VNE-Sim: A Virtual Network Embedding Simulator

Soroush Haeri and Ljiljana Trajković

Communication Networks Laboratory
http://www.ensc.sfu.ca/~ljilja/cnl/
Simon Fraser University
Vancouver, British Columbia, Canada
Roadmap

- Network virtualization
- Virtual network embedding
- VNE formulation
- VNE simulators
- VNE-Sim: architecture and components
- VNE: a discrete event system
- VNE algorithms
- Simulation scenarios
- Conclusions, future work, and references
Network virtualization

- Enables coexistence of multiple virtual networks on a physical infrastructure
- Virtualized network model *divides* the role of Internet Service Providers (ISPs) into:
  - Infrastructure Providers (InPs)
    - manage the *physical infrastructure*
  - Service Providers (SPs)
    - *aggregate resources* from multiple InP into multiple Virtual Networks (VNs)
Substrate network vs. virtual network

- InPs operate physical substrate networks (SNs)
- SN components:
  - physical nodes (substrate nodes)
  - physical links (substrate links)
- Substrate nodes and links are:
  - interconnected using arbitrary topology
  - used to host various virtualized networks with arbitrary topologies
- Virtual networks are embedded into a substrate network
Roadmap

• Network virtualization
• Virtual network embedding
• VNE formulation
• VNE simulators
• VNE-Sim: architecture and components
• VNE: a discrete event system
• VNE algorithms
• Simulation scenarios
• Conclusions, future work, and references
Virtual network embedding

- **Virtual Network Embedding (VNE)** allocates SN resources to VNs
- InP’s *revenue* depends on VNE *efficiency*
- VNE problem may be reduced to the multi-way separator:
  - NP-hard
  - *optimal* solution may only be obtained for *small* instances

VNE solution

- Two subproblems:
  - **Virtual Node Mapping (VNoM)**: maps virtual nodes to substrate nodes
  - **Virtual Link Mapping (VLiM)**: maps virtual links to substrate paths
- VNE algorithms address the VNoM while solving the VLiM using:
  - Shortest-Path (SP) algorithms
  - Multicommodity Flow (MCF) algorithm
VNE solution: VLiM and path splitting

- The shortest-path algorithms do not permit path splitting:
  - stricter than the MCF algorithm
- MCF algorithm enables path splitting:
  - a flow may be divided into multiple flows with lower capacity
  - flows are routed through various paths

Roadmap

• Network virtualization
• Virtual network embedding (VNE)
• VNE formulation
• VNE simulators
• VNE-Sim: architecture and components
• VNE: a discrete event system
• VNE algorithms
• Simulation scenarios
• Conclusions, future work, and references
VNE formulation: constrains

- Substrate network graph: $G^s(N^s, E^s)$
- Resources:
  - substrate nodes: CPU capacity $C(n^s)$
  - substrate links: bandwidth $B(e^s)$
VNE formulation: constrains

- Virtual network graph: $G_{\Psi_i}(N_{\Psi_i}, E_{\Psi_i})$
- Resources:
  - virtual nodes: CPU capacity $C(n_{\Psi_i})$
  - virtual links: bandwidth $B(e_{\Psi_i})$
VNE: example

R = 39 \equiv 15+5+10+3+2+4
C = 43 \equiv 15+5+10+3+2+4+4
Roadmap

- Network virtualization
- Virtual network embedding
- VNE formulation
- VNE simulators
- VNE-Sim: architecture and components
- VNE: a discrete event system
- VNE algorithms
- Simulation scenarios
- Conclusions, future work, and references
Simulation of VNE algorithms

- Performance of VNE algorithms is only evaluated using discrete event simulations
- There is a need for a scalable, flexible, reliable, and modular VNE simulator
Existing VNE simulators

• Embed and ViNEYard:
  • both are early developed C-based VNE simulators
  • no longer maintained
  • lack documentation and do not provide interfaces for developing new VNE algorithms and performance metrics

Existing VNE simulators

• **Alevin:**
  • written in Java
  • modular design
  • provides a graphical user interface
  • design does not enable users to define network elements of their choice such as defining CPU, memory, and storage
  • writing scripts to perform batch simulations is rather cumbersome

Roadmap

- Network virtualization
- Virtual network embedding
- VNE formulation
- VNE simulators
- VNE-Sim: architecture and components
- VNE: a discrete event system
- VNE algorithms
- Simulation scenarios
- Conclusions, future work, and references
VNE-Sim

• A discrete event VNE simulator written in C++
• Used for performance evaluation of a number of VNE algorithms
• Publicly available under the terms of the MIT License
• Provides extensible interfaces for users to implement algorithms and customizable network elements
• Good memory management


VNE-Sim architecture

- Written using the 2011 C++ standard (C++11)
- The CMake build system is employed for code compilation
- VNE-Sim relies on several external tools:
  - Boost File System, Log, Thread, and Unit Test Framework libraries
  - GNU Scientific Library (GSL)
  - GNU Linear Programming Kit (GLPK)
  - SQLite3 library
VNE-Sim architecture

