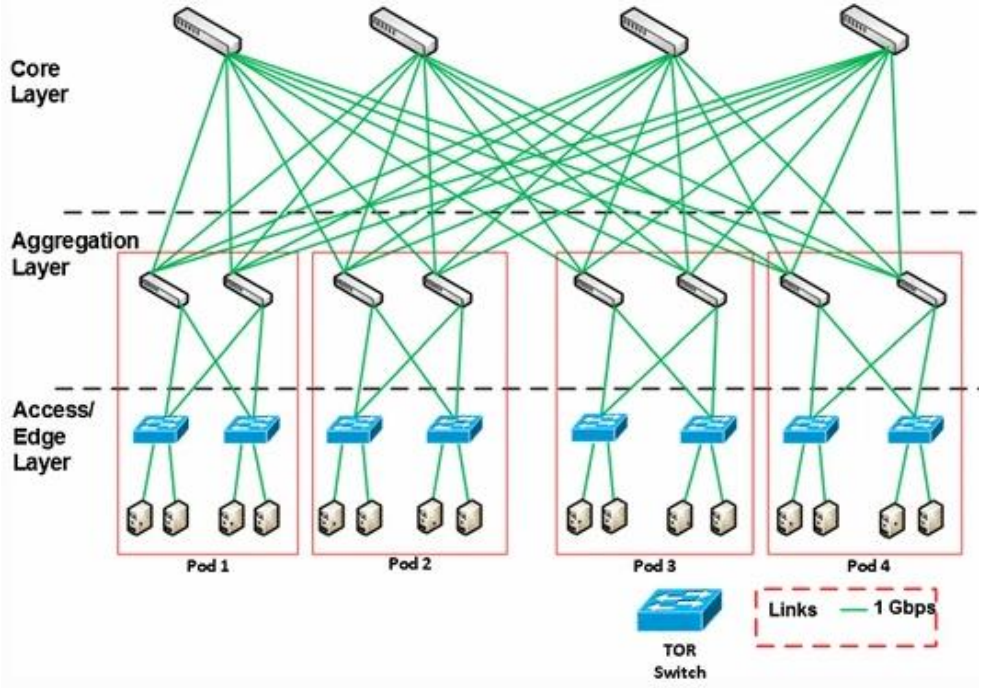
Overview

Introduction

- Software Defined Network (SDN)
  - Segregation of Control Plane and Data Plane in Network Layer (Layer-3)
  - Software abstraction of management layer
  - Intelligent programmable function to control underlying components
  - Internet of Things (IoT), Cloud integration and services
- Network Function Virtualization (NFV)
  - Software abstraction of network functions from hardware
  - Network services such as routers, firewall, load balancer, *etc.* can be replace by software running in virtual machines

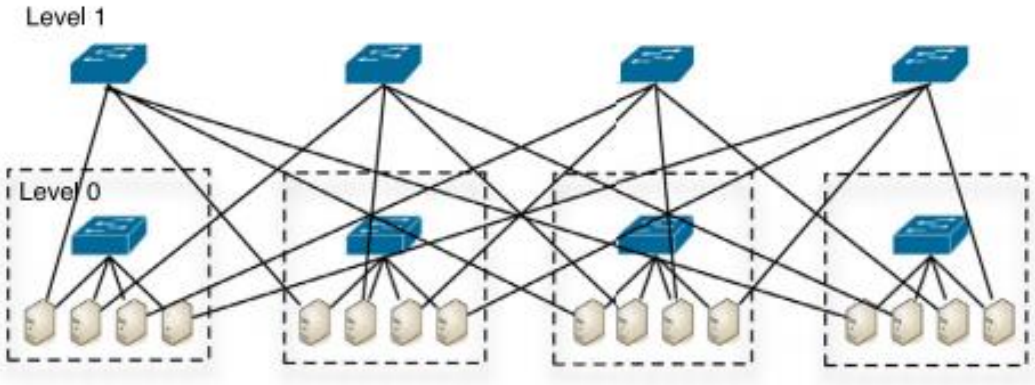
Introduction – Datacenter Topologies

Switch Centric
Clos Architecture - Fat Tree [1]



- Based on Two-tier and three-tier topologies

Server Centric - BCube [2]



- Recursively connect the cells and components

Related Work

- S. Haeri. (2019) Vne-sim: A virtual network embedding simulator. Available: http://www.sfu.ca/~ljilja/cnl/projects/VNE-Sim/vne-sim-web/index.html.
  - This paper specifically worked on VNE-SIM and descriptive guides to working with the simulator
- Y. Liu, J. K. Muppala, M. Veeraraghavan, D. Lin, and M. Hamdi, Data center networks:Topologies, architectures and fault-tolerance characteristics. Springer Science & Business Media, 2013.
  - In this research work, different DCN topologies were shown and explained in absence of Dcell.
- H. Ben Yedder, Q. Ding, U. Zakia, Z. Li, S. Haeri, and L. Trajkovic, “Comparison of virtualization algorithms and topologies for data center networks,” in 2017 26th International Conference on Computer Communication and Networks (ICCCN), July 2017, pp. 1–6.
  - This work is the very much related to our work but was done in absence of Dcell topology to evaluate acceptance ratio using GRC, GRC-M, R.VINE AND D-VINE algorithms.

