Aggregation-based Discovery for Virtual Network Environments

Heli Amarasinghe, Abdeltouab Belbekkouche, Ahmed Karmouch
School of Electrical Engineering and Computer Science, University of Ottawa, Canada
hamar040@uottawa.ca, abelbekk@uottawa.ca, karmouch@site.uottawa.ca

***Abstract*— Resource discovery aims to assist the Virtual Network Provider (VNP) in selecting the best Infrastructure Provider (InP) which has the best matching resources, for a Virtual Network (VN) request. It provides information about substrate resources that are advertised by InPs. In the current literature, resource discovery focuses mainly on static (functional) attributes of physical infrastructures with an emphasis on efficient organization of discovered data. In this paper we are proposing an aggregation based approach, called Aggregation-based Discovery for Virtual Network Environments (ADVNE). ADVNE allows VNP to make use of information on dynamic (non-functional) attributes to enhance the efficiency of resource discovery process while minimizing the continuous monitoring overhead. We analyze the performance of ADVNE and show its importance in VN embedding.**

Keywords—Network virtualization; Resource Discovery; Aggregation; Static and Dynamic Attributes;

I. INTRODUCTION

The diversity of infrastructure technologies which supported multiple protocols and distributed applications has resulted in the remarkable popularity of the internet over past two decades. However, this diversification itself has become a factual impediment to Internet development in terms both of the underlying technologies and the services offered. This is known as the ossification problem [1]. In fact, because the current architecture is owned by a large number of ISPs, it is impossible to adopt a new architecture without the agreement of many stakeholders. On the other hand, any initiative to improve Internet services will be difficult in nature and limited in scope unless consensus is reached. In this context, network virtualization has increasingly been seen as a solution to Internet ossification.

Network Virtualization focuses on allocating substrate network resources dynamically and efficiently to run multiple heterogeneous networks simultaneously on a single substrate network infrastructure. The fundamental concept of virtualization is introduction of the virtualization layer which decouples the role of the conventional network service provider to VN provider and InP. This virtualization middle layer takes care of identifying and selecting suitable infrastructure providers and provisioning virtual networks on top of their substrate networks to satisfy users who request VNs. This whole VN Embedding process can be done in three steps [2];

- *Resource discovery*: to find a set of potential Infrastructure providers and their resource attributes;
- *InP Selection:* to select the best candidate InP;
- *Binding:* to allocate resources from substrate to set-up VN

Prior to any VN provisioning operation, InPs advertise substrate resources they wish to offer to new incoming VN requests. Resource discovery refers to the phase when VNPs try to discover InPs and their available resources. After identifying a suitable set of potential candidates, best candidate will be selected in the candidate selection phase. After selecting best candidate, it will be requested to allocate resources to provision new virtual networks in the binding phase.

Resource discovery is a continuous process that is particularly crucial when many InPs are involved. In that case, resource discovery starts before resource allocation to VN requests and assists VNPs to decide on the best InP that will satisfy each request requirements most appropriately. However, there is always a tradeoff between accuracy and cost when monitoring the required network parameters for resource discovery. Hence, most of the existing research works either assume complete availability of this information or consider relatively static network parameters. Generally, candidates discovered using static parameters have higher tendency of rejection during candidate selection and binding phases, incurring higher processing overhead and delays. In this paper, we use the concept of aggregation to propose an efficient approach to monitor and discover network resources considering attributes which are dynamic in nature.

The rest of this paper is organized as follows. Section 2 briefly surveys related work in the field of VN resource discovery, emphasizing on advantages and disadvantages of existing approaches. The proposed approach, called Aggregation-based Discovery for Virtual Network Environments (ADVNE), is presented in Section 3. Performance evaluation and numerical results analysis of ADVNE is presented in Section 4. Section 5 concludes the paper and provides directions for future work.

II. RELATED WORK

Although VN discovery has a significant importance in the process of VN embedding, it has attracted very little attention in literature so far. Enhanced business models such as 4WARD [3], [4] consider the existence of a Service Provider

(SP), a Virtual Network Operator (VNO), a Virtual Network Provider (VNP) and a number of Infrastructure Providers (InPs), with the last two categories involved in resource discovery and allocation. The 4WARD model has the merit of distributing tasks related to the separation of services and infrastructure on various players at the cost of more interactions between different players. This applies to resource allocation which is requested by VNP instead of SP in other models.

Authors in [5] point out that even though there are multiple commercial substrate topology discovery applications, there's not enough work done in virtual topology discovery. They have provided an architecture to facilitate the creation and management of VN and a distributed topology discovery algorithm that performs virtual and substrate topology discovery. Topology discovery algorithm exchange messages between nodes using the spanning tree algorithm with elected root and relies on neighborhood and multicast concepts to optimize message exchanging. The platform they introduced has three modules. VN agent module runs on each and every substrate node and sends their local resource information to the managing module. Managing module gathers information provided by agents and builds virtual/substrate topologies and maintains an up-to-date database containing static and dynamic information of network resources. Control center module is a graphical user interface which is used as an interface between the user and the managing module. It allows users to create, manage and monitor virtual network resources and topologies.

A description and clustering technique for matching and discovering resources for virtual networks is proposed in [2] for the 4WARD model. Its basic function is to select an appropriate InP from a number of possibilities. A resource discovery framework has two main parts: resource description and resource clustering. Each network element is described by two types of attributes: functional and non-functional. Functional attributes include type of element, execution environment, virtualization tools, operating system and network stack. Non-functional attributes include performance parameters, capacity, location, cost/price and QoS. The InPs are responsible for advertising their functional attributes in external repositories, and non-functional attributes are updated and stored in their own local repositories. The VN provider has access to the external repositories through Resource Discovery Framework and matches requested resources with available resources.

Resource clustering involves arranging resource information in a tree structure called a dendrogram. This facilitates the matching process using a simple matching algorithm that starts at the root of the dendrogram. If the root does not satisfy the resource requirement, the request is rejected. If the root satisfies the resource requirement, the searching process goes through the branches of the dendrogram until an InP is selected. A local knowledge base is used to maintain history to reduce load and latency during resource discovery. Disregarding non-functional attributes when discovering resources is a significant drawback of this framework. This is because the time varying nature of non-functional attributes demands real-time monitoring overhead. But, without information on the non-functional aspects, satisfying requests becomes uncertain. Another drawback is that a number of InPs can have the same functional attributes. In this case, the VN provider has to deal with all the InPs that have the same set of functional attributes (since pricing and location are not advertised) leading to additional delays and computational costs.

In [6], the same authors propose to advertise the cost of allocating resources in a given InP which is a non-functional attribute. The same technique is used for resource matching in [7].

The authors in [8] use the same framework as in [2] but propose a different organization of resource discovery data anticipating reduction of searching range and cost, and improvements in the efficiency of resource discovery. Functional and non-functional attributes are referred to as static and time variant attributes. The authors emphasize that advertising time variant attributes is not reasonable because of their heavy overhead in communication, operation and maintenance. They propose storing time variant attributes (such as the residual capacity of a substrate link) in the management node of each InP. Static attributes are stored in Micro Clusters (MiCs) at the InP level and the MiCs are grouped into Macro Clusters (MaCs) at the resource discovery level based on their root attribute. Cluster Index Servers (CISs) are used at the discovery framework level to improve the searching process in MaCs

The dynamic nature of certain crucial network attributes required for VN mapping and the overhead incurred by continuous monitoring of these attributes stands as one of the challenges for discovering substrate resources to embed virtual networks. We propose an aggregation-based discovery approach to extract the values of some of these crucial network attributes and use them to enhance the resource discovery process.

III. RESOURCE DISCOVERY APPROACH

In this Section we present our resource discovery approach ADVNE. First we present an overview of ADVNE. Then we present the details of the proposed approach by presenting ADVNE architecture and its main functionalities.

A) Overview

The aim of ADVNE is to address the major drawbacks that occur when disregarding non-functional (non-static) attributes when selecting potential infrastructure providers from a pool of infrastructure providers. We do not intend to discuss how to manipulate static attributes since the previous contributions (e.g., [2], [6] and [8]) provided appropriate solutions.

B) Architecture

The general architecture of ADVNE is shown in Fig. 1. Each infrastructure provider has a ***Monitoring Agent*** which performs the task of monitoring aggregation values of certain dynamic attributes. Infrastructure providers bear the responsibility of advertising their static attributes in a common repository that can be accessed by VN provider. Therefore, sophisticated description of static attributes is assumed to be

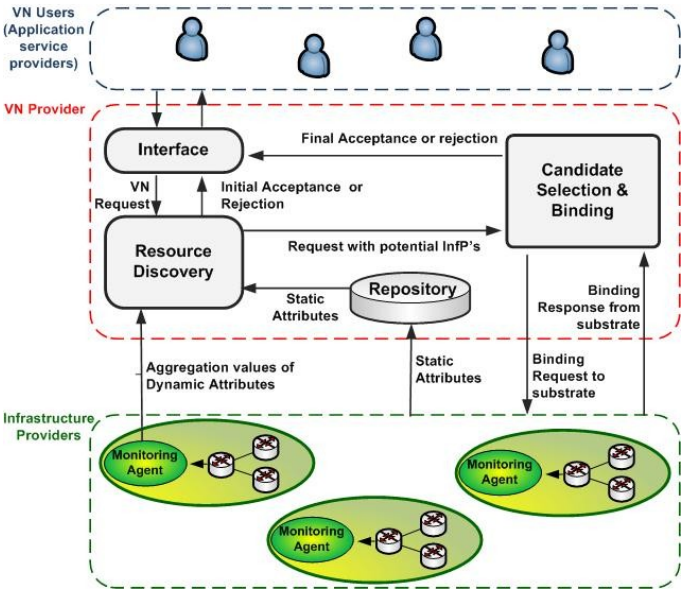

Figure 1. ADVEN Architecture

available for VNP to perform static resource matching. Moreover, as emphasized in [2], infrastructure providers can disclose information of their network resources up to a preferred degree maintaining their privacy and security requirements.

VN requests arrive to VN provider (more precisely, to its interface with VN users) with a description of node/edge requirements/constraints such as CPU capacity and link bandwidth. This request is forwarded to the ***Resource Discovery*** module in the VN provider. Resource discovery module retrieves static attributes values of resources from ***repository*** and aggregation values of dynamic attributes from monitoring agents in InPs. After an initial filtering to select InPs that are potentially able to embed the incoming VN request (this filtering will be described later in the Section), the remaining InPs will be sorted and forwarded along with the request description, to ***Candidate Selection and Binding module***. Candidate selection and binding module will forward the request description to the first infrastructure provider in the sorted order. If embedding returned with a failure from the first substrate, request will be forwarded to the second InP in the sorted list. This process iterates until the VN request is embedded or none of the selected InPs can embed the VN request. In this case, the VN request will be rejected.

C) Continuous monitoring of dynamic attributes

Continuously monitoring values of network attributes such as bandwidth, delay, CPU and memory of network links and nodes is significantly resource consuming. Since these resources are taken from the network itself, monitoring cost appears as an overhead burden to the network. On the other hand, accurate embedding decisions demand detailed information of network resources including highly dynamic attributes which require continuous monitoring. An existence of tradeoff between accuracy and cost is obvious since high accuracy comes at the cost of high overhead and similarly low estimations accuracy results in less overhead. We address this requirement of continuous monitoring of dynamic network attributes by calculating aggregates of the monitored parameters instead of monitoring each single attribute individually. This will reduce the overhead when discovering network resources of substrate networks. Aggregates assure significant reduction of the number of messages passed over the network and avoidance of monitoring packet congestion around the centralized monitoring node. This will make it possible to minimize the monitoring overhead of dynamic attributes of the network, and hence, consider dynamic attributes when performing resource discovery.

In this approach, we use the concept of aggregate discussed in [9] and [10] to find total, average, maximum and minimum values of bandwidth and CPU capacity of substrate network resources. Monitoring agents in each InP construct optimum spanning trees for their network graphs. After constructing the tree, a distributed pull-based approach is used to calculate aggregation values at each node of the tree by dividing the tree into depth levels and starting from the nodes at highest depth. Initially, aggregation values are calculated in every substrate node and the values will be forwarded to the resource discovery module. Thereafter, aggregation values are calculated for individual substrates only after a change occur in resources or their attribute values such as successful VN embedding or VN departure. Since aggregates are calculated each time after a successful embedding or departure of a VN, resources used to calculate aggregates do not impose a significant overhead on InPs. In Section IV, we present and analyze the required time to construct optimum spanning trees and calculate aggregates.

D) Initial Filtering/Pre-Matching(PM) for VN requests

Clustering approaches ([2], [6] and [8]) perform filtering of VN requests by rejecting the requests which do not comply with the description of the root of the dendrogram (as explained in related work). Requests satisfying this initial requirement will go along the branches of dendrogram and return a cluster that gives best match for the required resources. However, these clustering approaches only capable of guaranteeing that VN requests satisfy general static attributes such as VN type, virtualization environment, operating system, network stack and link type. It's obvious that although all these requirements are satisfied, possibility of rejection is still higher at candidate selection and binding due to mismatching of dynamic requirements such as available CPU capacity and link bandwidth. Authors in [2], [6] and [8] have encompassed this point by arguing that maintaining up-to-date values of real-time dynamic (non-Functional or non-Static) attributes require heavy monitoring process between discovery framework and substrate nodes. Disregarding dynamic attributes leads to higher selection and binding delays and inefficient utilization of resources since a VN request fall into a particular cluster need to be forwarded to every InP who meets the description of that cluster, requesting their responses.

We use aggregation values to reinforce the initial matching mechanism by filtering VN requests over the set of conditions that assure each VN request satisfies certain set of basic requirements. For each VN request, Maximum CPU

requirement, Total CPU requirement, Maximum BW requirement and Minimum total BW requirement will be calculated. After that, each of these requirements will be compared respectively with Maximum CPU aggregation, Total CPU aggregation, Maximum BW aggregation and Total BW aggregation, obtained from substrates. Calculated attribute values of a particular VN request will be compared with aggregation values of all substrates. If a particular substrate fails to satisfy either of these conditions, VNP will simply ignore that substrate and will consider the next substrate in the list. For a particular VN request, if all the selected substrates in the pre-matching phase fail to satisfy the conditions, that particular VN request will be rejected.

E) Sorting selected infrastructure providers

Sorting InPs according to a criterion which is proportional to their potential of embedding a new VN request will assist VNP to significantly reduce the InP selection process latency. We have chosen average path bandwidth when sorting substrates because the capacity of a substrate network to embed VNs is mainly determined by the ability of its paths to embed virtual links.

For a given substrate network $s \in S$ the set of all substrate networks, suppose that s has n nodes. The shortest path between two nodes u and v in s is noted p^s_{uv}. The bandwidth of path p^s_{uv} is defined as:

$$Bw(p^s_{uv}) = \min_{l \in p^s_{uv}} Bw(l) \quad (1)$$

And the average path bandwidth of node u is

$$AvgBw(u) = \frac{\sum_{p(uv)} Bw(p^s_{uv})}{n-1} \quad (2)$$

And Aggregation value of average path bandwidth in substrate s is

$$AggBw(s) = \sum_{u \in s} AvgBw(u) \quad (3)$$

The aggregation process is carried out for average path bandwidth over optimum spanning tree constructed for a substrate and final aggregation value is obtained from the root of the spanning tree. Substrate networks will be sorted in a queue based on their aggregation value shown in Eq. (3).

The first substrate in the queue that has maximum of average path bandwidth value will be given priority when forwarding for embedding. In case of first substrate fail to embed the request, request will be forwarded to next substrate in the queue.

IV. PERFORMANCE EVALUATION

In this section, we have evaluated the performance of ADVNE and assert the ability of pre-matching in enhancing the overall performance and efficiency of VN embedding. For the numerical analysis, we have used an open source C++ based simulator known as ViNE [11]. We have modified the original ViNE by adding new functions to support multiple InPs, construct optimum spanning tree, calculate aggregates, filter VN requests and sort substrate networks before selection and binding phases. We have used a computer with quad core 3.4GHz processor and 4 GB memory for simulations.

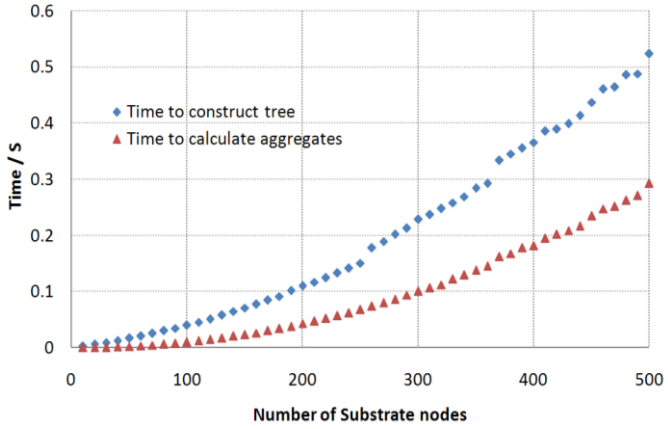

Figure 2. Performance and Behavior of Tree and Aggregation Algorithms

We have used Edmonds's optimum branching algorithm as described by Tarjan in [12] to construct an optimum spanning tree from the network graph. Substrate networks and VN requests were generated using open source, GT-ITM topology generator [13]. Substrates and virtual topologies were created using Waxman random graph model with α = 0.5 and β = 0.2. 30% of the requests have star connected topologies while remaining 70% of the requests and all the substrates have general random topologies.

Initially we have observed the scalability and behavior of the network tree construction algorithm and aggregation algorithm. Both algorithms were evaluated by measuring the time taken over number of substrate nodes and the corresponding results were plotted as shown in Fig. 2. We can see that for substrates containing 500 nodes, average time taken to construct tree is less than 0.55 seconds and average time taken to calculate aggregates is less than 0.3 seconds. Therefore it is noticeable that both of these algorithms are scalable and do not impose significant time latency.

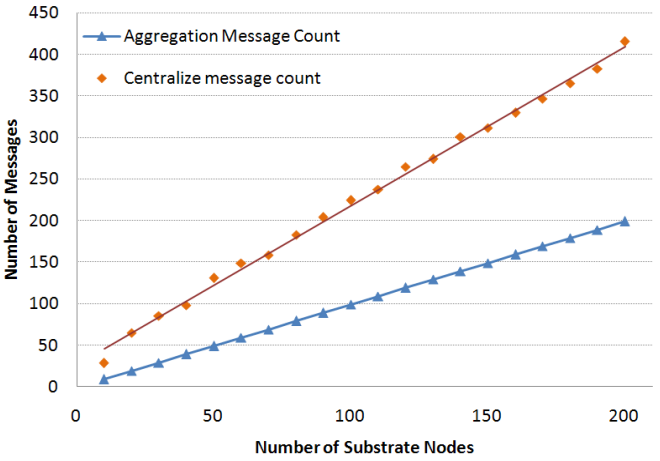

Figure 3. Number messages exchanged vs. Number of physical nodes in Aggregation and centralized approaches

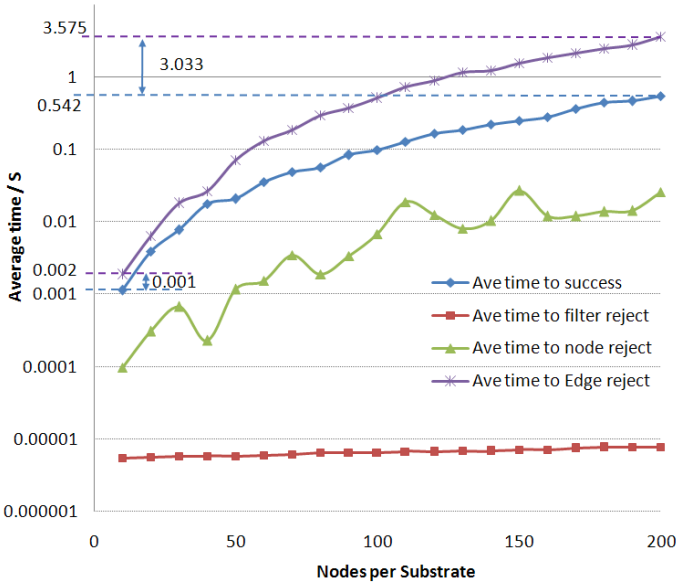

Figure 4. Average time, to take final decision of acceptance or rejection due to different factors

Numbers of monitoring messages exchanged in aggregation vs. centralized approaches are presented in Fig. 3. In centralized messaging systems, all the nodes of the network sends update messages directly to a central node over shortest paths.

Aggregation approach provided by ADVNE starts sending update messages from child nodes to their parents along the optimum spanning tree. Since child nodes and parents are always neighboring nodes, one update from any node $\boldsymbol{n}$ needs to exchange one message in the network. Hence for a substrate having $\boldsymbol{N}$ nodes, the total number of messages exchanged in the aggregation approach $\boldsymbol{U_{agg}}$ is;

$$\boldsymbol{U_{agg} = N - 1}$$

For a particular VN request, final result can be either successful embedding, rejection due failure of node mapping, rejection due to failure of edge mapping or rejection from initial filtering. We have carried out time evaluation for each of these decisions. Averages of the results were calculated for 10 trials for 5000 requests and 10 substrates containing number of substrate nodes between 10 to 200. Corresponding results are plotted in Fig. 4 with logarithmic scale vertical axis. For small substrate networks, difference between average time of rejection due to initial filtering and rejection due to node mapping failure is between 10^1- 10^2. For larger substrates, this difference is between 10^3- 10^4. Thus we can assert that initial filtering can improve decision making time of VNP significantly with large scale substrates.

If a VN request is rejected due to edge mapping failure, simulator has taken time to verify its node mapping possibility. Hence we can consider it as worst case scenario. However in Fig. 4, we can see that with the increase of substrate network size from 10 to 200, difference between average time to reject due to edge mapping failure and average time for successful mapping increases from 0.001 to 3.033 s. We can argue that this significant time difference can be observed because substrate sorting resulted in reduction of average embedding time.

V. Conclusion and Future Work

In this paper we attempted to address the challenge of improving resource discovery in network virtualalization by making information on non-functional attributes available at the VNP level. For this we proposed an aggregation-based monitoring mechanism called ADVNE. VNPs can use ADVNE approach to eliminate irrelevant InPs from the selection process and select the most appropriate InPs. We have successfully demonstrated that ADVNE approach can extract dynamic attribute values to make relevant embedding decisions while striking a tradeoff between cost and accuracy for resource discovery. The more challenging scenario, in which many VNPs are interacting with many InPs, will further complicate the problem and will have to be considered in future work. Failure detection and capability of splitting VNs over multiple InPs are important points to consider in future work.

References

[1] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in Proc. IEEE GLOBECOM'05, St. Louis, MO, USA, 2005.

[2] I. Houidi, W. Louati, D. Zeghlache, and S. Baucke, "Virtual Resource Description and Clustering for Virtual Network Discovery," in Proc. IEEE ICC'09 Workshops, Dresden, Germany, 2009.

[3] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. W. A. Greenhalgh, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," in Proc. ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures (VISA'09), Barcelona, Spain, 2009.

[4] R. Bless and C. Werle, "Network Virtualization from a Signaling Perspective," in Proc. International Workshop on the Network of the Future (Future-Net'09), Dresden, Germany, 2009.

[5] J. Nogueira, M. Melo, J. Carapinha, S. Sargento, "A Distributed Approach for Virtual Network Discovery," IEEE Globecom, Miami, Florida, USA, 2010.

[6] I. Houidi, W. Louati, W. Ben Ameur, and D. Zeghlache, "Virtual Network Provisioning Across Multiple Substrate Networks," Elsevier Computer Networks, vol. 55 no. 4, pp. 1011-1023, 2011.

[7] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive Virtual Network Provisioning," in Proc. ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA'10), New Delhi, India, 2010.

[8] B. Lv, Z. Wang, T. Huang, J. Chen, and Y. Liu, "Virtual Resource Organization and Virtual Network Embedding Across Multiple Domains," in Proc. International Conference on Multimedia Information Networking and Security (MINES'10), Nanjing, China, 2010.

[9] D. Jurca and R. Stadler, "H-GAP: Estimating Histograms of Local Variables with Accuracy Objectives for Distributed Real-Time Monitoring," IEEE Transactions on Network and Service Management, vol. 7 no. 2, 2010.

[10] A. G. Prieto and R. Stadler, "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives," IEEE Transactions on Network and Service Management, vol. 4 no. 1, 2007.

[11] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in Proc. IEEE INFOCOM'09, Rio de Janeiro, Brazil, 2009.

[12] R. E. Tarjan, "Finding optimum branchings," Wiley & Sons Inc., Networks, Volume 7, Issue 1, pages 25–35, Spring 1977.

[13] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an Inter-network," in Proceedings of IEEE INFOCOM, 1996, pp. 594-602.