VNE-Sim components and their dependencies
VNE-Sim components

• Core package:
  • contains classes and interfaces for implementing various virtual network embedding algorithms

• Network-generator package:
  • generates various network topologies and Virtual Network Requests:
    • Boston University Representative Internet Topology Generator (BRITE) for random graphs
    • Fast Network Simulation Setup (FNSS) for well defined topologies
  • relies on configuration.xml file to define various parameters for simulation scenarios
VNE-Sim components

- **GRC package:**
  - contains the implementation of the Global Resource Capacity (GRC) algorithm

- **ViNEYard package:**
  - contains the implementation of R-ViNE and D-ViNE


Roadmap

- Network virtualization
- Virtual network embedding
- VNE formulation
- VNE simulators
- VNE-Sim: architecture and components
- **VNE: a discrete event system**
- VNE algorithms
- Simulation scenarios
- Conclusions, future work, and references
VNE: a discrete event system

- In VNE-Sim, VNE process is modeled as a discrete event system using the discrete event system specification (DEVS) framework
  - Adevs library is employed for the implementations
  - enables seamless implementation of various VNE approaches including single and batch processing

VNE: a discrete event system

Directed acyclic graph used for modeling the VNE process in VNE-Sim
VNE: a discrete event system

- Virtual Network Requests (VNRs) are first generated by the VNR Generator and passed to the Embedding Process
- Successfully embedded VNRs are forwarded to the Release Process
- The Statistic Collection module is an observer that receives information from all input/output ports of other modules
Roadmap

- Network virtualization
- Virtual network embedding
- VNE formulation
- VNE simulators
- VNE-Sim: architecture and components
- VNE: a discrete event system
- VNE algorithms
- Simulation scenarios
- Conclusions, future work, and references
VNE objective

- Maximize the profit of InPs
- Contributing factors to the generated profit:
  - embedding revenue
  - embedding cost
  - acceptance ratio

VNE objective: revenue

- Maximize revenue:

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

- \(w_c\): weights for CPU requirements
- \(w_b\): weight for bandwidth requirements
- general assumption: \(w_c = w_b = 1\)
VNE objective: cost

- Minimize the cost:

\[ C(G_{\Psi i}) = \sum_{n_{\Psi i} \in N_{\Psi i}} C(n_{\Psi i}) + \sum_{e_{\Psi i} \in E_{\Psi i}} \sum_{e^s \in E^s} f_{e^s}^{e_{\Psi i}} \]

- \( f_{e^s}^{e_{\Psi i}} \): total allocated bandwidth of the substrate link for virtual link \( e_{\Psi i} \)
- \( C(G_{\Psi i}) \) depends on the embedding configuration
VNE objective: acceptance ratio

• Maximize acceptance ratio:

\[ p^\tau_a = \frac{|\Psi^a(\tau)|}{|\Psi(\tau)|} \]

• \(|\Psi^a(\tau)|\): number of accepted Virtual Network Requests (VNRs) in a given time interval \(\tau\)

• \(|\Psi(\tau)|\): number of all arrived VNRs in \(\tau\)
VNE objective function

- Objective of embedding a VNR is to maximize:

\[
\mathcal{F}(\Psi_i) = \begin{cases} 
R(G^{\Psi_i}) - C(G^{\Psi_i}) & \text{successful embeddings} \\
\Gamma & \text{otherwise}
\end{cases}
\]

- \(\Gamma\): large negative penalty for unsuccessful embedding

- The upper bound:

\[
\mathcal{F}(\Psi_i) \leq 0
\]
VNE algorithms:  
Global Resource Capacity (GRC)

- Node-ranking-based algorithm:
  - computes a score/rank for substrate and virtual nodes
  - employs a large-to-large and small-to-small mapping scheme to map the virtual nodes to substrate nodes
- Employs the Shortest-Path algorithm to solve VLiM
- Outperforms earlier similar proposals

VNE algorithms: Global Resource Capacity (GRC)

- Calculates the embedding capacity \( r(n^s_i) \) for a substrate node \( n^s_i \):

\[
r(n^s_i) = (1 - d)\hat{C}(n^s_i) + d \sum_{n^s_j \in \mathcal{N}(n^s_i)} \frac{\mathcal{B}(e^s(n^s_i, n^s_j))}{\sum_{n^s_k \in \mathcal{N}(n^s_j)} \mathcal{B}(e^s(n^s_j, n^k))}
\]

- \( 0 < d < 1 \): damping factor
- \( e^s(n^s_i, n^s_j) \): substrate link connecting \( n^s_i \) and \( n^s_j \)
- \( \hat{C}(n^s_i) \): normalized CPU resource of \( n^s_i \)

\[
\hat{C}(n^s_i) = \frac{\mathcal{C}(n^s_i)}{\sum_{n^s \in \mathcal{N}^s} \mathcal{C}(n^s)}
\]

August 23, 2016

SIMUTools 2016, Prague, Czech Republic
VNE algorithms:
Global Resource Capacity (GRC)