Motivation and Goals

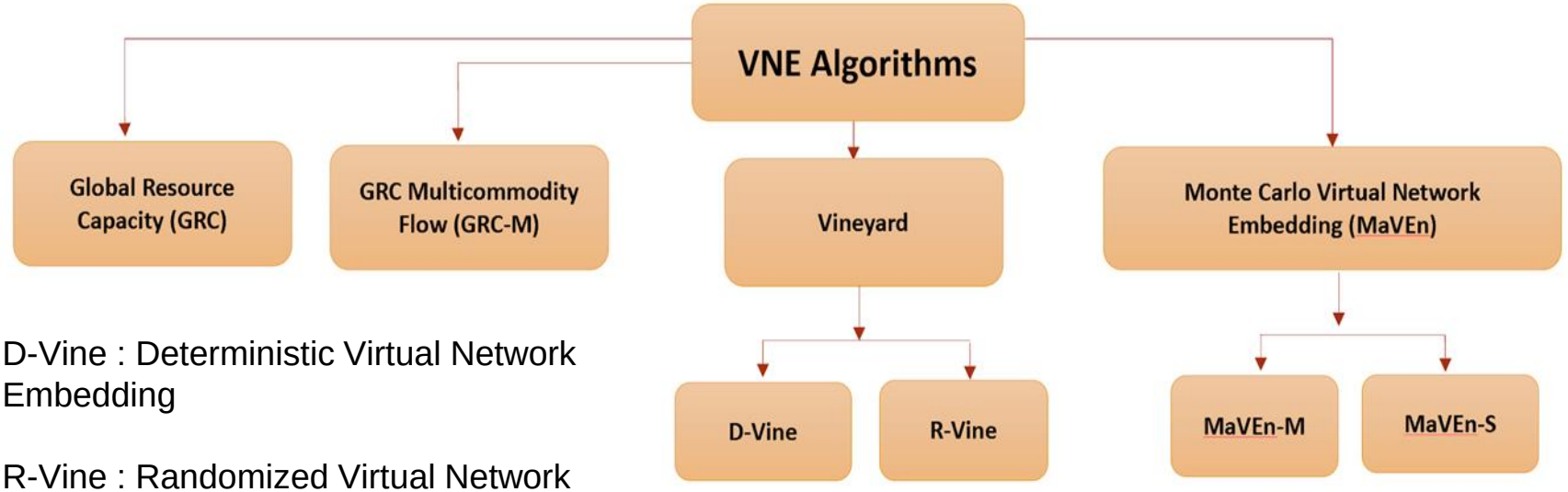
- In recent times, SDN has gained strides in terms of its application over the use of conventional form of networking. Reduced operating cost is a strong factor attached to SDN and this is the motivating force behind this work.
- Moreover, our goals are:
  - Understanding modelling with VNE-SIM for networks.
  - To generate different DCN topologies using BRITE and FNSS
  - To analyze the performance of VNE algorithms in terms of acceptance ratio on DCN topologies using Hiberlite.

Acknowledgement

Our Project team is using VNE-Sim for testing the performance of DCN Topologies which was developed by Dr. Sourosh Haeri for his Ph.D Thesis under the guidance of Dr. Ljiljana Trajkovic at the Communication Networks Laboratory (CNL). We are very thankful to both of them for giving us an opportunity to use VNE-Sim for our project and for the knowledge that we have gained from this project.

VNE-SIM Algorithms

§**Goal**: To efficiently allocates the resources to Virtual Nodes and Virtual Links



D-Vine : Deterministic Virtual Network Embedding

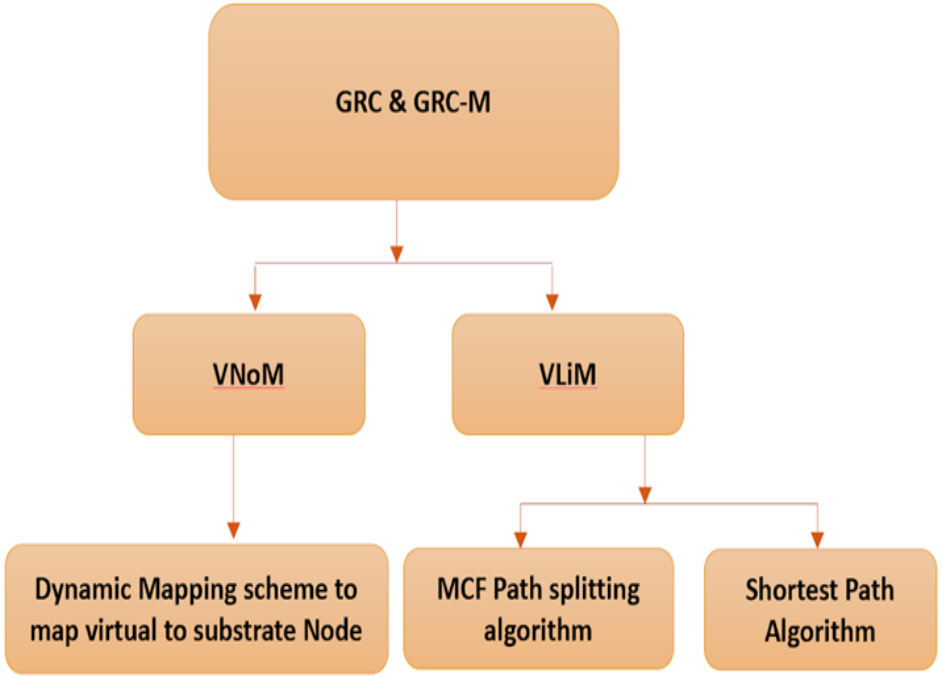
R-Vine : Randomized Virtual Network Embedding

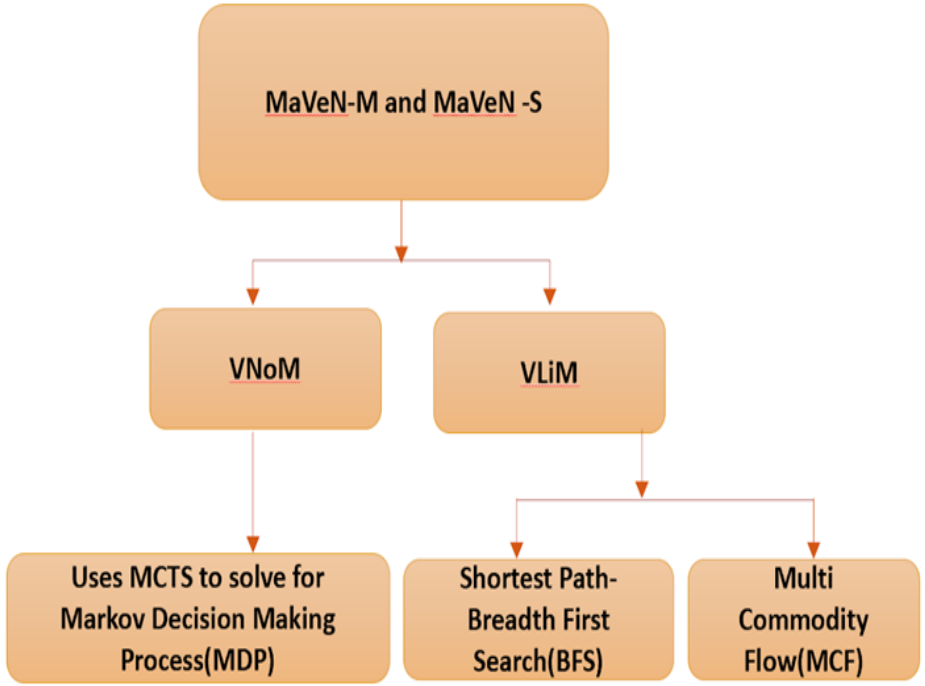
MaVEn-M : Monte Carlo Virtual Network Embedding with Multimedia Flow

MaVEn-S : Monte Carlo Virtual Network Embedding with Shortest Path

M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206—219, Feb. 2012.
L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1—9.
S. Haeri, Q. Ding, Z. Li, and Lj. Trajković, "Global resource capacity algorithm with path splitting for virtual network embedding," in *Proc. IEEE Int. Symp. Circuits Syst.*, Montreal, Canada, May 2016, pp. 666–669.
S. Haeri and Lj. Trajkovic, "Virtual network embedding via Monte-Carlo tree search," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 1—12, Feb. 2017.

VNoM & VLiM

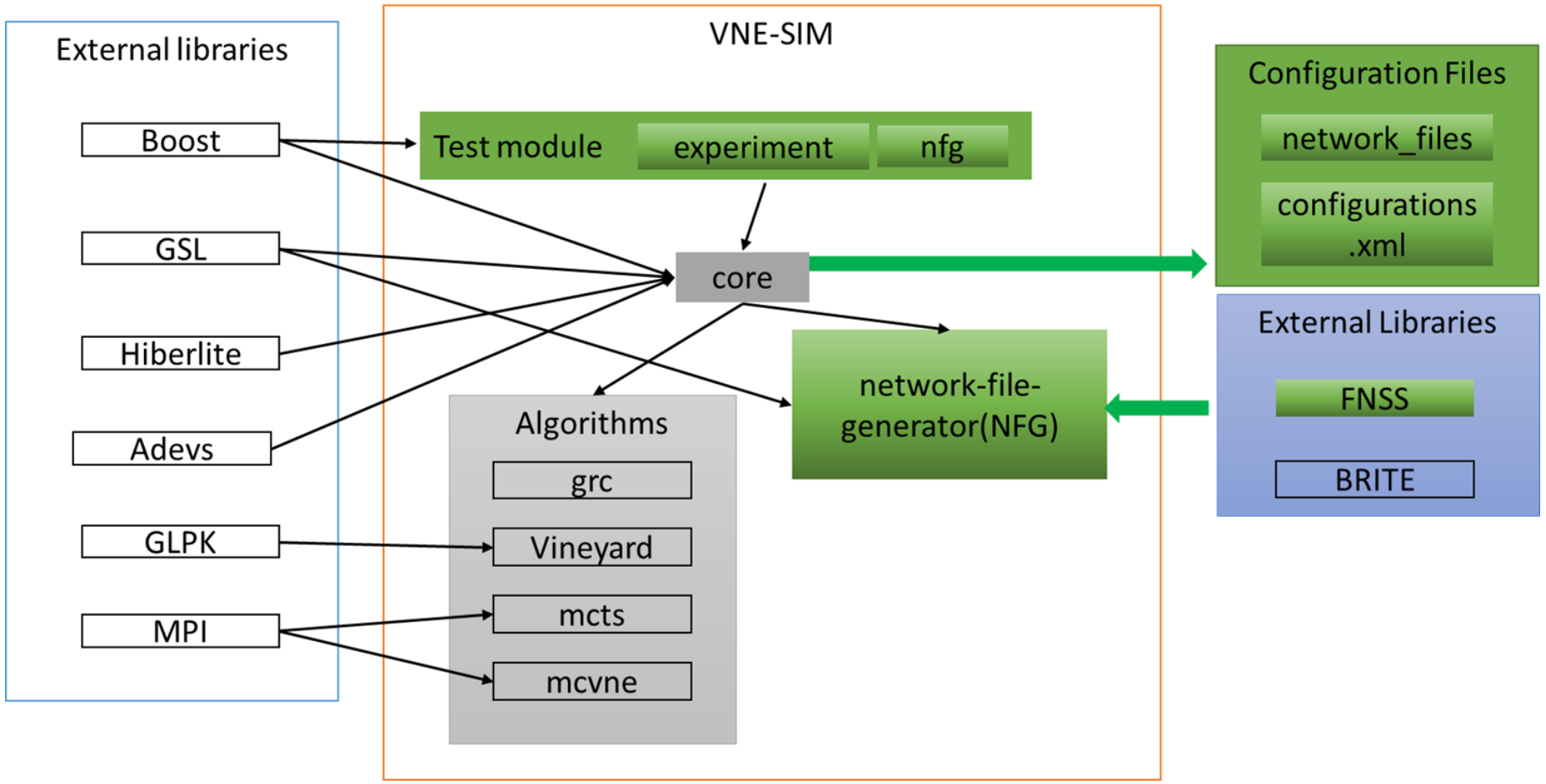




L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1–9.
S. Haeri, Q. Ding, Z. Li, and Lj. Trajković, "Global resource capacity algorithm with path splitting for virtual network embedding," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Montreal, Canada, May 2016, pp. 666—669.

S. Haeri and Lj. Trajkovic, "Virtual network embedding via Monte-Carlo tree search," *IEEE Trans. on Cybern.*, vol. 47, no. 2, pp. 1—12, Feb. 2017.

Implementation



High Level Architecture showing the inter-dependencies of the module[$]

$ Reimaged from : S. Haeri. (2019) Vne-sim: A virtual network embedding simulator. Accessed: 2020-04-06. [Online]. Available: http://www.sfu.ca/~ljilja/cnl/projects/VNE-Sim/vne-sim-web/index.html

Implementation

VNE-Sim High Level Architecture:

- **Core library**: Synchronizes and manages the calls between the modules. Uses GNU Scientific Library (GSL)
- **Algorithms**: Handles the call for embedding using the algorithms Monte Carlo Virtual Network Embedding (MaVEn), Global Resource Capacity (GRC), Vineyard. Uses GNU Linear Programming Kit (GLPK) and Message Passing Interface (MPI)
- **Network File Generator** : Uses Boston University Representative Internet Topology Generator (BRITE) and Fast Network Simulation Setup (FNSS)
- **Unit Test** : Using Boost Unit Test Framework for various simulation scenarios.

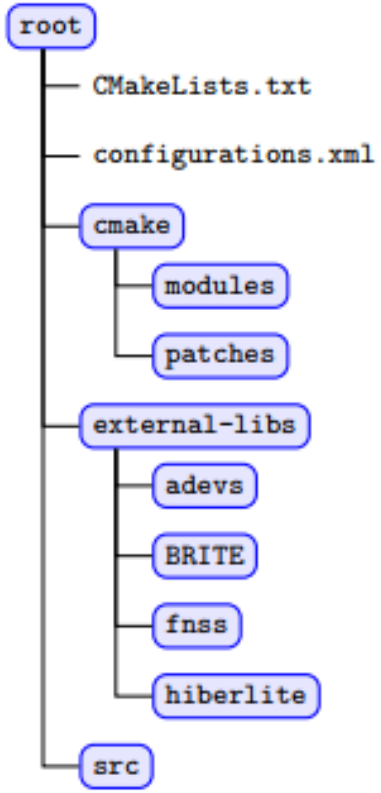
Implementation - Tools

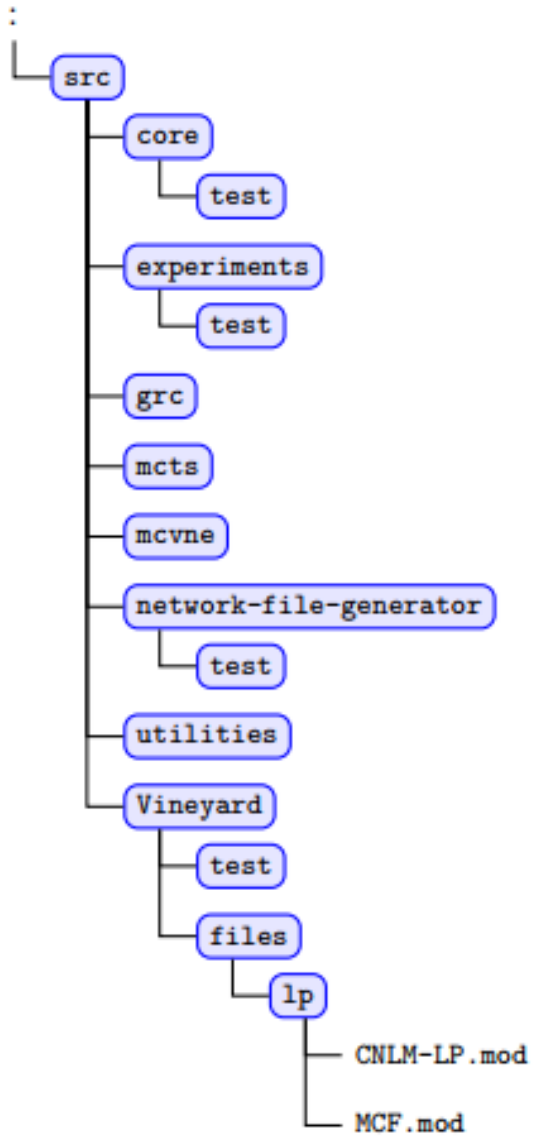
- ***Boost Library*** : C++ library for managing file handling, testing, and logging
  - [Accessed on: 6-Apr-2020] https://www.boost.org/
- ***GNU Scientific Library (GSL)*** : Numerical library for C and C++ for generating random numbers
  - [Accessed on: 6-Apr-2020] https://www.gnu.org/software/gsl/
- ***GNU Linear Programming Kit (GLPK)*** : C package for large scale linear programming used in VNE algorithms
  - [Accessed on: 6-Apr-2020] https://www.gnu.org/software/glpk/
- ***Message Passing Interface (MPI)*** :Message passing library standard for parallelized computing such as MCTS (np-hard)
  - [Accessed on: 6-Apr-2020] https://www.open-mpi.org/
- ***Adevs library*** : C++ library for modelling the virtual network embedding process as a discrete event system
  - [Accessed on: 6-Apr-2020] https://web.ornl.gov/~nutarojj/adevs/

Implementation - Tools

- ***Boston University Representative Internet Topology Generator (BRITE)*** : Toolchain for large scale topology generation for VNR
  - [Accessed on: 6-Apr-2020] https://www.cs.bu.edu/brite/
- ***Fast Network Simulation Setup (FNSS)*** : Toolchain for creating network scenario especially DCN topology for substrate networks
  - [Accessed on: 6-Apr-2020] https://fnss.github.io/
- ***SQLite3*** : Simple Query Language (SQL) database engine used for handling simulation results
  - [Accessed on: 6-Apr-2020] https://www.sqlite.org/index.html
- ***DB Browser*** : Used for reading and exporting CSV format for database files compatible with SQLite
  - [Accessed on: 6-Apr-2020] https://sqlitebrowser.org/
- ***Hiberlite*** : C++ library for object relational mapping
  - [Accessed on: 6-Apr-2020] https://github.com/paulftw/hiberlite

Implementation - VNE-Sim Enhancement





Hirarchy of VNE-SIM Codebase[#]

#S. Haeri. (2019) Vne-sim: A virtual network embedding simulator. Accessed: 2020-04-06. [Online]. Available: http://www.sfu.ca/~ljilja/cnl/projects/VNE-Sim/vne-sim-web/index.html

Implementation - VNE-Sim Enhancement

- Customize local FNSS Python library to support DCell Topology, file changes:
  - ./fnss/topologies/datacenter.py

DatacenterTopology type: dcell_topology

```
def dcell_topology(t, k):
 Parameters
    k : int
        The level of DCell
    t : int
        The number of host per :math:`DCell_0`
```

Implementation - VNE-Sim Enhancement

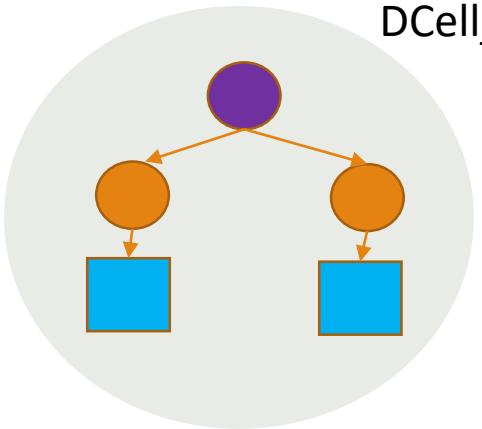
- Extend VNE-Sim by adding DCell Topology to the current implementation file changes:
  - ./src/configuration.xml
  - ./src/network-file-generator/fnss-handler.h
  - ./src/network-file-generator/ network-file-generator.cc
  - ./src/network-file-generator/test/network-file-generator-test.cc
  - ./src/core/experiment-parameters.cc
  - ./src/experiments/test/experiments-test.cc
  - Disable BOOST debug logs to improve the computational speed

Implementation – Dcell[3][7] Example

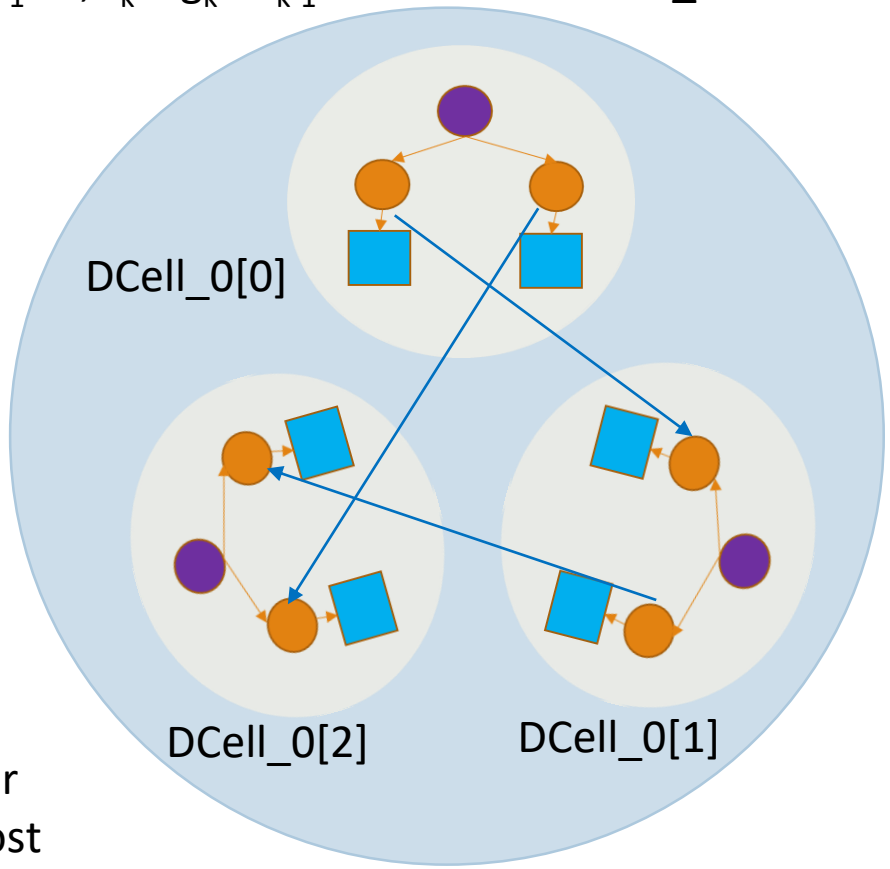
t = Number of servers at Dcell_0 = 2
k = Number of levels in DCell = 2

t_k = Number of servers at Dcell_k
g_k = Number of levels in DCell _k
if k == 0, $g_0 = 1$, $t_0 = t$
if k > 0, $g_k = t_{k-1} + 1$, $t_k = g_k * t_{k-1}$

DCell_0



DCell_1



Mini-switch

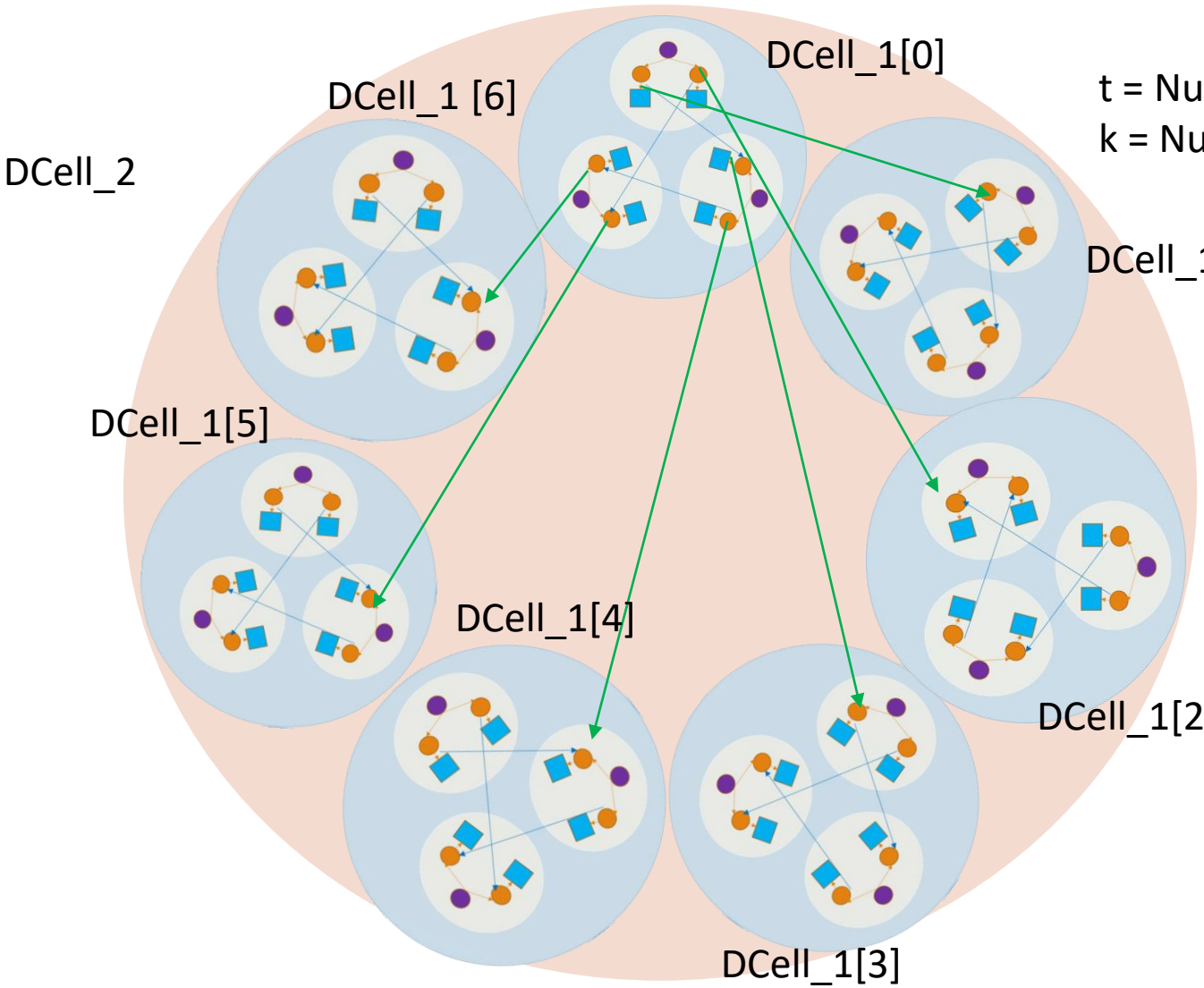
Server-switch

Hosts

k	t_k	g_k
0	2	1
1	6	3

FNSS uses networkx Python library[6] and for server are represented using a combination switch and host

Implementation – Dcell[3][7] Example



t = Number of servers at Dcell_0 = 2
k = Number of levels in DCell = 2

k	t_k	g_k
0	2	1
1	6	3
2	42	7

Simulation Scenarios

- During the development phase of VNE-Sim, we have performed various test cases and small scale validation scenarios to ensure the correctness of the implemented algorithms.
- The BRITE library is used to generate the substrate and VNR graphs.
- Performance of the ViNE and the GRC algorithms in terms of revenue, cost, revenue to cost ratio, acceptance ratio, node utilization, and link utilization are employed.
- Unprocessed data generated by discrete events that occur during simulations are saved as SQLite databases.
- These data may be processed using SQL commands or other statistical analysis tools.

Simulation Scenarios

- The output database of the simulation results indicate the outcome of the embedding for every VNR arrival. Successfully embedded VNR shows revenue, cost, and queuing and processing times of the request.
- The database entries are processed to calculate revenue, cost, and resource utilizations.
- The R-ViNE and D-ViNE algorithms utilize the substrate link resources more efficiently than the GRC algorithm as a result of employing the MCF algorithm for link mapping.
- GRC algorithm employs a shortest-path-based algorithm.
- We compared the Dcell, BCube and Fat-Tree data center network topologies for virtual network embedding by employing the R-ViNE, D-ViNE, and GRC algorithms . Simulation results indicate that the DCell topology is capable of accepting additional number of virtual network requests even though the topology consists of slightly fewer number of nodes and links.

Simulation Scenarios

- We used the VNE User Guide to setup and install the prerequisite and run the test case. The unit test scenario we have used is called ARRIVAL_RATE_TESTS. We use the pre-defined setting. The network files have following configuration: -
- From vnr_brite_param.xml and vnr_param.xml files
- BRITE RTWaxman algorithm to create connections between the nodes
- VNR nodes are located on 25x25 Cartesian plane
- Distance between VNR nodes are 3 and 10. Maximum number of nodes is 3.
- Each simulation time is set to 50,000-time unit
- VNR file for arrival distance parameter of 12.5 (traffic of 80 Erlang)

Simulation Scenarios

- From substrate_net_params.xml and substrate_net_generation_algo_params.xml
- Substrate nodes CPU distance, substrate link bandwidth and delay distance parameters are distributed between 50 and 100 units
- This unit testcase run all cases the algorithm configuration combinations and the result are stored in SQL DB files (in the order which they were run):

Simulation Scenarios

- From the statistics of the result, we can infer the following:
- If the state is EMBED_FAIL, the embedding process is unsuccessful, and no revenue is generated because the request is rejected
- The results show that the MaVEn algorithms performance better compared to the Vine and Global Resource Capacity (GRC) VNE algorithms.

Simulation Result

- Comparing the processing time and computation time of the VNE algorithm, the combination of “mcvne_bfs_mcf” takes the longest time. Logging was enhanced to reduce the simulation time (to be verified). This will be beneficial once we scale the test the performance on realistic DCN configuration.

Result and Analysis

Five metrics to compute the performance of VNE-Sim[5]

1. ***Acceptance ratio*** : In a given time interval τ, the ratio of the number of accepted VNRs $|\Psi^{a}(\tau)|$ to the total number of received VNRs $|\Psi(\tau)|$
2. ***Revenue to Cost ratio:*** Revenue generated embedding VNRS

$$\mathbf{R}(G^{\Psi_i}) = w_c \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + w_b \sum_{e^{\Psi_i} \in E^{\Psi_i}} \mathcal{B}(e^{\Psi_i})$$

where w_c and w_b are the weights for CPU and bandwidth requirements

Cost of resource allocation for the VNE,

$$\mathbf{C}(G^{\Psi_i}) = \sum_{n^{\Psi_i} \in N^{\Psi_i}} \mathcal{C}(n^{\Psi_i}) + \sum_{e^{\Psi_i} \in E^{\Psi_i}} \sum_{e^s \in E^s} f_{e^s}^{e^{\Psi_i}}$$

where $f_{es}^{e\Psi i}$ denotes the total bandwidth of the substrate edge e^s that is allocated for the virtual edge $e^{\Psi i}$

Result and Analysis

3. ***Profitability (Θ) :*** Calculated as a product of acceptance and revenue to cost ratio

$$\theta = p_a^{\tau} \times \frac{\sum_{\Psi^i \in \Psi^a(\tau)} \mathbf{R}(G^{\Psi^i})}{\sum_{\Psi^i i \in \Psi^a(\tau)} \mathbf{C}(G^{\Psi^i})},$$

3. ***Average Link Utilization:***

$$\mathcal{U}(N^s) = 1 - \frac{\sum_{n^s \in N^s} C(n^s)}{\sum_{n^s \in N^s} \mathcal{C}_{max}(n^s)},$$

where $C(n^s)$ is the available CPU resources of a substrate node n^s while $C_{max}(n^s)$ is the maximum CPU resources of the node

Result and Analysis

5. Average Node Utilization

$$\mathcal{U}(E^s) = 1 - \frac{\sum_{e^s \in E^s} \mathcal{B}(e^s)}{\sum_{e^s \in e^s} \mathcal{B}_{max}(e^s)},$$

where B (e^s) is the available bandwidth of a substrate link e^s while $B_{max}(e^s)$ is the maximum bandwidth of the link

Result and Analysis

Initially, the result of the Revenue to Cost computation shows that MCTS algorithm is better than GRC and Vine.

Virtual Network Request (VNR) and Substrate Network : RT Waxman Graph
Arrival time : 12.5, Processing time: 50,000 unit time
Total generated VNR request : 1000

Algorithm	Revenue to cost ratio
GRC-BFS	0.54
GRC-MCF	0.54
MCVNE-BFS	0.68
MCVNE-MCF	0.67
D-Vine MCF	0.49

Result and Analysis

We can infer from the table that the MCTS and GRC algorithms have better acceptance ratio than Vineyard

Algorithm	Acceptance ratio
GRC-BFS	0.589
GRC-MCF	0.589
MCVNE-BFS	0.593
MCVNE-MCF	0.593
D-Vine MCF	0.581
R-Vine MCF	0.581

Result and Analysis

For our test on the Dcell topology, we used the following parameters:

DCell Substrate Network Generation : Using FNSS
- Number of server on level 0, t = 2
- Number of levels, k = 2

Virtual Network Request (VNR) Network Generation: Using Brite
- RTWaxman Graph
- Arrival time: 12, 14, 16, 20, 25, **33, 50**, 97
- Processing time: 50,000 unit time
- Request file dataset: 4166, 3571, 3125, 2500, 2000, **1515, 1000**, 514

Result and Analysis

For the result using DCell, Revenue to Cost computation shows that MCTS algorithm fairly better than GRC and Vine as well

Algorithm	Revenue to cost ratio (Traffic load - 33 Erlangs)	Revenue to cost ratio (Traffic load - 50 Erlangs)
GRC-BFS	0.243	0.20
GRC-MCF	0.241	0.23
MCVNE-BFS	0.297	0.30
MCVNE-MCF	0.303	0.30
D-Vine MCF	0.248	0.247
R-Vine MCF	0.248	0.247

This reflects that DCell is a scalable network structure, and the distributed structure DCell utilizes the resources[3][4]

Result and Analysis

For the result using DCell, Acceptance ratio shows that MCTS and Vine algorithm are best.

Algorithm	Acceptance ratio (Traffic load - 33 Erlangs)	Acceptance ratio (Traffic load - 50 Erlangs)
GRC-BFS	0.0534	0.055
GRC-MCF	0.0594	0.056
MCVNE-BFS	0.0627	0.059
MCVNE-MCF	0.0627	0.059
D-Vine MCF	0.0594	0.058
R-Vine MCF	0.0627	0.059

High acceptable ratio is a desirable quality. Currently, DCell shows a low acceptance ratio.

Challenges and Future Work

- Python embedding in C++ has shortcoming as the Python interpreter cannot be initialize multiple times
  - Code fixes
- Monte Carlo method is an np-hard problem so simulation takes a long time for MCTS algorithm (max. up to 3 days)
  - GLPK library is already used to alleviate this issue
  - Disabling Boost library logging calls
- System limitation with datapoints generation for the VNR files
  - Arrival time parameter limited to 97
- Successfully implemented DCell topology and tested the performance
- Animate the topology. Refactoring the code

References

1. O. Popoola, B. Pranggono, “On energy consumption of switch-centric data center networks,”. *Supercomputer*, 334–369 (2018). https://doi.org/10.1007/s11227-017-2132-5
2. F. P. Tso, S. Jouet, D P. Pezaros, “Network and server resource management strategies for data centre infrastructures: A survey, Computer Networks,” Volume 106, 2016, Pages 209-225, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2016.07.002.
3. C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, “Dcell: a scalable and fault tolerant network structure for data centers,” in Proceedings of the ACM SIGCOMM 2008 conference on Data communication, 2008, pp. 75-86
4. B. Kashif, S. U. Khan, and A. Y. Zomaya. "Green data center networks: challenges and opportunities," In *2013* 11th International Conference on Frontiers of Information Technology, pp. 229-234. IEEE, 2013.

References

5. S. Haeri and Lj. Trajkovic, “Virtual network embeddings in data center networks," in Proc. IEEE Int. Symp. Circuits Syst., Montreal, Canada, May 2016, pp. 874-877
6. Software for complex network, [Accessed online - 2020-04-06] https://networkx.github.io/
7. S.K. Kappagantula, “Virtualization of Data Centers : Case Study on Server Virtualization,” (M.S. Thesis) Master of Science in Telecommunication Systems , Faculty of Computing, Blekinge Institute of Technology , February 2018

Thank you!
Questions