- Matrix form:

\[
\mathbf{r} = (1 - d)\hat{\mathbf{c}} + d\mathbf{M}\mathbf{r}
\]

- \(\hat{\mathbf{c}} = (\hat{C}(n_{1}^{s}), \hat{C}(n_{2}^{s}), \ldots, \hat{C}(n_{j}^{s}))^{T}\)

- \(\mathbf{r} = (r(n_{1}^{s}), r(n_{2}^{s}), \ldots, r(n_{k}^{s}))^{T}\)

- **M is a k-by-k matrix:**

\[
m_{ij} = \begin{cases} 
\frac{\mathcal{B}(e^{s}(n_{i}^{s}, n_{j}^{s}))}{\sum_{n_{k}^{s} \in \mathcal{N}(n_{j}^{s})} \mathcal{B}(e^{s}(n_{j}^{s}, n_{k}^{s}))} & e^{s}(n_{i}^{s}, n_{j}^{s}) \in E^{s} \\
0 & \text{otherwise}
\end{cases}
\]
VNE algorithms:
Global Resource Capacity (GRC)

- \( r \) is calculated iteratively:

\[
r_{k+1} = (1 - d)c + dMr_k
\]

- Initially: \( r_0 = \hat{c} \)

- Stop condition: \( |r_{k+1} - r_k| < \sigma \),
  - \( 0 < \sigma << 1 \)
VNE algorithms: R-Vine and D-Vine

- Formulate VNE problem as a Mixed Integer Program (MIP)
- Their objective is to minimize the cost of accommodating the VNRs
- Use a rounding-based approach to obtain a linear programming relaxation of the relevant MIP
- Use Multicommodity Flow algorithm for solving VLiM

Roadmap

• Network virtualization
• Virtual network embedding
• VNE formulation
• VNE simulators
• VNE-Sim: architecture and components
• VNE: a discrete event system
• VNE algorithms
• Simulation scenarios
• Conclusions, future work, and references
## Performance metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceptance ratio</td>
<td>Ratio of virtual networks that were successfully embedded into the substrate topology</td>
</tr>
<tr>
<td>Revenue to cost ratio</td>
<td>The virtualization overhead</td>
</tr>
<tr>
<td>Average revenue</td>
<td>Sum of CPU and bandwidth demands realized for the virtual networks</td>
</tr>
<tr>
<td>Average cost</td>
<td>Sum of CPU and bandwidth resources being used for the embedding</td>
</tr>
<tr>
<td>Node and link utilizations</td>
<td>CPU and bandwidth utilization of substrate nodes and links</td>
</tr>
<tr>
<td>Runtime</td>
<td>Average runtime of the algorithm</td>
</tr>
</tbody>
</table>
Simulation scenarios

- Substrate network:
  - 50 nodes and 221 edges
  - Each node of the substrate network is randomly placed on a $25 \times 25$ grid
  - The CPU capacity of substrate nodes and the bandwidth of substrate links are uniformly distributed between 50 and 100 units
Simulation scenarios

- Virtual network requests:
  - number of nodes uniformly distributed between 3 and 10
- CPU requirements:
  - uniformly distributed between 2 and 20 units
- Bandwidth requirements:
  - uniformly distributed between 0 and 50 units
Simulation results: average cost

![Graph of simulation results showing average cost against traffic load for R-ViNE, D-ViNE, and GRC.]
Simulation results: average revenue

![Graph showing average revenue vs. traffic load](image)

- **R-ViNE**
- **D-ViNE**
- **GRC**
Simulation results: acceptance ratio

![Graph showing simulation results for acceptance ratio vs traffic load. The graph compares R-ViNE, D-ViNE, and GRC across different traffic loads. The acceptance ratio decreases as the traffic load increases.]
Simulation results: revenue to cost ratio
Simulation results: link utilization

![Simulation results: link utilization graph](image-url)

- **Average link utilization (%)**
- **Traffic load (Erlang)**
- **R-ViNE**
- **D-ViNE**
- **GRC**

**Legend:**
- R-ViNE: represented by blue triangles.
- D-ViNE: represented by purple squares.
- GRC: represented by red crosses.
Simulation results: node utilization

![Graph showing node utilization against traffic load for R-ViNE, D-ViNE, and GRC.]
Roadmap

• Network virtualization
• Virtual network embedding
• VNE formulation
• VNE simulators
• VNE-Sim: architecture and components
• VNE: a discrete event system
• VNE algorithms
• Simulation scenarios

• Conclusions, future work, and references
Conclusions and future work

VNE-Sim: a virtual network embedding simulator
• formalizes the VNE process as a discrete event system based on the DEVS framework
• modular and scalable design
  • enables researchers to seamlessly implement new VNE algorithms
Future work includes implementing:
• other VNE algorithms
• a source code documentation system
• a scripting infrastructure for visualization of results
References


References

Discrete Event System Simulation:


Network Virtualization:


Virtual Network Embedding Algorithms:


