Distributed Virtual Network Embedding without Sharing Topology Details

Master-Thesis von Tejamurthy Shivakumar aus Darmstadt
Tag der Einreichung:

1. Gutachten: Prof. Dr. Patrick Eugster
2. Gutachten: M.Sc. Patrick Jahnke

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Distributed Systems Programming

Distributed Virtual Network Embedding without Sharing Topology Details

Vorgelegte Master-Thesis von Tejamurthy Shivakumar aus Darmstadt

1. Gutachten: Prof. Dr. Patrick Eugster
2. Gutachten: M.Sc. Patrick Jahnke

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den June 29, 2017

(Tejamurthy Shivakumar)

Contents

1 Introduction 1
1.1 Requirements . . . 3
1.2 Outline: . . . 3

2 Related Work 4
2.1 Virtual Network Embedding . . . 4
2.2 Heuristic Approaches . . . 6
2.3 Integer Linear Programming Approach . . . 6
2.3.1 Co-ordinated Node and Link Mapping: . . . 6
2.3.2 Cross-Domain Embedding . . . 9
2.3.3 Efficient Inter-Domain Embedding . . . 11
2.3.4 Edge Wise Node Mapping (EdWiN) . . . 14
2.3.5 Recursive and Hierarchical Embedding . . . 18

3 Approach 24
3.1 Requirement I: Partial Embedding . . . 28
3.2 Requirement II: . . . 37

1 Introduction

Internet has been a huge success with its variety of applications and services contributing positively to the change, the world is witnessing today. Billions of connected devices and thousands of heterogeneous services running on the devices have challenged the performance of existing carrier networks. Current carrier networks are proprietary network infrastructures belonging to a telecommunications service provider which are not fully designed to handle the increase in traffic. In order to meet the diversified requirements of next-generation services and to be compatible with different levels of Quality of Service (QoS) demands by end users, traditional network architecture needs major changes. Researchers are focused on designing a new architecture for the future internet to address the challenges faced by existing Internet [?].

Alternative solution to the future internet is to accommodate multiple generations of mobile network in a same shared network substrate. This network sharing [1] solution although minimize the network cost, but has limitations in the field of network management such as: 1) Traffic isolation: clear separation between multiple networks' traffic. 2) Cross-domain Sharing: to share resources across multiple technology domains. 3) Monitoring: performance of each network needs to be monitored after its creation. 4) Dynamic sharing: automated dynamic resource sharing and 5) Security: Security for each network.

Designing a new architecture for future carrier networks overcomes the resistance of existing Internet to its architectural changes and is the main need of the time. Future Carrier Networks (FCN) should be able to scale billions of interconnected devices with thousands of heterogeneous services running on the network. The main challenges of FCNs [2] are to achieve service flexibility in order to adaptively provide a wide range of network services to each customer and to support applications with different levels of network QoS. Other research issues with FCNs include infrastructure sharing, optimal resource utilization and performance guarantees which needs more focus. Several ideas have been proposed to address these research challenges in order to design a new Internet architecture. Figure 1 shows proposed Future Internet Architecture from the book [?].

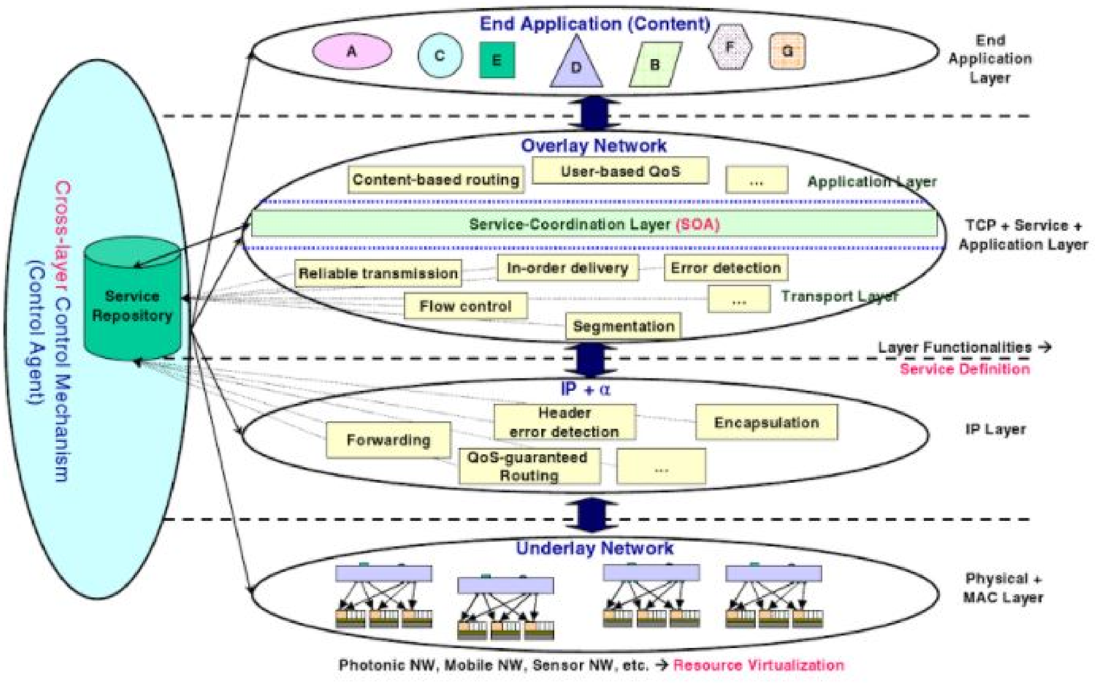


Figure 1: Architecture for Future Internet

The key parameters of the Future Internet are resource virtulization, cross layered architecture, service oriented architecture. The proposed architecture has four layers namely a) Physical layer (underlying physical networks) b) IP layer c) Overlay networks (virtual networks) and d) Application layer. The functionalities in each layer are implemented as services. Services are used by the service consumers requesting service providers.

Network Virtualization has been considered as a long-term solution to the problem faced by the existing carrier networks [?]. Network Virtualization creates virtualized environment that allows multiple heterogeneous virtual networks to coexist on a shared infrastructure [2]. Polymorphic future Internet networks can be realized that can ensure diversification and softwarization of the networks. In network virtualization the roles of the traditional Internet service providers (ISPs) are decoupled into physical infrastructure providers (PIPs) and service providers (SPs). This decoupling results in modularization of network management. PIPs and SPs are two independent entities where PIPs owns and manage the physical infrastructure and SPs create virtual networks (VNs) by aggregating resources from multiple PIPs to offer end-to-end services. VN is the primary entity in virtualization which is a combination of active and passive network nodes interconnected through virtual links. Multiple virtual topologies can be created and deployed on a physical substrate with its both nodes and links resources are virtualized. Physical ICT (Information and Communication Technology) networks become more scalable with the deployment of coexisting multiple networks. With no changes in existing network architecture each service provider can freely implement arbitrary network with customized routing and forwarding. Conceptually, existing internet can also be considered as another VN. This helps to overcome the resistance of current carrier networks to any architectural changes.

The process of mapping VNs on to a physical substrate is often referred to "Virtual Network Embedding"(VNE), which deals with the allocation of virtual resources. The main challenge in embedding VNs is to how to allocate physical resources to VN elements (nodes and links) in an optimized way. This resource allocation problem is usually referred as VNE problem. Optimized allocation of resources can be based on different objectives such as different level of QoS, minimizing the cost of embedding, minimizing resource utilization, maximizing the profit by embedding more VNs and more. Many algorithms have been presented to embed VNs on a substrate infrastructure, while optimizing service-relevant metrics. These algorithms are known as VNE algorithms. Below figure (Fig. 1) illustrates the resource allocation in future Internet with the usage of network virtualization technology.

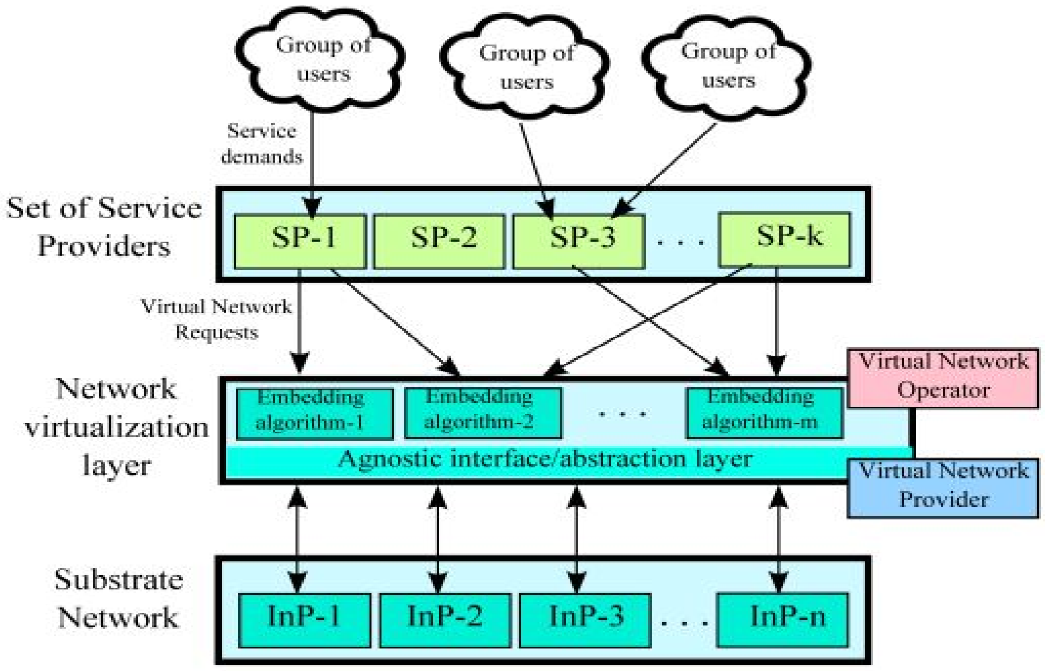


Figure 2: Resource allocation in future Internet

In the above figure SP refers to Service Providers. SPs are responsible for deploying end-to-end services. InPs refers to Infrastructure Providers who lease their physical resources to Virtual Network Providers (VNP). Virtual Network Owners (VNO) are responsible for creating, managing an operating VNs based on the requirements of SPs. Each role is independent and interacts with each other to create a VN for deploying services. VNOs uses VNE algorithms to find out suitable virtual resources that are provided by VNPs.

In order to increase the scalability there is a need for embedding VNs in various administrative, geographical and technical domains. This cross-domain embedding is another requirement to achieve softwarization of networks. VNs that span multiple physical networks results in coalitions and cooperation of providers which can be used to deploy multimedia services (VoIP, Online Gaming and live streaming) over multiple network domains. Such VNs can be used as a use case for Software Defined Networking (SDN) and management. An automated network management system is necessary to create and manage such virtual networks whose main functionality is to incorporate the diverse technologies, ownership and geographically different domains into an overarching single virtual network.

The following section briefly describes the requirements to achieve "Distributed Broker Architecture". In the thesis work, these requirements are defined and implemented to realize multi-domain orchestration and distributed brokering architecture for optimized VN Embedding.

1.1 Requirements

The main requirement of the thesis is to:

"Deploy services across multiple operators without any knowledge of underlying topologies"

By creating virtual networks that span over multiple physical substrates, we can achieve scalability and better resource utilization of the underlying substrate resources. The main problem with multi-operator virtual network is none of the operators wants to disclose their topology details. In the thesis work, we introduce VN embedding algorithm that can split/distribute VNs in multiple substrates without the operators exposing their topology details.

In order to fulfill the above requirement we derive the following sub-requirements:

1. Deploying partial services on operators own domain: Algorithm should be capable of dividing VN request so that the divided partial VN request can be optimally embedded on to operator's own domain. The VN splitting is such that the operator maximizes the profit by leasing the resources to high revenue VN with minimum resource utilization.
2. Communicate with peer operators for embedding partial services: Operators must be able to communicate with peer operators to embed partial VN, by sending/receiving messages or notifications about the VN offers. After mapping a part of VN in its own domain, operator should offer the remaining part of that VN to other operators to complete the embedding request.
3. Orchestration: An automated network management system to orchestrate successful embedding of virtual networks in a multi-domain substrate. The system should be able to retry a VN request excluding the pseudo node on which the first embedding failed.

1.2 Outline:

This section highlights the organization of the thesis report.

Chapter 2 explores state of the art techniques or related work. Existing methodologies for optimized multi-domain VN embedding and their limitations are discussed. A brief summary of related works are also provided.

Chapter 3 provides an overview of the tasks carried out during the course of this master thesis. This includes lists of tasks implemented and their scope in details.

Chapter 4 provides an overview of the proposed methodology and approaches to meet above mentioned requirements. Sub chapters include complete details on implementation phase of above mentioned requirements. This chapter also includes details of simulation set-up for carrying out experiment on proposed methodology. Following this, the results of the experiment is documented with visual graph statistics.

Chapter 5 presents future work that can be carried out as an extension to the thesis work and Chapter 6 provides a summary of the work.

2 Related Work

Virtualization is a technique of creating virtual of something, a device, resource or an operating system. Same idea is extended to communication networks in order to make it scalable, to support diversified network services while reducing operational and capital expenditures of the network. This technique makes a network to efficiently consolidate multiple virtual structures in a single network with minimum changes to its architecture. As we mentioned in the introduction section, network sharing presents major challenges like isolation between multiple networks, their security and management. Virtualized environment to create on-demand networks, provided by network virtualization (NV) technology is considered as a possible solution to address these challenges. Heterogeneous virtual networks can be built on same physical substrate by virtualizing the resources of that physical network.

Virtual Network (VN) is the fundamental element in the network virtualization [2] technology. It is the combination of virtual nodes that are connected through virtual links. Each virtual network represents network resources necessary to deploy network services such as VoIP, firewall and so on. Virtual nodes allow implementation of customized control protocols on them. In order to realize network virtualization, the virtual links connecting virtual nodes also needs to be virtualized. The users of such VNs can dynamically create, modify and remove VNs in response to the demands of the applications running on them.

The participants in the network virtualization model are classified as: 1) Service Providers (SPs) and 2) Physical Infrastructure Providers (PIPs). A network service provider requests physical infrastructure provider (PIP) for resources in the form of virtual networks. The requirements of such a VN are expressed in terms of network topology, compute power, bandwidth and so on. These virtual networks can extend across multiple PIPs. Each VN is operated and managed by single SP even though the underlying PIPs are different. As a use case for Software Defined Networking (SDN) these virtual networks can be used to deploy different network services to offer customized end-to-end services to the end users. Running network functions as VNs on top of general purpose hardware can significantly reduce cost to the network operators. When using virtualization, virtual nodes and virtual links are equally important in the economic aspect. SPs are the customers in this virtualization model who buy (lease) physical resources from PIPs. But there can also be a third participant, a broker, who plays as a mediator between SPs and PIPs. A broker also manages VN request between multiple VN operators.

Many research groups have presented design goals to realize network virtualization. In order to materialize network virtualization the following criteria should be fulfilled. 1) Flexibility: Each SP should be flexible enough to deploy arbitrary VNs, customized control protocols irrespective of underlying substrate and co-existing VNs. 2) Manageability: With the separation of SPs and PIPs, network management will be modularized. Hence virtualization should provide complete control of VNs to the SPs. 3) Heterogeneity: Both VNs and underlying substrate should heterogeneous. SPs should be able to deploy cross domain end-to-end heterogeneous VNs without adhering to any specific underlying substrate technology. 4) Isolation: Coexisting VNs must be completely isolated from each other. Any misconfiguration in one VN should not affect other VNs. This improves fault tolerance, privacy and security of the VNS. 5) Scalability: Virtualized underlying networks must be able to accommodate large number of coexisting VNs without affecting their performance. 6) Legacy Support: Virtualization should support current existing Internet by treating it as another VN in the collection of other VNs.

The efficient way of realizing virtualization is to create flexible inter-domain virtual networks that can span over multiple physical substrates where each substrate is separated from another not only geographically but even with different underlying technologies. Network architecture that provides end to end provisioning of virtualized resources is well described in the work [3].

2.1 Virtual Network Embedding

The main challenge in mapping VNs on to the physical substrate is the optimal allocation of resources to these VNs. This resource allocation problem (VNE problem) requires algorithms that can dynamically map VNs while respecting the constraints of both VNs and underlying substrates. A VNE algorithm with dynamic resource allocation that leads to self configuring system is necessary to provide customized end to end services to end users. The optimality in the allocation of resources can be calculated with regard to different objectives that can be defined by operators, service providers or different levels of QoS. Such optimal VNE algorithm can be used to decide which virtual resources can be requested from virtual network operator.

Virtual Network Embedding (VNE) algorithm deals with resource allocation of both virtual nodes and virtual links. Figure 3 shows two virtual networks VNR1 and VNR2 mapped on to same physical substrate. It can be noted from the figure (fig.3) that physical nodes can accommodate several virtual nodes and physical links can accommodate several virtual links. One virtual link can also span several physical links which demonstrates how several physical resources can be used to accommodate one virtual resource.

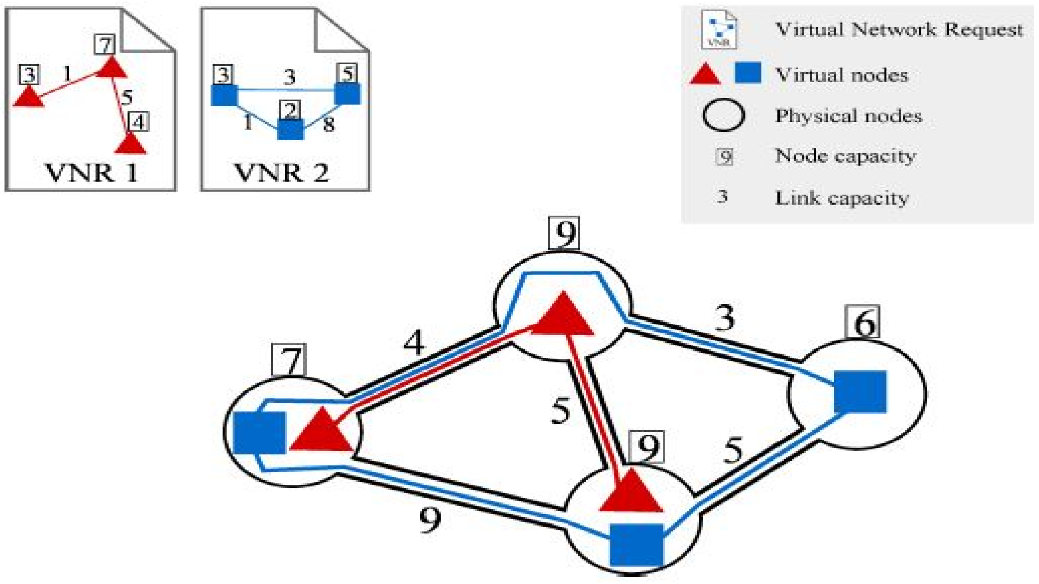


Figure 3: Virtual Networks on same substrate

It is well known that, resource allocation is the main part of VNE algorithms. Typically, every VN mapping comes with restrictions related to the resources. These restrictions should be obeyed by the underlying substrate. For example, if a virtual node request with compute power 'C' units is to be mapped, then the underlying physical node should have at least 'C' units of compute power. Similarly, a virtual link with bandwidth 'B' units cannot be mapped on to substrate link with bandwidth less than 'B' units. These are the basic restrictions that should be considered while embedding, but there can be other restrictions related to economy, profit, type of network services to be hosted etc. If there are more VNs, then the physical resource needs to be reserved for all VNs. This brings up the challenge of optimal resource allocation keeping in mind all the restrictions imposed by the virtual network operators.

The other practical challenges for VNE are admission control for VNs. Physical substrate resources are finite and some of the incoming VN requests might get rejected or put in a queue so that resource allocation for existing VN should not be affected. Another real time issue is embedding on-line requests. It needs very less computational effort to allocate resources to a predefined set of VNs as we know the resource demand before embedding. But in real time VN requests arrive dynamically and may stay in the substrate for unknown period of time. It is very difficult to predict resource demands for on-line VNs. Embedding algorithms should be capable of serving each and every VN as they arrive, instead of serving a set of known VN requests at once. Embedding dynamic VNs are difficult because algorithm has very less knowledge about those requests. The main expectation from network virtualization is to support diverse VNs. Serving VNs with arbitrary topologies and with different technologies like different operating system, different routing mechanism might be difficult.

Cross-Domain Embedding: Embedding virtual networks across multiple technologies is another challenge in achieving distributed VNE. For the Future Carrier Networks to be scalable to accommodate large number of VNs and to support requirements to host diversified services with different levels of QoS, creating and hosting VNs on single physical network is not sufficient. Network operators often fragment their networks into multiple domains for administrative reasons and to cope with its management. Such fragmented domains are different from each other in terms of vendors, underlying technologies, network resource capacity and other business prospective. Existence of orchestrating system to create and monitor VNs deployed on such heterogeneous physical domains is more important. Such a system should be able to interact with different technical domains in order to create, manage and monitor VNs that are embedded on top of these domains. The main question that arises from an orchestrating entity is- where to embed which part of VN in which technical domain so as to meet the embedding objectives. Several heuristic and exact solutions [4] have been proposed on how to configure a system that can simultaneously embed large number of VNs and solve the optimal resource allocation problem.

The main challenge with multi-domain embedding is that, underlying PIPs may want to hide their topology details from their competitors. In such a situation, it is very difficult for an orchestration system to know about complete topology information of these domains. In most of the recent solutions to the multi-domain embedding, orchestrating entity assumes the knowledge of physical domains. Earlier works on VN embedding have used two main approaches to solve VNE problem. One is through heuristic approach and other is exact solution. The following section discusses such approaches in the area of VN embedding followed by the graph models and embedding formulations used in each approach.

2.2 Heuristic Approaches

In heuristic approach [5], mapping of virtual networks on to physical network is performed by calculating low cost physical resources that can sustain general traffic of the VNs. Resource requirements for each VN is defined in terms of a set of general traffic constraints. These traffic constraints are the assumed traffic flows between designated set of nodes, defined as a matrix. With this set of traffic constraint, set of virtual nodes and links are selected to form a virtual network such that the virtual links are capable enough to allow predefined traffic to pass through them. Then suitable physical nodes and links are selected randomly which can accommodate already created virtual networks. Out of the selected set of physical nodes and links, for each VN, a best choice of physical network is selected.

In another heuristic approach [6], author tries to embed online VNs, but limits it to a time window. Incoming VN requests are collected during a time window and then embedded on the substrate network along with the constraints posed by the VNs. A queue of VN requests is maintained and VN requests are added when they are deferred because of insufficient resource in the underlying substrate. Every VN request in the queue has its own time frame for embedding, after which the request will be dropped from the queue. Requests with highest revenue will be picked first from the queue. Mapping virtual nodes of all VN request is performed first without embedding corresponding virtual links. For mapping virtual nodes, set of substrate nodes are selected which satisfy the constraints of the virtual nodes and then for each virtual node a single substrate node out of a set is selected which has maximum available resources. When the substrate nodes are selected for mapping, substrate links are selected for corresponding virtual links mapping. The k-shortest path algorithm is used to select substrate link with minimum cost between selected substrate nodes.

It is inefficient to restrict virtual link to a single substrate path. Assume if a VN request with virtual link bandwidth 100 units arrives and none of the substrate link has more than 50 units of bandwidth. Such a situation requires path splitting of virtual link to accommodate on two substrate links. Link embedding algorithm with path splitting and migration is also presented in the same work [6].

The limitation of such heuristic methods is that assumptions are made regarding traffic flows in VNs and the capacity of physical substrate as infinite. Also embedding is performed on collection of offline VNs with requirements already known. In real time, VNs are dynamic and it is not possible to predict the resource demand or their traffic flows. Most of the heuristic methods [5, 6, 7, 8] uses shortest-path algorithms for edge mapping and some greedy techniques for node mapping. Without relating edge mappings to node mappings, algorithm restricts the solution space, and may result in poor performance. Also the approaches explained here presents method for mappings of VNs only in one single physical domain. Cross-domain embedding is not possible with such assumptions and smaller substrate size.

2.3 Integer Linear Programming Approach

write about ILP, MCF, MIP

2.3.1 Co-ordinated Node and Link Mapping:

All the previous heuristic approaches presents VN embedding methods with clear separation of node mapping and link mapping. Embedding process is carried out in two phases by embedding virtual nodes in first phase and then embedding virtual links in another phase. Embedding VNs without any relation between node mapping and edge mapping will restrict the solution space for VNE problem and may result in poor performance of the algorithm.

A coordinated node and link mapping is presented in the work [9] by formulating VNE problem as Mixed Integer Programming (MIP). This work provides the basics for next set of VNE algorithms that are more effective in mapping VNs on to physical substrate. MIP problem contains decision variables of an objective function. These variables are constrained to be integer values and are used to model yes or no decisions. The objective of such a problem is a linear function that can be minimized or maximized. Such kind of objective functions with constraints is useful in solving optimization problems like optimal resource allocation.

In coordinated node and link mapping [9] a better correlation between node mappings and link mappings is introduced. Two VNE algorithms 1) Deterministic Virtual Network Embedding (D-ViNE) and 2) Randomized Virtual Network Embedding (R-ViNE) are presented. These algorithms allow mapping of virtual links to physical links right after the mapping of virtual nodes is performed. Since solving MIP is known to be NP- hard, using MIP for optimal VN mapping is also NP-hard. So the integer program is made as linear programming formulation. After that above mentioned algorithms D-ViNE and R-ViNE are used in linear programming solution to approximate the values of binary variables in original MIP. Then mapping of virtual nodes and virtual links is performed. This work tries to address the limitations of heuristic approach such as a) embedding offline VNs b) ignoring node or link requirement c) assuming substrate capacity as infinite and d) assuming all VN topology similar. This method focuses mainly on embedding online VNs using MIP formulations.

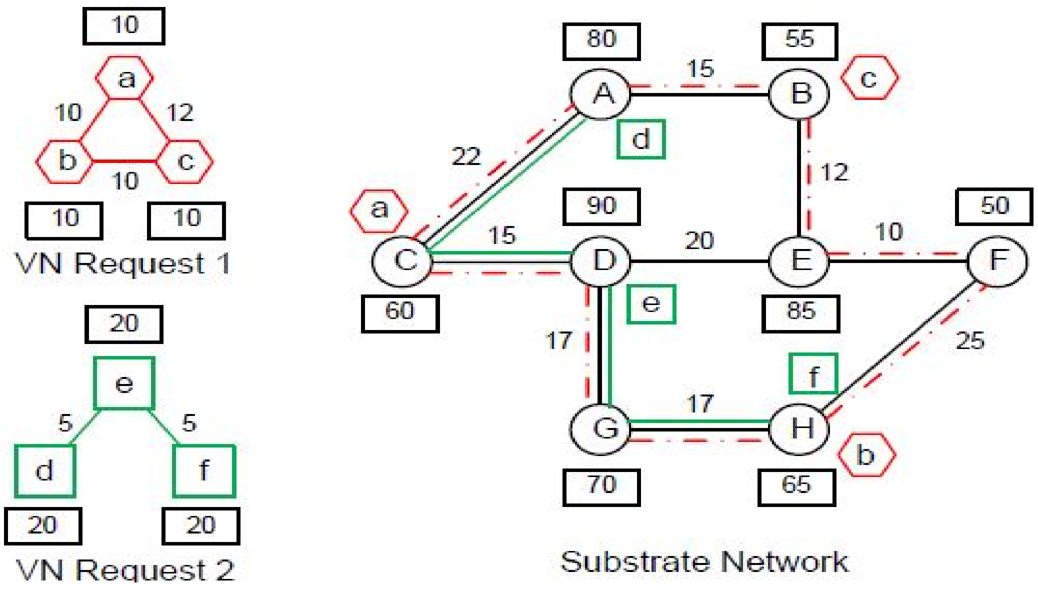


Figure 4: Substrate and VN

Network Model: Substrate network and virtual network is modeled as weighted graph G= (V, E) where V is set of vertices ad E is set of edges connecting the nodes. Substrate network is denoted by $(G^S = N^s, E^S)$, where 'S' denotes substrate. Each substrate node $n^s \epsilon N^S$ has CPU capacity cn^s along with its location details $loc(n^s)$. Physical link between two substrate nodes i and j (i,j ϵN^S) is represented by $e^s(i,j)\epsilon E^S$ with bandwidth $b(e^s)$. Similarly virtual node is denoted by $G^V = (N^V, E^V)$. Each VN request is associated with distance variable $D^V \geq 1$ which represents maximum distance from the location specified $loc(n^v)$ by a virtual node $n^v \epsilon N^V$. Available CPU resource at substrate node n^s is given by $R_N n^s$ which is the result of remaining CPU resource at that substrate node after assigning it to one of the virtual node. Similar variable is defined for physical link's bandwidth resource that is $R_E(e^s)$ where E is substrate edge and e is virtual edge. Fig. 4 shows a substrate network and a VN request, where the numbers on the links(lines) represent available bandwidth and the numbers in rectangles(nodes) represent available CPU resources.

VN Embedding process is divided into two major components node assignment and link assignment. When a VN request arrives, mapping takes place in two steps:

1. Node Assignment: Virtual nodes of a VN are mapped to those physical nodes which have sufficient CPU resource to satisfy virtual node's CPU demand. Also the distance between two substrate nodes, on which virtual nodes are mapped, must be less than or equal to D^V. Formulation for node assignment is given by:

$$M_N : N^V \rightarrow N^S \tag{1}$$

where M_N represents mapping of N^{th} node, N^V represents N^{th} virtual node and N^S represents N^{th} substrate node.

Subject to

$$c(n^v) \leq R_N(M_N(n^v))$$
$$dis(loc(n^v), loc(M_N(n^v))) \leq D^V \tag{2}$$

where $c(n^v)$ is the capacity demand from virtual node n^v, $R_N(M_N(n^v))$ is the available resource of substrate node 'N', if it is mapped with virtual node n^v. dis(i, j) denotes distance between two substrate nodes i and j.
2. Link Assignment: Each virtual link $e^v \epsilon E^V$ is mapped on to the substrate link connecting the substrate nodes on which the end nodes of this virtual link is mapped. Formulation for link mapping is given by:

$$M_E : E^V \rightarrow P^S \tag{3}$$

where M_E represents edge mapping, and P^S is set of all the substrate links.

Subject to

$$b(e^v) \leq R_E(P) \, \forall P \, \epsilon \, M_E(e^v) \tag{4}$$

As we mentioned earlier, objective functions can be maximized (minimized) depending upon the optimality factor. If suppose a netwrok operator wants to increase its revenue by mapping more number of VNs , then the VNE problem can be modeled as a optimization problem whose objective is to maximize revenue. So the goal function can be defined as maximizing the number of VN mappings by minimizing physical resource utilization. The basic VN resource constraints (eqn. 2 and eqn. 4) is still applied along with goal function. The formulations provided in this work is the basis for future embedding algorithms that we are going to discuss in next set of approaches. The set of integer constraints presented in [9] are computationally intractable. Those constraints are relaxed to obtain linear program and the solution is applied to D-ViNE and E-ViNE techniques. D-ViNE accepts online VN requests and maps them on to substrate one at a time. Mapping decision is solely based on previous VN requests. The linearized solution of MIP used in this technique will help in breaking the ties in selecting unmapped substrate node. After successful mapping of nodes, multi-commodity flow algorithm is used to map virtual links on to physical links.

R-ViNE is similar to D-ViNE but uses randomized selection od substrate nodes instead of deterministic selection. It maintains a range of probability values for each virtual node mapped to each substrate node. Based on this probability, substrate node hosts virtual nodes of a VN request. From the experiment conducted on these two algorithms, it was observed that coordinated mapping of links and nodes results in higher acceptance ratio of VNs. The objective function presented here helps in load balancing at the substrate nodes which in turn increase VN acceptance ratio. It is to be noted that better performance and increased resource utilization is achieved compared to heuristic methods.

The methods proposed above allows coordinated link and node mappings unlike heuristic methods where either node mapping or link mapping is performed using some greedy techniques. The new mathematical approach using MIP formulations is also presented for VN provisioning which increases quality of heuristic methods. This approach becomes basis for next set of MIP based VNE algorithms that try to optimally allocate resources to VNs. It is to be noted that coordination between mappings increases solution space and also increase in VN acceptance ratio.

2.3.2 Cross-Domain Embedding

Heuristic and coordinated link and node mapping approaches introduced VN provisioning techniques using max-flow/min-cut and Integer Linear Programming methods respectively. Both approaches did not discuss on embedding VNs on multiple substrate networks and also are not optimal solutions. Embedding VNs across multiple substrates enables increased VN provisioning which in turn reduces cost of the services running on them. VNs that span over multiple substrates enables cooperation between providers and instant deployment of multimedia services. The efficient way to do this is by running multimedia services on VNs with their resources spanning multiple physical networks.

Embedding VNs in single substrate is often non-realistic in today's world of increased number of internet users and its services. Embedding across multiple domains are more practical because, for one substrate it is almost impossible to manage an entire virtual network end-to-end. Inter-domain (cross-domain) embedding provides the service providers, a low cost end-to-end services which in turn decreases the cost of internet services to the end users. Different internet services have different requirements like data transmission across geographically separated networks. Such services require VNs across multiple domains. But inter-domain embedding comes with the cost of knowing topology information of all the underlying domains which is another challenge addressed in the recent research .

Mapping VNs over multiple physical substrates arises one important question. That is, how to split VN request over the substrates while reducing the cost for infrastructure providers (infps). Author in [10] introduces algorithm, which is first of its kind, for mapping VNs over multiple substrates. Splitting VNs is solved using both heuristic and Linear programming method. Also algorithm to simultaneously map virtual links and virtual nodes is presented. VNE problem is formulated by considering the participants: infrastructure provider (infp) and VN provider. Mapping VNs over multiple physical substrates is performed in three steps, 1) Resource matching 2) Embedding and 3) Binding. infp must publish its resource details to VN providers and when a VN provider receives VN request, identify suitable substrates based on the resource information exposed by every infp. VN provider must split the VN request to match against each infp at a high level of abstraction. Each single infp will embed the virtual node and virtual link on the appropriate substrate node and substrate link identified in matching phase. The following section explains the three steps (matching, embedding, binding) in details.

Both substrate and VN request is modelled as a weighted graph G = (V, E) where 'V' is set of nodes and 'E' is set of links connecting those nodes. Substrate network is represented by a graph $G^S = (N^S, L^S)$ where 'S' denotes substrate, VN request is represented by a graph $G^V = (N^V, L^V)$ where 'V' represents virtual network. A substrate G^S belonging to a particular infp is denoted by G^S_{infp}.

1. Resource Matching: VN provider performs resource matching phase based on each VN request resource descriptions and infp resource descriptions. Every node whether its a virtual node or a substrate node includes details of links attached to it. So, resource matching is only performed between nodes of VN request and physical substrate. In this phase a set of substrate nodes $Match(n^v)$ capable of hosting a virtual node $n^v \epsilon N^V$ is derived. For every VN request that arrives at VN provider, $Match(n^v)$ is calculated such that $Match(n^v) = n^s \epsilon G^S_{infp}$.

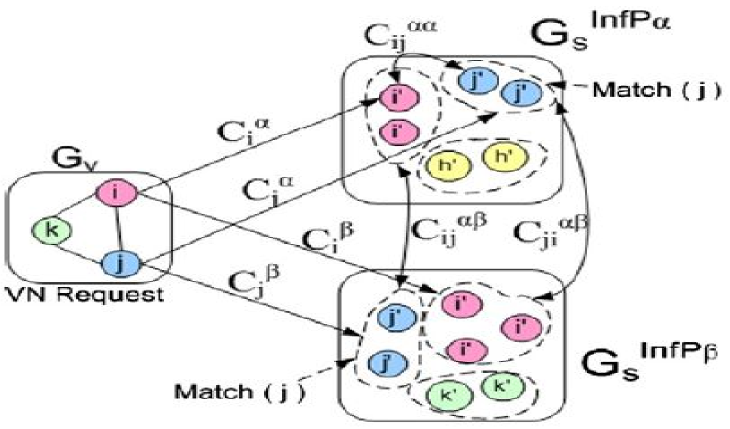


Figure 5: Splitting of VN request

2. VN Request splitting: For every virtual node, VN provider finds out several substrate nodes in multiple infps from the above resource matching step. Now that for each virtual node there are several infps, VN provider has to decide which virtual node should be embedded in which infp. After deciding the infp, VN provider will split the VN request to compose sub-requests that can be embedded into target infp. The main problem here for the VN provider is to find an optimal way of splitting VN request over the infps. For this optimal problem 'cost' of each infp is considered. The 'cost' refers to VN provisioning cost, that is the total cost of all the resources used to embed a VN. Each infp has its own cost for lending its resources for virtual networks. In order to solve this optimal problem, both heuristic ad exact solution based techniques are proposed. Exact based solution technique uses formulations in the form of quadratic program. Figure (fig. 5) shows an example of how VN splitting happens between two infps $infp^{\alpha}$ and $infp^{\beta}$ where i, j ,k are virtual nodes and $C_i^{\alpha}, C_j^{\alpha}$ and C_i^{β}, C_j^{β} are the cost for embedding the virtual nodes i, j in the infrastructure $infp^{\alpha}$ and $infp^{\beta}$ respectively.

3. VN Embedding: From the VN splitting step, VN provider finds a optimal infp for each virtual node which should host the sub VN request designed to it. In this step, for virtual node and virtual link of the sub VN request a suitable substrate node and substrate path is selected to ensure optimal embedding. Author proposes an exact solution method to simultaneously embed virtual nodes and virtual links of a VN request in an optimal way. Figure (fig. 6) shows embedding step for split VN graphs G_v' and G_v''.

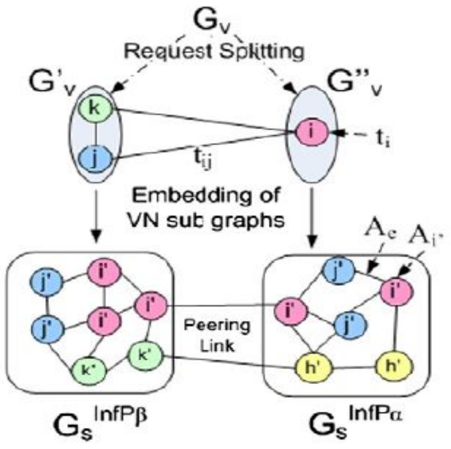


Figure 6: Embedding of sub VN requests

With the experiment on VN splitting algorithm, it was noted that exact solution based method is faster than the heuristic method to find an exact optimal solution. The objective of this work was to present an exact embedding algorithm that can simultaneously perform optimal node and link mapping in one phase. The proposed method is the first to embed online VNs that arrive in dynamic and unpredictable time. Apart from heuristic and exact methods, the work also presents MIP formulation for VN embedding. The objective function of a linear program presented here aims at minimizing the VN embedding cost. Objective function and constraints presented in this work is shown below.

2.3.3 Efficient Inter-Domain Embedding

Inter-domain embedding algorithms require more information about the substrates participating in the embedding process. The algorithms designed for intra-domain embedding does not work with inter-domain embedding. In a single domain embedding the infrastructure providers (Infp) has complete knowledge of the substrate topology and obvious knowledge about the VN request. Thus, it is very much easy that the Infp can straight away make optimal embedding calculation according to the objectives predefined. But for inter-domain embedding, Inps does not know each others topology details. So service provider has to co-ordinate with every Infp who are willing to participate and then a decision is made about mapping individual VN components to the suitable Infp. This is a major drawback when a service provider wants to deploy very large VN in less amount time. Hence a third party broker is essential to decouple service provider and Infp for a flexible and robust inter-domain VN embedding. T

In the previous section (section 2.3.2) we discussed cross domain embedding approach that uses integer linear programming method for solving optimal allocation problem. Here in this section, we discuss another cross-domain embedding approach which presents information sharing scheme between the Inps as most of the Inps are reluctant to expose their topology details. We earlier understood the new roles in the network virtualization technique. Same basic roles (service provider-SP and Infps) are extended in the work [11] by adding a broker like role called as virtual network provider (VNP) in order to make centralized embedding decisions. The information sharing scheme presented here allows VNP to have partial knowledge of underlying substrate topologies.

Here we introduce the main concepts presented in the work [11] to support inter-domain embedding along with ILP formulation for optimal embedding of VN requests. Following which we discuss the results of this inter-domain embedding approach.

(A) Business Roles:

a) Service Provider (SP): Service Providers are the customers for virtual network providers (VNP) and request virtual network topology to run on Infps. SP requests VNP with specific resource requirements which is used to create a virtual topology. SPs run their services on top of these VN to provide value based services to the end users.

b) Virtual Network Provider (VNP): VNPs are broker like agents who mediate between SPs and Infps. VNP receives VN request from SP and decompose it into the components like virtual nodes and virtual links. Then these components are embedded on the physical substrates with the help of inter-domain embedding algorithm.

c) Infrastructure Providers(Infps): These are the operators who lease their physical network resources to deploy virtual topologies. Each Infps provide resources requested from VNP according the agreement between them.

In this work VNP is assumed to be unique which can communicate with every participating Infps. Any VN request involves SP who does not care about physical substrate or the embedding technique, except requesting VNP with VN requirements and paying for it. Similarly underlying physical substrate does not bother about VNs running on them except serving the resource request made by VNP. By this way SPs and Infps are decoupled with the introduction of VNPs.

(B) Information Sharing Scheme:

The solution presented here prevents service providers from multiple negotiations with every Infp and also protects each Infps private information. The new role in the system VNP communicates and negotiates with Infps on behalf of SPs. For this, VNP decomposes a VN request and embeds on the physical substrate. In general the cost of embedding in intra-domain links is cheaper than that of external links. so a VNP, in order to maximize its profit and to minimize the expenditure tries to embed maximum parts of a VN request in the same substrate. This requires each Infp to expose partial information about their resource capacity to the VNPs.

A VN request comes with resource requirements and desired locations, that is in which physical substrate a particular virtual node has to be embedded. Along with these attributes, each virtual node has a restriction which represents the maximum distance from which the node can be embedded from its desired location. So the resource demand by each virtual node of a VN request is represented by a triple <required capacity, desired location, distance constraint>. An example of a simple VN request is pictured in the figure .7 (a) where, the numbers on the links represent required bandwidth demand of the corresponding virtual link. The figure .7 represents the information known to each role SP, VNP and Infp.

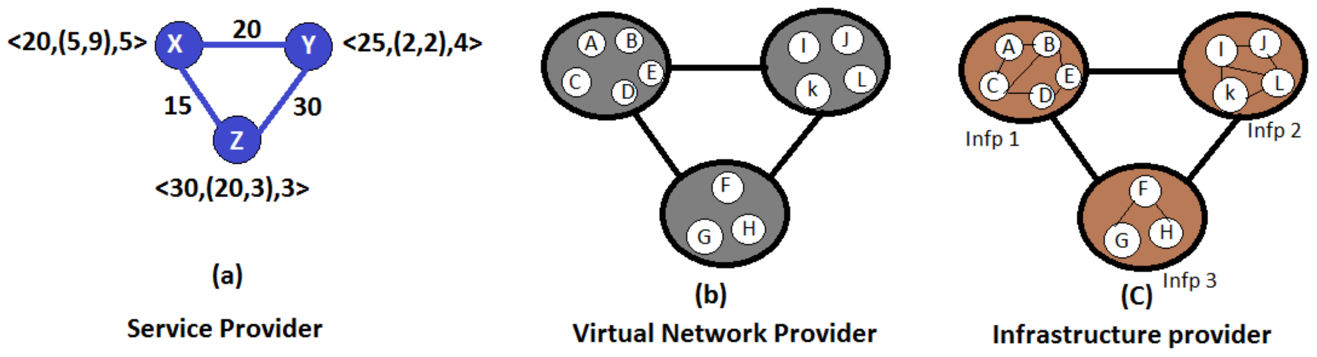


Figure 7: Level of Information known to each Role

A service provider only knows what resource requirements it needs for the VN topology. The information about underlying substrate is not known to the SP. Similarly VNP only knows which Infps are participating or ready to lease their physical resources. And Infps have knowledge of their own topologies only. Since for VNPs, physical substrate information is critical and Infps does not want to expose it, the information sharing scheme allows Inps to provide a) node location, node capacity and price b) vertices, available bandwidth and unit price of inter-domain link c) length-based price intra-domain links. From this set of information. VNP can easily allocate resource by minimizing the cost in a optimal way since it now has the knowledge of resource price and resource capacity.

(C) Inter-Domain VN Embedding:

With the help of business model presented (fig .7) and the information sharing scheme the embedding process can be carried out in the following steps:

Step 1: VN Request decomposition: After receiving VN request from the SP, VNP starts decomposing (fig .8) the VN into multiple components in a best way, such that the decomposition minimizes the overall expenditure/cost. The next step is to pre-locate the virtual nodes in the substrates. For pre-locating virtual nodes, a set of candidate substrate nodes are selected who can support the resource demands of each virtual node. In the example candidate set for virtual node 'X' is {A, B}, 'y' is {C, D} and 'Z' is {J}. Locating virtual nodes in this stage also affect path selection for virtual links. Therefore the cost of embedding nodes and links depends on each other. And for VNP, the substrate topology is invisible which makes path selection for virtual link difficult. To overcome this, author assumes a mesh topology for every participating substrates. An augmented network (see fig .8)is derived by estimating intra-domain topologies which helps VNP to coordinate node mapping and link mapping while decomposing VN request. Because of the co-existing node and link constraints, the embedding problem becomes NP-hard and hence formulation is requires for the optimal embedding. Decomposed VN request is then sent to the Infps for the actual embedding.

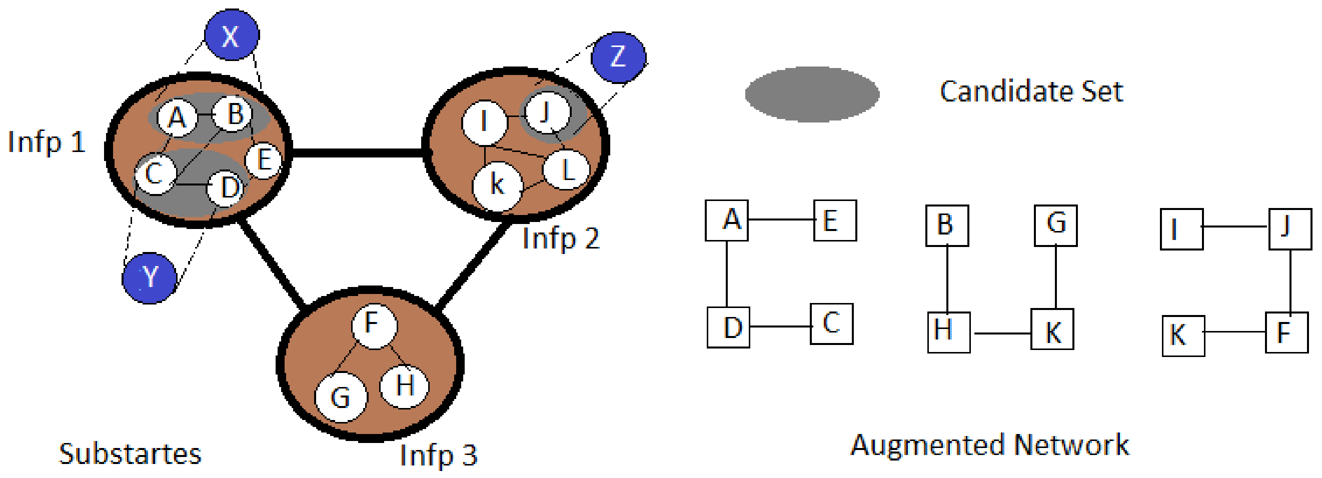


Figure 8: VN Decomposition

Step 2: Sub-VN request processing: Each participating Infps receive their own share of decomposed VN request from the VNP. These sub-VN requests are embedded by Infps in their own domains based on their own strategies. This phase has same existing intra-domain embedding procedure. From the step 1, virtual nodes of sub-VN requests are already embedded on to the substrate nodes. So the only task for Infps is to map corresponding virtual links from the sub-VN requests. Each Infp will respond to the VNP with either success message or failure message indicating the embedding result of their portion of VN request.

Step 3: Substrate Resource Allocation: If VNP receives success message from all the Infps, then embedding is considered as success for that particular VN request. For successful embedding VNP replies every Infp with confirmation message to continue with the actual resource allocation to the sub-VN requests. If even one failure message is received from the Infps then the embedding of whole VN request is considered as failure and VNP proceeds to next VN request terminating the current one.

(D) Formulation for Embedding:

Each substrate and VN are modelled as undirected graph $G^S = (N^S, L^S)$ and $G^V = (N^V, L^V)$ respectively, where N^S is set of substrate nodes, L^S is set of substrate links and N^V is set of virtual nodes, L^V is set of virtual links. Each substrate node $n^S \epsilon\, N^S$ is associated with its geographic location $g(n^S)$, its available CPU capacity $C(n^S)$ and a domain index $dom(n^S)$ representing the domain it belongs. Each substrate link $l^S(u, v) \epsilon\, L^S$ between the pair of nodes (u, v) has bandwidth capacity $B(l^S)$. Let $p^S(m^S, n^S)$ be the set of all possible path from source node m^S and destination node n^S. The available bandwidth of the path p^S is given by

$$R(P^S) = \min_{l^S \epsilon p^S} B(l^S) \tag{5}$$

Similar to the substrate node, each virtual node $n^V \epsilon\, N^V$ is associated with its desired location $g(n^V)$, its CPU demand $C(n^V)$ and non-negative radius d^V representing distance constraint. The capacity demand of each virtual link $l^V \epsilon L^V$ is denoted by $B(l^V)$. A candidate set for the virtual node $n^V \epsilon N^S$ is given by $\theta(n^V)$ which is a set of all possible substrate nodes that satisfy the distance constraint of the virtual node n^V. Hence

$$\theta(n^V) = \{n^V \epsilon N^S | dis(g(n^V), g(n^S)) \leq d^V\} \tag{6}$$

Augmented network is given by $G^A = (N^A, L^A)$ where $N^A = N^S \cup N^V$ and $L^A = L_{vs} \cup L^{ss}$. L_{vs} is a set of inter-domain links and L^{ss} is a set of intra-domain links. Augmented path set is given by $p^A(m^A, n^A)$ and available bandwidth for the augmented path set is given by

$$R(P^A) = \min_{l^A \epsilon p^A} B(l^A) \tag{7}$$

When a VN request G^V arrives at VNP, it pre-locates G^V in the augmented network G^A for which the VN is decomposed into multiple sub-VN requests. After that each Infp will be assigned with this sub-VN request. So the first phase of VN assignment starts with VNP mapping nodes and links onto the augmented network. Even though the substrate nodes are capable of hosting multiple virtual nodes, only one virtual node per VN request is mapped to one substrate node at a time. Hence node mapping $M_N : N^V \rightarrow N^A\ \forall n^V, m^V \epsilon N^V$ from virtual nodes to augmented nodes is given by:

$$M_N(n^V) \epsilon\ \theta(n^V)$$

$$M_N(m^V) = M_N(n^V), \text{ iff } m^V = n^V$$

subject to

$$C(n^V) \leq C(M_N(n^V))$$

Similarly link mapping for every virtual link is given by:

$$M_L : L^V \rightarrow P^A \text{ such that}$$

$$M_L(m^V, n^V) \subseteq P^A(M_N(m^V), M_N(n^V))$$

subject to

$$B(l^V) \leq R(P^A)$$

After decomposing VN request each sub-VN request is assigned to every domain i, denoted by $G_i^{\overline{V}} = (N_i^{\overline{V}}, L_i^{\overline{V}})$. VNP has already decided the mapping of each node $n^{\overline{V}} \epsilon\ N_i^{\overline{V}}$ on to the substrate node belonging to the domain i. So, the only task for an Infp which owns the domain i is to select the substrate node n^S for each $n^{\overline{V}}$ such that

$$M_{\overline{N}}(n^{\overline{V}}) = n^S$$

subject to

$$dom(n^S) = i \text{ and } g(n^S) = g(n^{\overline{V}})$$

The link mapping inside every Infp finds a set of substrate paths and is denoted by $M_{\overline{L}} : L_i^{\overline{V}} \rightarrow P^S$ such that

$$M_{\overline{L}}(m^{\overline{V}}, n^{\overline{V}}) \subseteq P^S(M_{\overline{N}}(m^{\overline{V}}), M_{\overline{N}}(n^{\overline{V}}))$$

subject to

$$B(l^{\overline{V}}) \leq \sum_{P^S \epsilon M_{\overline{L}}(l^{\overline{V}})} R(P^S)$$

Embedding of VN request is considered success only if VNP mapping and Infp mapping are successful. Otherwise the embedding for that VN request is considered as failure.

2.3.4 Edge Wise Node Mapping (EdWiN)

All the previous algorithms presented VN embedding methods based on heuristic and exact solution approaches for separate, coordinated and simultaneous mapping of virtual nodes and virtual links. As we know VN embedding is a resource allocation problem. Such an problem is characterized by large number of solutions and finding a best solution depends on some aim or overall objective that is implied in the problem statement. The best optimal solution is that solution which satisfies the condition of the problem as well as the objective of the problem.
Many previous works presented an optimal VN embedding solutions using Mixed Integer Programming (MIP) formulations. MIP is a mathematical optimization program in which the some or all of the variables are integers. MIP is used in linear programming in which an objective function and constraints related to it are present. The objective function is the goal of the linear programming which can either be maximized and/or minimized and constraints represents restrictions of the problem.

The methods presented in the work [10] is the first step in mapping VNs in multiple domains. It also provided a model to simultaneously map multiple VN requests using lookahead property. Another linear programming based VN embedding approach is presented in the paper [12]. The method proposed here aims at embedding VNs with larger lookahead values with predefined policies. New set of MIP formulation is presented for supporting larger lookahead values and the formulation is named as EdWiN: Edge Wise Node Mapping. Also new approach for describing both substrate structure and virtual networks is provided. The flexibility in describing the physical substrate and virtual networks in this work leads to many use cases for management and orchestration of Software Defined Networks (SDN). SDN decouples control plane and data plane which allows running network applications on top of virtual machines connected to controllers in the control plane. EdWiN method can be used in modelling virtual machines as virtual nodes and using virtual links to connect them. This approach of virtualization can be used as the basic component in the management and orchestration of future carrier networks. Figure 9 shows reference architecture of network orchestration where the network orchestrator receives VN requests from VN owners and are embedded into virtualised infrastructures.

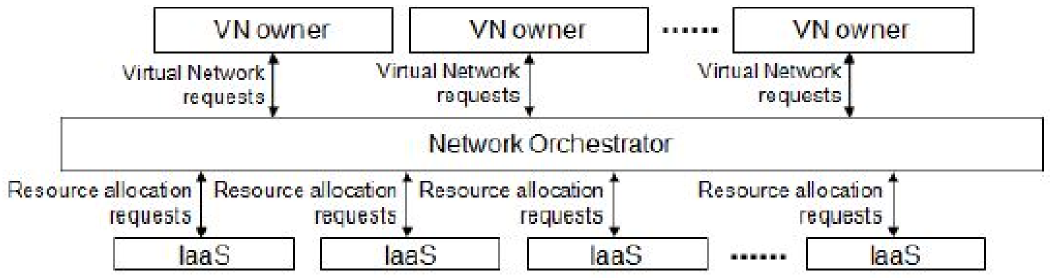


Figure 9: Network Orchestration

The formulation presented here is the basis for the formulations presented in the thesis work. Substrate model and VN model is also used in the thesis work. The network orchestrator concept presented here is also taken for the thesis work to design distributed broker architecture. The main functionality of the orchestrator is to dynamically dispatch virtualized resources to network functions. The following section provides the basic graph model and formulation used for simultaneous, optimal embedding of multiple VNs.

(A) Substrate model:

Both physical and virtual networks are modelled as undirected graph G=(V, E) where V is the set of vertices and E is the set of edges. The model for physical substrate $G^S = (V^S, E^S)$ is characterized by

- A set of N nodes $V^S = \{n_1, n_2 ... n_N\}$
- A set of $L \leq$ N(N -1) edges $E^S = \{e_{ij}\}_{i,j \epsilon \{1...N\},;\ i \neq j}$ with $e_{ij} \equiv e_{ji}$
- Node capacity matrices, for example:
  - CPU matrix $[C_i]_{i=1...N}$, where $C_i > 0\ \forall i \epsilon \{1...N\}$
  - Memory matrix $[M_i]_{i=1...N}$, where $M_i > 0\ \forall i$
  - Storage matrix $[S_i]_{i=1...N}$, where $S_i > 0\ \forall i$
- Connectivity Matrix $B = [B_{ij}]_{i,j=1...N}$, where $B_{ii} = 0$ and $B_{ij} \epsilon \{0,1\} : B_{ij} = 1$ iff e_{ij} exists; G^P is an undirected graph, thus $B_{ij} = B_{ji}$
- Link capacity matrix $B_c = [B_{c_{ij}}]_{i,j=1...N}$, where $B_{c_{ii}} = 0$ and $B_{c_{ij}} \geq 0\ \forall i,j\ \epsilon\ \{1...N\}$; e_{ij} has capacity $B_{c_{ij}}$; G^P is an undirected graph, thus $B_{c_{ij}} = B_{c_{ji}}$.

(B) VN requests:

VN requests are resolved by simultaneous embedding of G VN request, $G_g^V = (V_g^V, E_g^V)$, by splitting each VN request g ϵ {1...G} link wise into g_k 2-node sub graphs. The k-th subgraph, k $\epsilon \{1...g_k\}$, of the g-th graph is characterized by:

- Source and Destination nodes $v_{s_{gk}}$ and $v_{d_{gk}}$ with capacity requirements $C_{s_{gk}}$, $S_{s_{gk}}$ and $C_{d_{gk}}$, $S_{d_{gk}}$ respectively.
- The requested directional bandwidth b_{gk}
- Boolean variables

$$y_{s_{gk}}^i = \begin{cases} 1, & \text{if } n_{s_{gk}} \rightarrow n_i \\ 0, & \text{if } n_{s_{gk}} \nrightarrow n_i \end{cases}$$
$$y_{d_{gk}}^i = \begin{cases} 1, & \text{if } n_{d_{gk}} \rightarrow n_i \\ 0, & \text{if } n_{d_{gk}} \nrightarrow n_i \end{cases} \tag{8}$$

  where a → b implies a is embedded in b in the solution.

- Boolean variables x_{ij}^{gk}, with value 1 if k -th subgraph of the g-th request uses bandwidth on the physical link e_{ij}. Hence,

$$x_{ij}^{gk} \epsilon \{0,1\}\ \forall i,j\ \epsilon\ \{1...N\} \tag{9}$$

  where $x_{ii}^{gk} = 0,\ \forall i\ \epsilon\ \{1...N\}$
  Finally the whole VN request G_g^V, $\forall g\ \epsilon\ \{1...G\}$, is characterizes by a boolean variable:

$$y_g = \begin{cases} 1, & \text{if } G_g^V \rightarrow G^P \\ 0, & \text{if } G_g^V \nrightarrow G^P \end{cases} \tag{10}$$

(C) Cost Function:

In order to identify embedding of g-th VN request as successful, a boolean variable y_g is used. The cost function involves weight w_g to characterize the priority of y_g based on factors like price of the VN request. The second part of the cost function involves the costs that are incurred for consuming the resources to embed VN request. The variable $w_{e_{ij}}$ represents cost for using resource on the physical link e_{ij}. The variables w_{C_i}, w_{M_i} and w_{S_i} represents costs for using compute, memory and storage resources at physical substrate node i. With these variables the goal function is modelled as:

$$max\{\sum_{g=1}^{G}[w_g y_g - \sum_{k=1}^{K_g}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{e_{ij}} b_{gk} x_{ij}^{gk}$$
$$- \sum_{k=1}^{K_g}\sum_{i=1}^{N}\big(w_{C_i}\big(C_{s_{gk}} y_{s_{gk}}^i + C_{d_{gk}} y_{d_{gk}}^i\big) + w_{M_i}\big(M_{s_{gk}} y_{s_{gk}}^i + M_{d_{gk}} y_{d_{gk}}^i\big) + w_{S_i}\big(S_{s_{gk}} y_{s_{gk}}^i + S_{d_{gk}} y_{d_{gk}}^i\big)\big)]\} \tag{11}$$

The above goal function (objective) provides optimal solution for resource allocation by maximizing the profit by embedding more VNs with high revenue while minimizing resorce utilization at the substrate nodes and substrate links. The first term is to maximize VN embedding, second term is for minimizing the bandwidth utilization and third term is for minimizing the utilization of compute, memory and storage resources at the physical substrate.

(D) Constraints:

a) Topology Constraints: The link e_{ij} $\forall i, j \epsilon \{1...N\}$ can be used only if that link exists:

$$x_{ij}^{gk} \le B_{ij}, \forall g \epsilon \{1...G\}, \forall k \epsilon \{1...Kg\} \tag{12}$$

b) Capacity Constraints: For each virtual link, bandwidth allocated on each substrate link must be less than or equal to the capacity of that substrate link:

$$\sum_{g=1}^{G} \sum_{k=1}^{K_g} \left[b_{gk} \left(x_{ij}^{gk} + x_{ji}^{gk} \right) \right] \le B_{c_{ij}} \forall i, j \epsilon \{1...N\} \tag{13}$$

For virtual nodes, the compute resource allocated must be less than or equal to the compute capacity available on the substrate node:

$$\sum_{g=1}^{G} \sum_{k=1}^{K_g} \left(C_{s_{gk}} y_{s_{gk}}^{i} + C_{d_{gk}} y_{d_{gk}}^{i} \right) \le C_{ij} \ \forall i \epsilon \{1...N\} \tag{14}$$

Similar equations can be written for memory and storage also.

c) Embedding Constraints:

- The bandwidth flow constraints of each virtual node must be satisfied $\forall g \epsilon \{1...G\}, ; \ \forall k \epsilon \{1...K_g\}$

$$\begin{gathered} \sum_{j=1}^{N} x_{n_j}^{gk} - \sum_{i=1}^{N} x_{i_n}^{gk} = y_{s_{gk}}^{n} - y_{d_{gk}}^{n}; \ \forall n \epsilon \{1...N\} \\ \sum_{j=1}^{N} x_{n_j}^{gk} \le 1, \sum_{i=1}^{N} x_{i_n}^{gk} \le 1; \ \forall n \epsilon \{1...N\} \end{gathered} \tag{15}$$

- Source node and Destination node of each VN subgraph is embedded only once. Embedding of source and destination nodes are valid only if whole VN graph to which they belong is embedded.

$$\begin{gathered} \sum_{i=1}^{N} y_{s_{gk}}^{i} = y_g; \ \forall g \epsilon \{1...G\}, ; \ \forall k \epsilon \{1...K_g\} \\ \sum_{i=1}^{N} y_{d_{gk}}^{i} = y_g; \ \forall g \epsilon \{1...G\}, ; \ \forall k \epsilon \{1...K_g\} \end{gathered} \tag{16}$$

- Source node and destination node of the same VN is not mapped to same substrate. That is co-location of source and destination virtual nodes is not allowed.

$$y_{s_{gk}}^{n} + y_{d_{gk}}^{n} \le 1; \ \forall n \epsilon \{1...N\} \tag{17}$$

- Since the VN graph is divided into sub graphs based on links, virtual node (source or destination) of one VN sub graph might be part of another VN sub graph. So $\forall g \epsilon \{1...G\}, ; \ \forall k \epsilon \{1...K_g\}$

$$\begin{gathered} y_{s_{gk}}^{n} = y_{s_{k'g}}^{n}; \ \forall n \epsilon \{1...N\} \quad if \quad s_{gk} = s_{k'g} \\ y_{d_{gk}}^{n} = y_{d_{k'g}}^{n}; \ \forall n \epsilon \{1...N\} \quad if \quad d_{gk} = d_{k'g} \\ y_{d_{gk}}^{n} = y_{s_{k'g}}^{n}; \ \forall n \epsilon \{1...N\} \quad if \quad d_{gk} = s_{k'g} \end{gathered} \tag{18}$$

- Since co-location of nodes is not allowed, the separation can be forced by:

$$
\begin{aligned}
y^n_{s_{gk}} + y^n_{s_{k'g}} &\leq 1;\ \forall n\epsilon\{1...N\} \quad if \quad s_{gk} \neq s_{k'g} \\
y^n_{d_{gk}} + y^n_{d_{k'g}} &\leq 1;\ \forall n\epsilon\{1...N\} \quad if \quad d_{gk} \neq d_{k'g} \\
y^n_{d_{gk}} + y^n_{s_{k'g}} &\leq 1;\ \forall n\epsilon\{1...N\} \quad if \quad d_{gk} \neq s_{k'g}
\end{aligned}
\tag{19}
$$

(E) Location:

Forcing a virtual node to be mapped on to a particular substrate node might be considered as a constraint. Hence, constraint to bind a virtual node to a fixed substrate node can be given as:

$$
\delta^i_{*gk} \begin{cases} 1, & \text{if } n_{*gk} \rightarrow n_i \\ 0, & \text{otherwise} \end{cases}
\tag{20}
$$

where * = (s, d) refers to source or destination and $a \rightarrow b$ implies node a is embedded on substrate node b.

Additional constraints to force fixed node embedding are $\forall g\epsilon\{1...G\},;\ \forall k\epsilon\{1...K_g\}$

$$
\begin{aligned}
y^i_{s_{gk}} &= y_g \quad if \quad \delta^i_{s_{gk}} = 1 \\
y^i_{d_{gk}} &= y_g \quad if \quad \delta^i_{d_{gk}} = 1
\end{aligned}
\tag{21}
$$

where i denotes a substrate node n_i where the virtual node n_{*gk} must be embedded.

(F) Variable y_g:

The variable y_g in above equations makes it possible to have partial solutions of the embedding problem. $y_g = 1$ if VN requests can be embedded under the constraints. When none of the requests can be embedded y_g becomes 0 to all for all the requests.

(G) Path Splitting:

Formulations presented here does not allow path splitting so a virtual link can only be embedded in one single path. But changing the variable x^{gk}_{ij} from boolean to real and using Eq. 13 and Eq. 15 with the cost function (Eq. 11) path splitting can be performed.

With the formulation presented here, author compares the approach with previous approaches presented as in [10]. The exact method presented in [10] simultaneously embeds multiple request which is inferred to be inefficient as it does not allow partial solutions. That is if all the VN requests are embedded then only the solution is valid. This degrades the performance of embedding. Whereas the algorithm presented in EdWiN supports partial solutions with the help of the variable y_g.

The MIP formulation presented in EdWiN is run using arbitrary topologies and each substrate node in a topology is assigned a colour with equal probability of assignment. The node capacity is same as that of the nodes used in [10]. Multiple VN requests are created with each virtual node assigned a colour. The only condition is that each virtual node must be embedded in a substrate node matching its colour. The VNs are generated sequentially and fed to the algorithm with lookahead values of 1, 20 and 40. For instance EdWin20 gathers 20 VNs and then start embedding characterized by the variable $y_g = 1$ by reducing the capacity of the substrate. It is to be noted that EdWiN algorithm with 20 and 40 VNs results in higher embedding rate compared to EdWiN with 1 VN request. Convergence time might be sometimes important than the optimized embedding rate. It has been observed that the simplicity of the formulation contributes to the rapidness of EdWiN algorithm.

The EdWiN algorithm presented novel MIP formulation for simultaneous embedding of VN requests with increased embedding rate. A two digit gain was observed in terms of embedding rate and resource utilization along with significant reduction in the convergence time.

2.3.5 Recursive and Hierarchical Embedding

Most of the algorithms explained above perform multi-domain VN embedding by using simplified models for physical resources where underlying physical networks are modelled as flat infrastructures. In such a network scenario, each VN request is split into multiple sub-graphs. Each sub-graph is then embedded on each physical domain separately that is embedding on multi-domain and single domain separately. For instance in the work explained above in section 2.3.2, algorithm to split VN at broker level is based on min-cut and algorithm to embed them at domain level is based on MIP formulation.

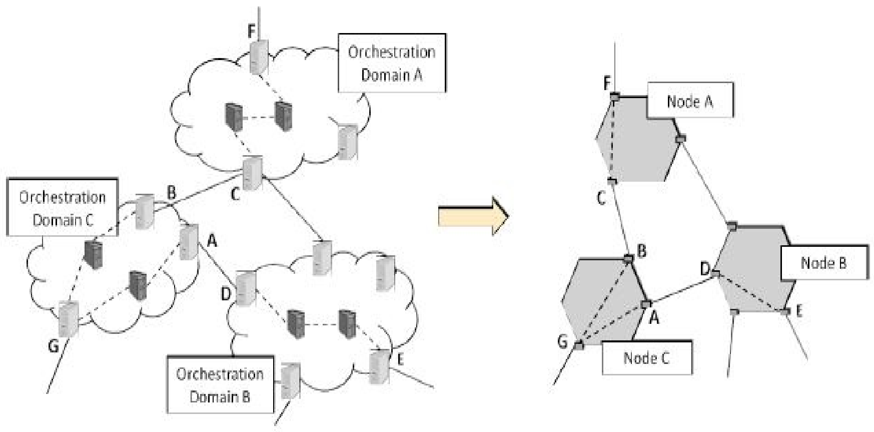


Figure 10: Network Abstraction

In the paper [?] author proposes the idea of 'abstracting physical networks' which makes multiple domains to appear as a pseudo flat infrastructure. This abstraction helps to reuse the previous embedding algorithms for flat infrastructure, used to embed VN at different domains separately, to apply for multiple domains. A slight modification to the existing algorithm is needed to use it for multiple domain embedding. The abstraction idea is presented in the project ETHICS [?] in which each physical domain abstracts its topology in order to decide the kind of information exposed to multi domain orchestrator. In this work [?] ETHICS idea of abstraction is extended to embed whole infrastructure with compute,storage and memory resources. The abstraction is done as shown in figure 10 where each physical domain is abstracted as a pseudo node so that a complex network with multiple domains look like a flat network. To this abstracted network we can now apply existing algorithms with modifications to suit multi domain embedding. It has been observed that embedding speed increases with level of network abstraction, that is higher the level of abstraction faster is the embedding speed, when compared to embedding with complete topology information. Also it is to be noted that, abstraction doesn't effect the performance of embedding algorithm.

In the figure 10 we can see each domain is abstracted into one pseudo node, for instance, domain A is abstracted into pseudo node A. The formulation presented in EdWiN (section 2.3.4) is modified to add more information about the resources of a physical domain. Algorithms including EdWiN does not allow multiple nodes inside single physical domain. Also embedding links on the physical nodes is important to support embedding sub graphs of a VN request into a domain. Hence, EdWiN formulation is modified to i) co-locate nodes and ii) to embed links on nodes- a) completely inside nodes and b) passing through the nodes, which also becomes main requirements for abstracting domains as nodes.

In order to fulfil the requirement of co-locating nodes in a single domain, author proposes the concept of interfaces inside a physical domain. Each link external to a pseudo node is connected to an interface of another pseudo node. Every interface inside a pseudo node is the real interface on the real edge node of a pseudo node to which the external link is connected as shown in figure 11. Typically the nodes/routers inside a domain is considered as interfaces. Switching bandwidth between the interfaces of a pseudo node is calculated as actual intra-domain paths between them. As shown in the figure (11), physical topology is abstracted as a pseudo graph on which existing embedding algorithms can be applied. A VN request graph is divided into sub graphs and each sub-graph is assigned to each domain. On receiving the VN sub-graph corresponding operator embeds the sub-graph at the domain level.

The level of abstracting physical network is important to decide the type embedding solutions we obtain. Lower level of abstraction leads to decreased embedding speed because large amount of information about the physical resources is to be processed. And higher level of abstraction results in less efficient solution which are far realistic. Inaccuracy in the abstraction of physical domains at multi-domain orchestrator level also leads to failed VN embedding.

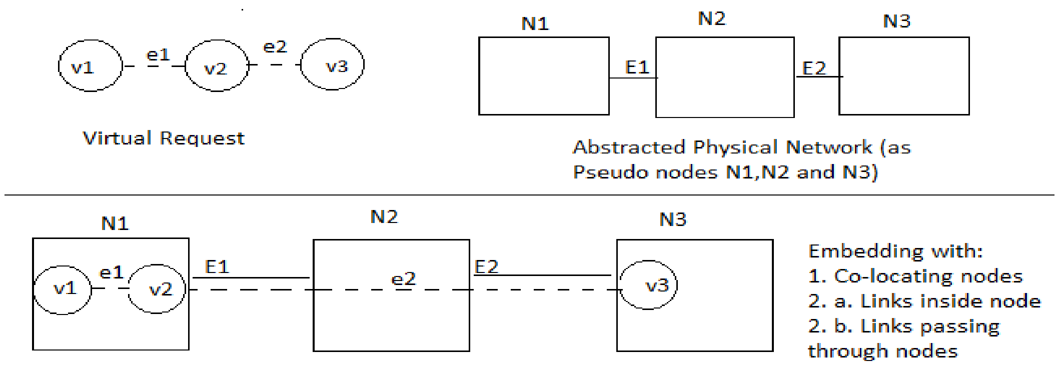


Figure 11: Fulfilling Requirements with abstraction

As explained earlier, EdWiN formulation is modified to include the interfaces inside a domain. Switching of networks enables consumption of transformed resources for forwarding nodes. This can be realized with the help of interfaces inside the domains. Interfaces can also be used to realize resources consumption while embedding links on the pseudo nodes. This is called as internal switching. Also by introducing the concept of interfaces we can keep an account of the internal bandwidth of of physical domains. Figure 12 compares old switching model and the proposed switching model that uses interfaces. The left side hexagon represents old switching model of a node n_i where as right side hexagon depicts a new switching model with interfaces (p, q in the figure).

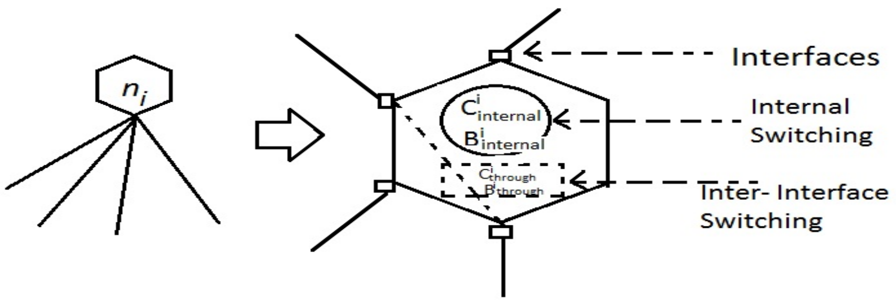


Figure 12: Switching Model

Now, here we introduce the basic graph models and the modified formulation of EdWiN to support multi-domain embedding. Basic model for the physical network graph and VN request graph is derived directly from EdWiN in the same format (substrate model and vn request model) hence equations Eq. (5) to Eq. (7) remains same.

(A) Formulation for Interfaces:

In EdWiN project, a physical vertex was a simple node having compute, storage and memory resources without the interfaces inside it. But introduction of interfaces inside a physical vertex as depicted in the figure 12 requires a new variable I_i representing set of interfaces of a physical node i. Hence,

- Every physical node n_i where $i \epsilon \{1..N\}$ (N is set of physical nodes) now has a set of interfaces I_i.
- Each interface of a node n_i is given by n_{i_p} where $p \epsilon \{1...I_i\}$.

(B) Links:

In EdWiN, links are represented as a pair (n_i, n_j) with $i, j \epsilon \{1...N\}$. With the introduction of interfaces, links are now characterized by (n_{i_p}, n_{j_q}), which says a link exists between interface 'p' of node 'i' and interface 'q' of node 'j'.

The boolean variable $x^{gk}_{i_p j_q}$ represents bandwidth allocated for the virtual link of k-th sub-graph of g-th request on the physical link $l_{i_p j_q}$. Therefore,

$$x^{gk}_{i_p j_q} \epsilon \{0,1\} \; \forall i, j \epsilon \{1..N\}; \; p \epsilon \{1..I_i\}; \; q \epsilon \{1..I_j\} \tag{22}$$

where $x^{gk}_{i_p j_p} = 0 \; \forall i \epsilon \{1..N\}; \; p \epsilon \{1..I_i\}$

(C) Nodes:

This section introduces formulation that satisfies the requirements mentioned earlier while introducing interfaces concept. That is formulation for

a) Resource Transformation and Forwarding: As we know, B_{ij} is the bandwidth capacity of a link connecting the nodes 'i' and 'j'. Now the variable $B_{i_p j_q}$ represents bandwidth between the interfaces 'p' and 'q' of a pseudo node n_i where $p,q\epsilon\{1...I_i\}$. Allocating bandwidth between any interfaces leads to resource consumption. Such resource consumption (compute, storage and memory) is expressed in term of functions $\{C^i_{through}, M^i_{through}, S^i_{through}\}$. For compute resource, formulation is given by:

$$C^i_{through} = f_{C^i_{through}}\Big(\sum_{g=1}^{G}\sum_{k=1}^{k_g}\sum_{p=1}^{I_i}\sum_{q=1}^{I_i} b_{g_k} x^{g_k}_{i_p j_q}\Big) \tag{23}$$

Similar equations can be derived for $M^i_{through}$ and $S^i_{through}$. In the paper all these functions $f_{C^i_{through}}$, $f_{M^i_{through}}$ and $f_{S^i_{through}}$ are replaced by one single function 'f' as $1/\beta$ i.e.

$$C^i_{through} = \Big(\sum_{g=1}^{G}\sum_{k=1}^{k_g}\sum_{p=1}^{I_i}\sum_{q=1}^{I_i} b_{g_k} x^{g_k}_{i_p j_q}\Big)/\beta, \beta \geq 1 \tag{24}$$

β determines the influence of allocated link with the resources of the node that link passes through.

b) Resource Transformation, Internal Bandwidth: The interfaces are used only when a physical node acts as forwarding node on the embedding path. While embedding if two connected virtual nodes are embedded on same physical node, then those virtual nodes will not only consume resources but also consume extra resource for the connection between them. This extra resource consumption is given by $r^i_{internal} = \{C^i_{internal}, M^i_{internal}, S^i_{internal}\}$ where:

$$C^i_{internal} = f_{C^i_{internal}}\Big(\sum_{g=1}^{G}\sum_{k=1}^{k_g} y^i_{s_{g_k}} y^i_{d_{g_k}} b_{g_k}\Big) \tag{25}$$

Similar equations are derived for $M^i_{internal}$ and $S^i_{internal}\}$. In order to make these functions linear, a single function is used. Hence above equation becomes:

$$C^i_{internal} = f_{C^i_{internal}}\Big(\sum_{g=1}^{G}\sum_{k=1}^{k_g} y^i_{s_{g_k}} y^i_{d_{g_k}} b_{g_k}\Big)/\alpha, \alpha \geq 1 \tag{26}$$

The equations Eq. (25) and Eq. (26) are not linear and convex. To linearise them, $y^i_{s_{g_k}}$ and $y^i_{d_{g_k}}$ are replaced by $z^i_{g_k}$. Boolean variable $z^i_{g_k}$ is such that:

$$z^i_{g_k} \geq y^i_{s_{g_k}} + y^i_{d_{g_k}} - 1,\ z^i_{g_k} \leq y^i_{s_{g_k}},\ z^i_{g_k} \leq y^i_{d_{g_k}} \tag{27}$$

c) Intra-node Bandwidth models: Here bandwidth limit for intra-node bandwidth consumption is introduced. Figure 13 depicts two bandwidth models used in the work. In 'Common Bandwidth Model' there exists single bandwidth value B_i, between all the interfaces. Since the bandwidth is shared by all the interfaces, bandwidth consumption in any of the interfaces reduces bandwidth availability for all the interfaces by the amount consumed. Hence B_i is given by:

$$B^{Available}_i = B^{total}_i - \sum_{g=1}^{G}\sum_{k=1}^{k_g}\sum_{p=1}^{I_i}\sum_{q=1}^{I_i}\{x^{g_k}_{i_p j_q} + x^{g_k}_{i_q j_p}\} \tag{28}$$

In 'Independent Bandwidth Model' every pair of interfaces inside a pseudo node 'i', have their own bandwidth which is independent of resource consumption between any other pairs of interfaces. Such bandwidth $B_{i_p i_q}$ is given by:

$$B^{Available}_{i_p i_q} = B^{total}_{i_p i_q} - \sum_{g=1}^{G}\sum_{k=1}^{k_g}\{x^{g_k}_{i_p j_q} + x^{g_k}_{i_q j_p}\} \tag{29}$$

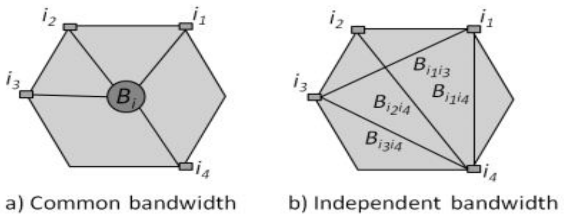


Figure 13: Switching Model

(D) Capacity Constraints:

The bandwidth allocated on each link must be less than the link's capacity. That is $\forall i, j \epsilon \{1...N\},;\ i \neq j;\ p = 1...I_i;\ q = 1...I_i$

$$\sum_{g=1}^{G}\sum_{k=1}^{k_g} b_{gk}\left(x_{i_pj_q}^{gk} + x_{j_qi_p}^{gk}\right) \leq B_{i_pj_q} \tag{30}$$

where $B_{i_pj_q} = B_{j_qi_p}$. This equation is valid for both inter-node $(i \neq j)$ and intra-node $(i = j)$ bandwidth constraint.

The compute capacity required by the virtual node must be less than the compute capacity available on the physical node. Thus,

$$\sum_{g=1}^{G}\sum_{k=1}^{K_g}\left(C_{s_{gk}}U_{s_{gk}}y_{s_{gk}}^{i} + C_{d_{gk}}U_{d_{gk}}y_{d_{gk}}^{i}\right) + C_{internal}^{i} + C_{through}^{i} \leq C_i \tag{31}$$

where

$$U_{s_{gk}} = \begin{cases} 0, & \text{if } \exists k' < k : v_{s_{gk}} \equiv v_{s_{k'g}} \text{or } v_{s_{gk}} \equiv v_{d_{k'g}} \\ 1, & \text{otherwise} \end{cases}$$

$$U_{d_{gk}} = \begin{cases} 0, & \text{if } \exists k' < k : v_{d_{gk}} \equiv v_{s_{k'g}} \text{or } v_{d_{gk}} \equiv v_{d_{k'g}} \\ 1, & \text{otherwise} \end{cases}$$

where 'v' represents virtual node and ≡ means ïs same as". And the parameters $U_{*_{gk}}$ avoids same virtual node to act as source and destination node for different 'k' sub-graphs. Equation similar to Eq .31 can be derived for other resources like memory and storage.

(E) Topology and Embedding Constraints:

For a successful VN embedding:

- For each node,inter-flow bandwidth constraint must be fulfilled. Thus $\forall n = 1...N$

$$\sum_{p=1}^{In}\left\{\sum_{j=1}^{N}\sum_{q=1}^{I_j} x_{n_pj_q}^{gk} - \sum_{i=1}^{N}\sum_{q=1}^{I_j} x_{i_qn_p}^{gk}\right\} = y_{s_{gk}}^{n} - y_{d_{gk}}^{n} \tag{32}$$

- Each sub-graphs source and destination node is embedded once and only if the whole graph of those sub-graphs is embedded. Hence $\forall i = 1...N;\ g = 1...G;\ k = 1...k_g$

$$\sum_{i=1}^{N} y_{s_{gk}}^{i} = y_g,\ \sum_{i=1}^{N} y_{d_{gk}}^{i} = y_g \tag{33}$$

- Since the VN graph is divided into sub graphs based on links, virtual node (source or destination) of one VN sub graph might be part of another VN sub graph. So $\forall n = 1...N$

$$\begin{aligned} y_{s_{gk}}^{n} &= y_{s_{gk'}}^{n}\ ;\ if \quad v_{s_{gk}} = v_{s_{gk'}} \\ y_{d_{gk}}^{n} &= y_{d_{gk'}}^{n}\ ;\ if \quad v_{d_{gk}} = v_{d_{gk'}} \\ y_{d_{gk}}^{n} &= y_{s_{gk'}}^{n}\ ;\ if \quad v_{d_{gk}} = v_{s_{gk'}} \end{aligned} \tag{34}$$

- If it is stated explicitly that co-location of nodes is not allowed, the separation can be forced by:

$$\begin{aligned} y^n_{s_{g_k}} + y^n_{s_{g_{k'}}} &\leq 1;\ if \quad v_{s_{g_k}} = v_{s_{g_{k'}}} \\ y^n_{d_{g_k}} + y^n_{d_{g_{k'}}} &\leq 1;\ if \quad v_{d_{g_k}} = v_{d_{g_{k'}}} \\ y^n_{d_{g_k}} + y^n_{s_{g_{k'}}} &\leq 1;\ if \quad v_{d_{g_k}} = v_{s_{g_{k'}}} \end{aligned} \tag{35}$$

(F) Intra-node Flow Constraints: For intra-node flow constraints, flow equation around each interface is required. For this new equation the variables $y^{i_p}_{s_{g_k}}$ and $y^{i_p}_{d_{g_k}}$ are used,whose value becomes 1 when gk-th source (destination) node uses port 'p' of the node 'i'. The equation is given by $\forall n - 1..N,\ p = 1..I_n$:

$$\sum_{j=1}^{N}\sum_{q=1}^{I_j} x^{g_k}_{n_p j_q} - \sum_{i=1}^{N}\sum_{q=1}^{I_i} x^{g_k}_{i_q n_p} + \sum_{q=1}^{I_n} x^{g_k}_{n_p n_q} - \sum_{q=1}^{I_n} x^{g_k}_{n_q n_p} = y^{n_p}_{s_{g_k}} - y^{n_p}_{d_{g_k}} \tag{36}$$

Also to ensure, in any pseudo node only one interface is used as sourceor destination node for a link:

$$\sum_{p=1}^{N} y^{n_p}_{s_{g_k}} = y^{n_p}_{s_{g_k}},\ \sum_{p=1}^{N} y^{n_p}_{d_{g_k}} = y^{n_p}_{d_{g_k}} \tag{37}$$

(G) Goal Function:

Here formulation for maximizing the goal function is defined.

$$\begin{aligned} w\sum_{g=1}^{G} y_g - \sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N}\sum_{p=1}^{I_i}\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{q=1}^{I_j} b_{g_k} x^{g_k}_{i_p j_q} \\ -(1+\gamma/\beta)\sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N}\sum_{p=1}^{I_i}\sum_{q=1}^{I_i} b_{g_k} x^{g_k}_{i_p i_q} \\ -\gamma/\alpha\sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N} b_{g_k} z^i_{g_k} \end{aligned} \tag{38}$$

where $\gamma = \frac{\text{percentage of free links in substrate}}{\text{percentage of free node resources in substrate}}$

The objective here is to maximize successful embeddings. The first term ensure that the embedding succeeds. The variable 'w' signifies that the success of the request is more or equally important compared to rest of maximization term. The second term tries to minimize the consumption of bandwidth on the link connecting two different physical domains. Similarly the third term tries to minimize amount of internal bandwidth allocated to virtual links inside a physical domain. The "1ïn the co-efficient $(1+\gamma/\beta)$ represents inter-node bandwidth used while β in γ/β stands for compute resource used. And γ is used to balance the utilization of links vs nodes while embedding links. The last term is used to minimize the compute resources used while embedding links into nodes. γ is again used to express the relative cost of nodes as compared to links.

The formulations are tested by running a simulation tool. As it is known by now, the embedding is performed in two rounds. In the first round, embedding is done over the pseudo nodes and in the second round embedding is performed inside domain represented by a pseudo node independent of first round. Embedding sub-graphs is considered only if the whole VN request graph is embedded. For simulation, the substrate topology contains two layers. First layer is the inter-domain topology which involves multiple domains interconnected and abstracted as pseudo nodes. Second layer is intra-domain layer which involves multiple nodes interconnected in every pseudo node of the first layer topology.

Formulation presented in this work with modification to EdWiN resulted in improved success rates of embedding. For a single physical substrate with 100 nodes, the proposed algorithm takes 50s to embed. But for a physical substrate of 10 pseudo nodes with 10 nodes per pseudo node, it takes one-tenth of a second. This speed up in embedding comes with loss of success rates. But 50 second wait time for embedding one VN request in a physical substrate of 100 nodes indicates the need for multi-domain orchestrator. The results clearly shows the effect of network abstraction and level of exposing information by the topologies. With less information exposed of high level of abstraction, algorithm results in inaccurate allocation of resources. When more information about the resources of a physical domain is exposed to the orchestrator, the rate of acceptance of VN request increases but consuming more time. The increase in time with increase in VN acceptance and increase in information exposure signifies a balance is required between information exposure and solution accuracy. This work makes effort in showing the trade-off between inaccurate exposure of information to multi-domain orchestrator and accurate solution of mapping VNs.

This work presented an flexible approach for VN embedding in multiple domains. It also presented a way for abstracting domain information to appear as flat infrastructure, so that existing embedding algorithms can be reused. It introduced possibility of co-locating virtual nodes. An extension to this work is Distributed Brokering Architecture in which an operator is able to request its peer operators to embed part of VN request that cannot be fulfilled by itself. So that orchestrator can retry the failed VN on different operators except where it failed. The thesis work puts effort in realizing the extension just mentioned i.e Distributed Brokering Architecture.

Add other papers from Ines Houidi, also just look at people who referred to the edwin and multi operator paper. Provide their high level details (important for your thesis too), explain broker and orchestrator for recursive and multi-domain, Explain MCF. MD domain———————End of state of art

In this section, we present approaches used for solving the problem statement mentioned in the introduction part. All the previous algorithms for multi-operator embedding requires topology and resource information of each underlying operator domains. Knowing operator's topology details or capacity details is very crucial for embedding in multiple domains. Embedding algorithm needs to know minimum details of the participating topologies or resource information for selecting suitable physical resources for every VN request. With all the necessary information about each participating topologies it is easier and less time consuming for an orchestrator to provide an accurate embedding solution.

Current approaches provide embedding solutions by assuming every operator exposes their private topology to the orchestrator entity which runs the embedding algorithm. The problem with such solutions is that in the real world, operators may not want to expose their topology details to any other operators. Many operators are reluctant to share their internal working topology, even in the abstracted form, as the topology details are paramount to to their operational success. The high level topology abstraction, if exposed by the operators, results in inefficient embedding solutions, as the algorithms will fail to calculate anything accurately. With this reluctance of exposing topology details, such solutions are very difficult to implement in the real world.

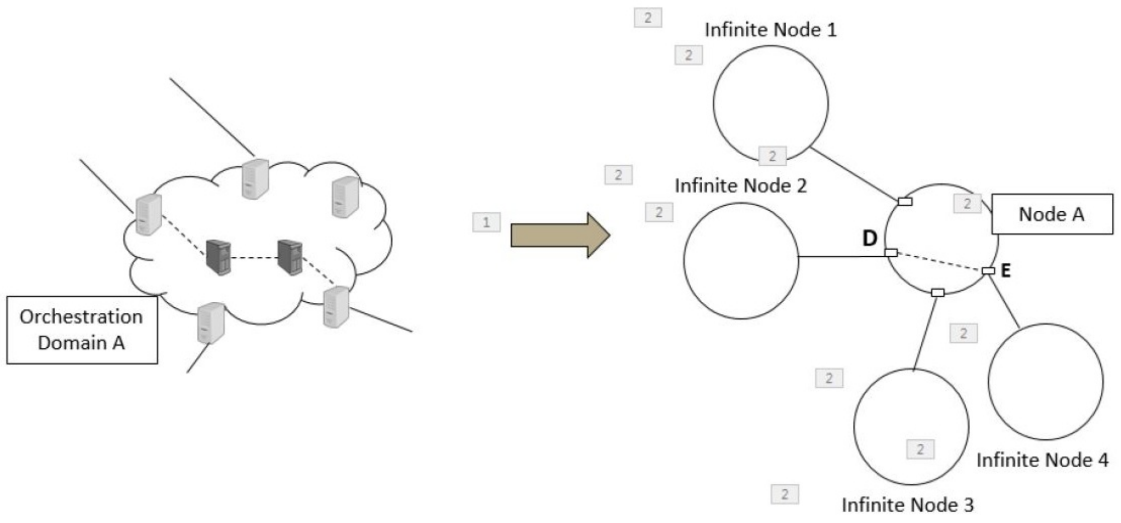


Figure 14: Partial Embedding

Here we propose a solution where operators does not need any information about the topologies of other operators. We propose an architecture where an operator instead of exposing its topology details, shares partial services/VN request that cannot be served by itself to other operators. An operator upon receiving a VN request calculates part of the request it can host and part of the request it cannot embed. Other operators can then evaluate if they are capable of hosting the partial VN request shared by other operator. By this way, an entire VN request can eventually be hosted without the need of topology details from every operator. To support this, we present ILP model where an operator can evaluate from a VN request, the part that it can host and the part that it cannot host.

Consider a scenario depicted in the figure .14 where the physical domains are abstracted as nodes. In the figure node A represents operators own node which receives virtual network requests from the customer. The main idea of this work is that the operator that receives the request (operator A) must divide the service request into a part that can be hosted by the operator himself and another that might be hosted by other operators. The part that cannot be hosted by the operator is exposed in an external database to which other operators can subscribe. Other operators may then internally evaluate if it can support this partial request. If they can then a negotiation procedure may take place to support the entire request. This way none of the operators need to expose their topologies to each other.

As we mentioned earlier, orchestrator should be able to calculate the part of VN request it can serve. The partially non-embedded part of the request may then be sent to an external database which can be accessed by other operators. The other operators can periodically check this database to see whether they can satisfy the request partially or completely. Some cases the external database may belong to third party broker. The following section provides brief picture on the system required to achieve the above mentioned goals. For this we define important components that are essential for every domains participating in the embedding process. The figure (fig .15) represents basic concept behind an orchestrator, that is responsible for managing and serving incoming VN requests.

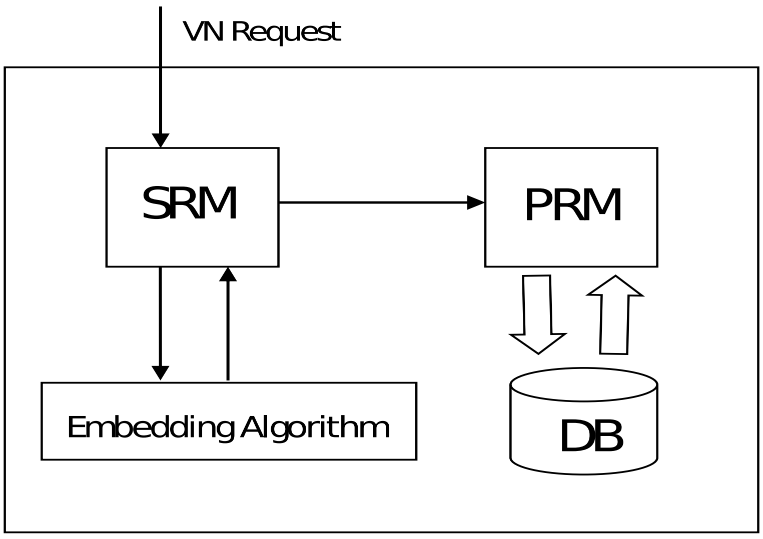


Figure 15: Orchestrator A

The orchestrator has these main components:

1. Service Request Manger (SRM): The SRM is responsible for calculating the partial request that it's own domain can fulfil with the help of embedding algorithm. SRM receives VN request and feed it to the 'embedding algorithm'. The algorithm performs mapping based on the objective function defined by the operator.

2. Pending Request Manager (PRM): This component receives either a part of VN request that failed in embedding or a whole VN request that failed to embed. In case of failed part of VN, PRM stores failed partial VN request into database. The PRM can be configured to apply some abstraction rules to the separated VN request (partial non-embedded request) received from SRM.

3. Databases: Orchestrator maintains two databases. The first database stores partial VN that succeeded in the local embedding and second stores failed partial VN for further processing. These two databases are crucial for complete embedding of a VN request as they together contain complete details of a VN request.

Now let us understand the operation of the orchestrator and its components in detail. The following flow charts (fig .16) explains how an VN request is embedded by sharing its parts between different domains. Consider two orchestrating entities Orchestrator A and Orchestrator B participating in the embedding process. The SRM component of the orchestrator A receives a VN request. The embedding algorithm is applied to this VN request to check whether local embedding is possible, i.e whether the VN can be embedded completely on the orchestrator domain A. If local embedding is possible then embedding is considered successful and orchestrator starts processing the next VN request. In case of failed local embedding, SRM with the help of the algorithm, separates the VN request into partial VN requests. The partial VN request may be part of VN request that failed to embed locally or the whole VN request that also failed to embed locally. In both the cases, partial requests are sent to the pending request manager (PRM) of that orchestrator.

The PRM component of an orchestrator saves the non-embedded part of VN request into a database which can be accessed by other operators (fig .16(a)). In order to notify other operators about the saved partial VN, orchestrator can implement a publish/subscribe system or a request crawler which periodically checks other orchestrator's PRM for new partial VN offers. Once the orchestrator B come to know about the partial VN offers, it contacts administrator of the orchestrator A with the corresponding partial VN details. If both orchestrator negotiates on embedding the partial VN request then SRM of orch. B receives the partial VN request.

After receiving the partial VN request from orch. A, now its orch. B's turn to decide whether it can support the received VN request or not (fig .16(b)). This is done by running the algorithm in orch. B. If orch. B fails to embed the VN request completely, then the embedding is considered as failure and notified to the orch. A. In this case orch. B now starts looking for new partial VNs. If in case orch. B is able to embed the received partial VN completely then the orch. A is contacted saying the VN request can be embedded completely. Now orc. B has to wait until orch. A agrees for complete embedding. If agreement is reached then embedding procedure is continued in both orchestrator for that VN request, if not orch. B stops serving this request an look for new one.

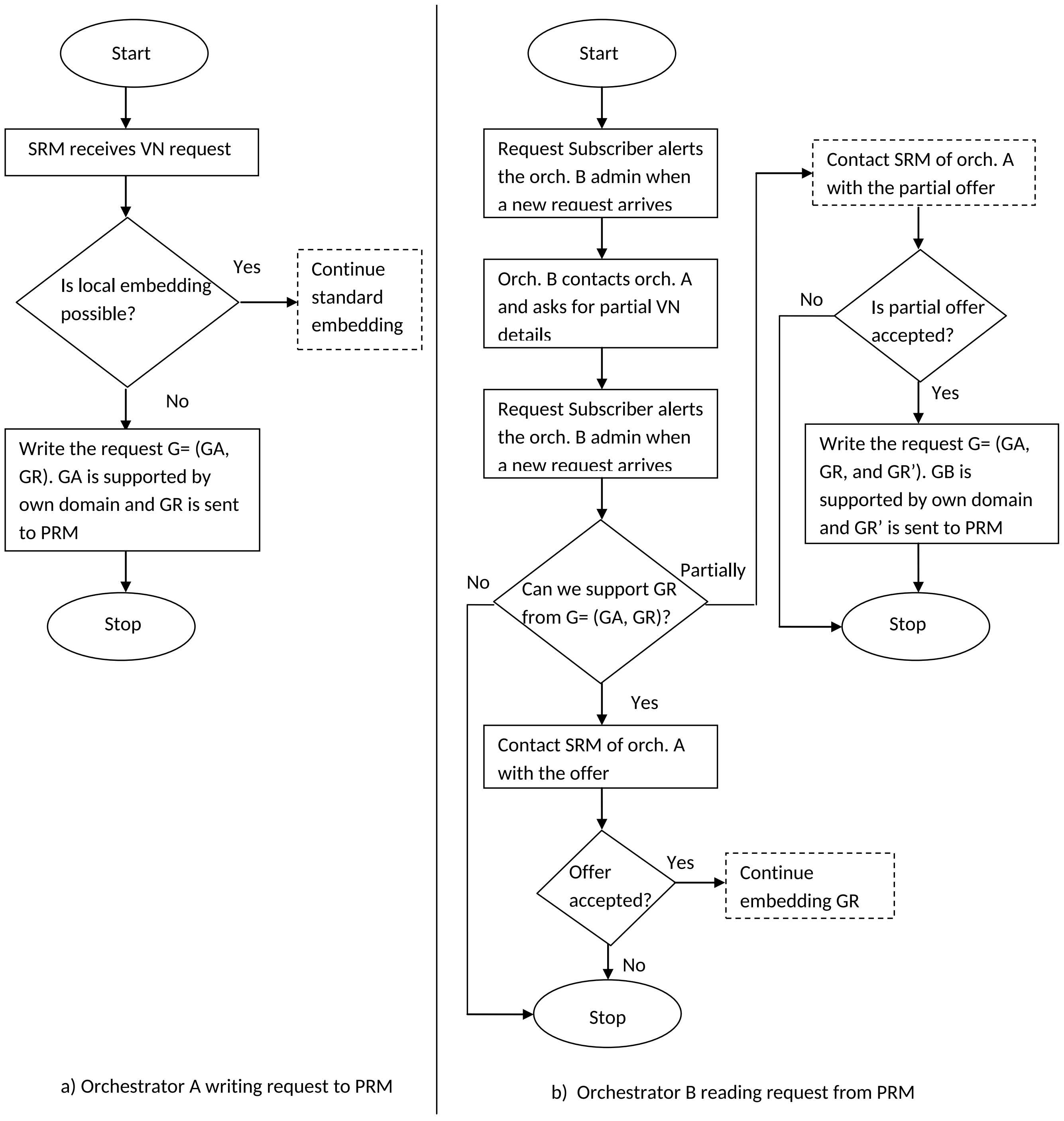

Start
SRM receives VN request
Is local embedding possible?
Yes
Continue standard embedding
No
Write the request G= (GA, GR). GA is supported by own domain and GR is sent to PRM
Stop
a) Orchestrator A writing request to PRM
Start
Request Subscriber alerts the orch. B admin when a new request arrives
Orch. B contacts orch. A and asks for partial VN details
Request Subscriber alerts the orch. B admin when a new request arrives
Can we support GR from G= (GA, GR)?
No
Partially
Yes
Contact SRM of orch. A with the offer
Offer accepted?
Yes
Continue embedding GR
No
Stop
Contact SRM of orch. A with the partial offer
Is partial offer accepted?
No
Yes
Write the request G= (GA, GR, and GR'). GB is supported by own domain and GR' is sent to PRM
Stop
b) Orchestrator B reading request from PRM

Figure 16: PRM Operations

In the thesis work, we have used publish/subscribe model for the communication between participating operators. The figure (fig .17) depicts an implementation example for the proposed solution. In general PRM is a concept under each orchestrator domain. But in this implementation, PRM is owned by third party broker and it is accessible by every orchestrator. An operator can access other operator's PRM via HTTP-based RESTful interface. PRM implementation follows publish-subscribe model where orchestrators can subscribe for certain categories of request. In our work, for simplicity we have assigned colours for partial VNs, hence type of the colour is considered as a category. For every partial embedding, the operator creates a partial VN request for the part of the request that failed in embedding. This partial VN request is assigned different colours. Once the partial VN request is generated, the corresponding orchestrator will publish a message in the system. All other orchestrators which have subscribed to a particular colour will receive a message in the form of notification, if a partial VN with the corresponding colour is arrived.

Consider a scenario as in fig .17 where three Orchestrators A,B and C are participating in the embedding process. A virtual network request arrives at orch. A with five virtual nodes and four virtual links. Orch. A runs the embedding algorithm to embed this VN request locally in its domain. The embedding solution allows only two virtual nodes to be embedded in the orch. A domain by splitting VN request into two parts. One part of VN that can be embedded locally (red coloured nodes and links) and other (Green coloured nodes and links) that cannot. In this case orch. A publishes the partial VN request (Green coloured nodes and links) to its PRM with the category "#GREEN". As we said earlier, PRM can be accessed by any operator, orch. B and orch. C subscribes to the PRM of orch. A for the category "#GREEN". Since the subscription matches with the published category of VN request, both orchestrators B and C receives notification from orch. A. The notification includes basic information about the partial VN offer and contact details of the publisher. After receiving the notification both orchestrator B and C contact administrator of orch. A requesting the partial VN details. In response to this, orch. A delivers partial VN with the category "#GREEN" to the orchestators B and C. Also, orchestrators can aggregate all the partial VN offers and publish them at once. The receiving orchestrators can select some or all of the partial VNs based on some optimization criteria.

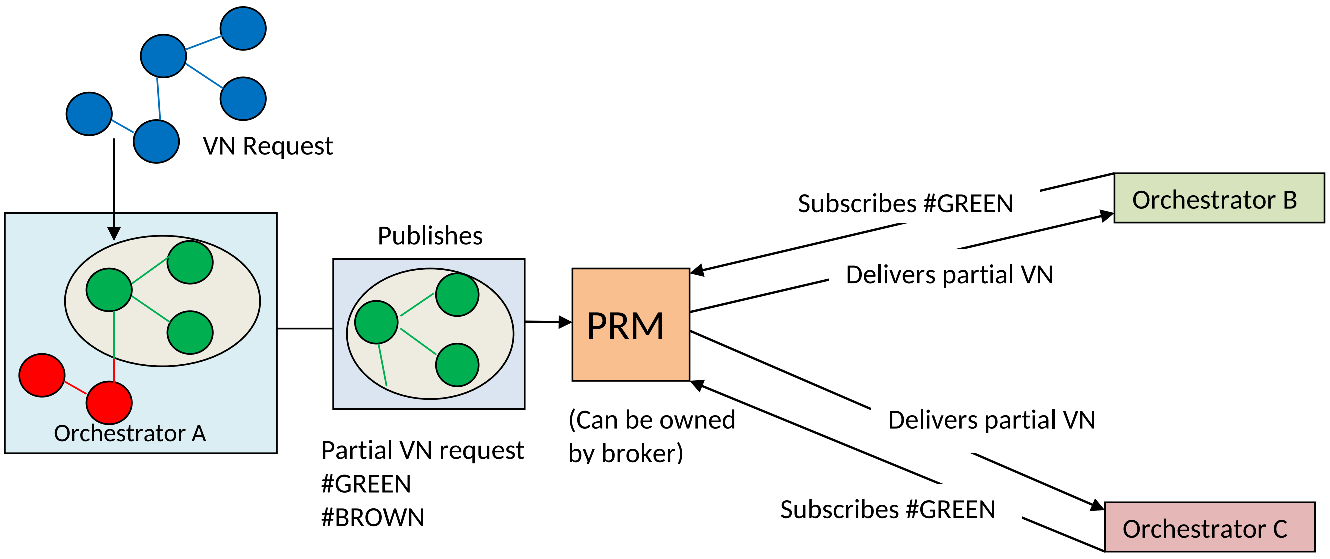


Figure 17: Publish-Subscribe Model 1

Now at the receiving end of partial VN offers, the orchestrator's SRM wakes up the system to contact publisher of the partial VN.

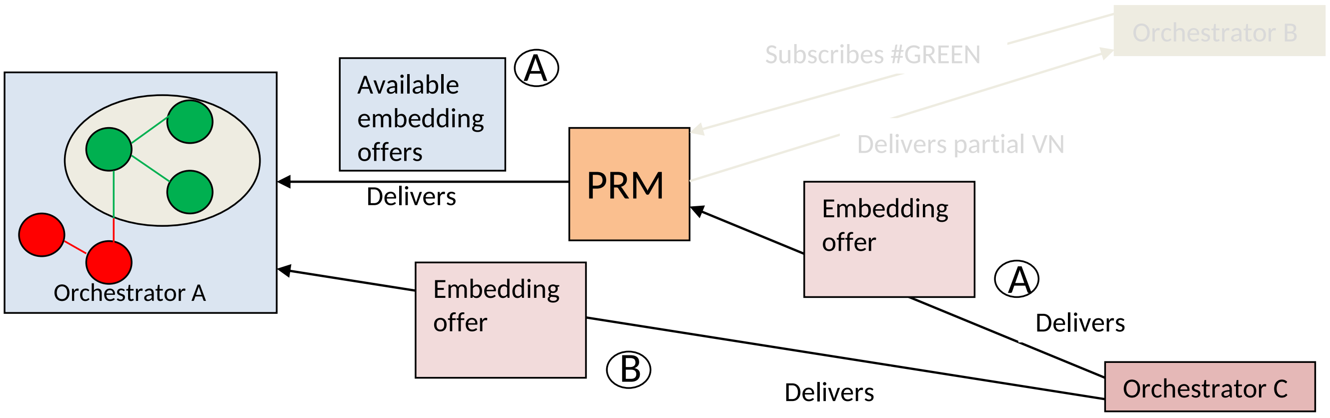


Figure 18: Publish-Subscribe Model 2

After negotiating with the publisher, receiving SRM gets complete details of published partial VN offer. SRM runs algorithm to check whether it can locally support the offer in its domain. Figure (fig .18) below depicts receiving end of the orchestration. Here both orchestrator B and C receive notification. But only the request of orch. C is accepted by the publisher (orch. A). So orch. C sends an embedding offer saying it can embed the partial VN published by orch. A. If orch. A accepts the request from orch. C then the embedding procedure is continued together by orch. A and C, which results in complete embedding of that VN request. If only part of partial request is supported by orch. C then the remaining part is jointly published to PRM by orch. A and C.

There might be more than one orchestrator that receives the same partial VN offer, in which case, all those orchestrators send embedding requests to the publisher. Multiple offers are queued up in the PRM and it depends on the publishing orchestrator to select one of the requests from the PRM (option A) based on optimization criteria. Embedding offer can be sent via PRM directly (optin B) to the requester also. By this way a VN request can be successfully embedded across multiple domains.

Now we have an overview of embedding process with the help of orchestrator components and pub-sub system. The implementation of the above explained concepts require three main tasks. At first we need to implement embedding algorithm for partial embedding that is able to calculate part of VN that can be embedded locally in operators own domain. We need a communication system between operators for sharing the partial VNs and offers. At last we need a orchestrator that can manage embedding a complete VN request across multiple operator domains. The following sections provides complete details on implementation phase of these three tasks. The tasks also fulfil the requirements of the thesis work mentioned in the introduction part.

3.1 Requirement I: Partial Embedding

At first we present the approach for embedding partial services on the physical substrate. To do this, we model an MILP for operator A considering another operator's topology as a pseudo node with infinite resource. In such scenario, operator A considers other nodes as infinite nodes that can host any number of VNs because they have infinite physical resources. Based on such a model and the knowledge of cost of hosting the other operators, our MILP can try to minimize the total cost of hosting the service over this entire modelled graph. The solution to the MILP results in a proposition of the parts of the VN that need to be placed in Operators AâĂŹs network and the parts that cannot be hosted by operator A. The latter part may then be exposed to other operators for evaluation. Therefore instead of exposing individual topology details, operator A exposes the partial VN request. With this approach, operators need not to expose their topology details to others.

This Section provides the solution on how to divide VN request in to parts that are mapped to operators own node and other infinite nodes. We present an ILP formulation that enables the operator to calculate parts of the graph that it can host in its own topology and the part that cannot be served by itself. We name this kind of embedding a VN partially in different domains as "Partial Embedding". We present two albeit related approaches in the following subsections: i) the domain representation where the operators own domain is abstracted ii) where the operators own domain is not abstracted. The general concept of the solution is the abstraction of each domain to infinite nodes, where in all requests can succeed. However, hosting a request in that infinite node incurs an additional cost factor of μ_d where 'd' is the index of the domain the infinite node represents. The objective then is to minimize the cost.

Self Topology Abstraction: As explained earlier, the domain is itself represented as a pseudo-node and the neighbouring nodes are represented as infinite nodes as shown in 14 For every operator, his own domain is considered as known domain and other domains as infinite domains. The edge nodes of a pseudo node A (known domain) are connected to infinite nodes. These infinite nodes can host everything. The capacity, bandwidth and other resources are infinite so that these nodes can process every requests and host every virtual network. Embedding virtual request on known node (operator A) incurs cost 'w'. Cost of hosting in infinite nodes is μ times costlier than hosting on our own node. Typically, $\mu \geq 1$ but this is not necessary.

Since we use the same basic model presented in the work EdWiN, most of the formulation remains same. Also, we use the concept of abstracting physical nodes and interfaces that we defined in [13]. At first, we present basic graph model for physical network and virtual network request which is modified to support partial embedding. Following this, modified formulation for resource constraints and goal function is presented. Hence we consider:

(A) Basic Model: As defined in EdWiN project, same graph model is used here for both physical network and virtual network request i.e a graph $G = (V, E)$ where 'V' is a set of vertices and 'E' is a set of edges connecting those vertices. Therefore, physical substrate is represented as a graph $G^S = (V^S, E^S)$ and virtual request as $G^V_g = (V^V_g, E^V_g)$. The physical substrate consists of 'N' number of domains (pseudo nodes and infinite nodes) represented as $V^S = \{n_1, n_2 ... n_N\}$. Without loss of generality say for $i = 1 ... N_{known}$ represent pseudo nodes where the capacity values are known while for $i = N_{known} + 1$ represent infinite nodes.

(B) A price 'p' associated with embedding each of these sub-graphs into physical substrate, which brings in p_g price for embedding g-th request.

(C) μ_i^C is the factor for cost of embedding compute resource in a domain 'i'. Similarly μ_i^B is the cost for internal bandwidth allocation in every i-th domain.

a) Each physical substrate node n_i where $i\epsilon\{1...N\}$ has I_i interfaces. Each interface for a node n_i is defined as n_{i_p} with $p\epsilon\{1...I_i\}$.

b) we formulate link as $\{e_{i_p j_q} = \{n_{i_p}, n_{j_q}\}\}$ where the link exists between p-th interface of node 'i' and q-th interface of node 'j'. An interface can only be associated to a single link resulting in a maximum of $L \leq N(N-1)$ edges.

c) Node capacity matrices, for instance: CPU matrix $[C_i]_{i=1...N}$, where $C_i \geq 0\ \forall i\epsilon\ \{1...N\}$. Identical matrices and equations can be created for other node resources such as storage and memory.

d) Link capacity matrix $B_c = [B_{c_{i_p j_q}}]$, $i, j\epsilon\ \{1...N\}$, $p\epsilon\ \{1...I_i\}$, $q\epsilon\ \{1...I_i\}$, where $B_{c_{i_p j_q}} \geq 0\ \forall i, j\epsilon\ \{1...N\}; e_{i_p j_q}$ has capacity $B_{c_{ij}}$; We assume G^P is an undirected graph, thus $B_{c_{i_p j_q}} = B_{c_{j_q i_p}}$. This has no significant impact on the formulation as the directional bandwidth would only introduce one equation in each direction. Note that $B_{i_p j_q}$ represents the bandwidth available between interfaces p and q belonging to node n_i.

(D) Virtual Request Model:

VN requests are resolved one by one by embedding G VN requests, $G_g^V = (V_g^V, E_g^V)$. We split each VN request $g\epsilon\{1...G\}$ link wise into g_k 2-node sub-graphs. The k-th sub-graph, $k\epsilon\{1..g_k\}$, of the g-th virtual network is characterized by:

a) Source and Destination nodes $v_{s_{g_k}}$ and $v_{d_{g_k}}$ with capacity requirements $C_{s_{g_k}}$, $S_{s_{g_k}}$ and $C_{d_{g_k}}$, $S_{d_{g_k}}$ respectively.

b) The requested directional bandwidth b_{g_k}

c) Boolean variables

$$y_{s_{g_k}}^i = \begin{cases} 1, & \text{if } n_{s_{g_k}} \rightarrow n_i \\ 0, & \text{if } n_{s_{g_k}} \not\rightarrow n_i \end{cases}$$
$$y_{d_{g_k}}^i = \begin{cases} 1, & \text{if } n_{d_{g_k}} \rightarrow n_i \\ 0, & \text{if } n_{d_{g_k}} \not\rightarrow n_i \end{cases} \tag{39}$$

where a → b implies a is embedded in b in the solution.

d) The embedding solution for a virtual link is defined by the boolean variable $x_{i_p j_q}^{g_k}$, identifying if bandwidth is allocated on the physical link $l_{i_p j_q}$ for the k-th sub-graph of the g-th virtual network request. Therefore,

$$x_{i_p j_q}^{g_k} \epsilon \{0, 1\}\ \forall i, j\ \epsilon\ \{1...N\}; p\epsilon\ \{1...I_i\}; q\epsilon\ \{1...I_j\} \tag{40}$$

where $x_{i_p j_i}^{g_k} = 0,\ \forall i\ \epsilon\ \{1...N\}; p\epsilon\ \{1...I_i\}$

e) Furthermore, allocation of any bandwidth between any pair of interfaces on the same node n_i results in a compute resource consumption $\{C_{through}^i\}$ is defined same as the equation Eq. 24. Similar equation for storage and memory requirement can be formulated. β determines the influence of allocated link through bandwidth in the resource of the node it passes through. Typical β values for currently available high end router slots can be calculated by assuming that the maximum switching capacity of the slot uses all the resource available therein. Then the ratio of the maximum switching capacity to the maximum supported link bandwidth gives us a good estimate of β.

f) Finally the whole VN request G_g^V, $\forall g\ \epsilon\ \{1...G\}$, is characterizes by a boolean variable:

$$y_g = \begin{cases} 1, & \text{if } G_g^V \rightarrow G^P \\ 0, & \text{if } G_g^V \not\rightarrow G^P \end{cases} \tag{41}$$

Intra-node bandwidth model: We are using 'Independent Bandwidth Model' where, between each pair of interfaces in any node i, there is a bandwidth $B_{i_p j_q}$ that is independent of the consumption between any other pair of resources. Therefore assuming the bandwidth consumption in both directions equally

$$B^{Available}_{i_p j_q} = B^{total}_{i_p j_q} - \sum_{g=1}^{G}\sum_{k=1}^{k_g}\{x^{gk}_{i_p j_q} + x^{gk}_{i_q j_p}\} \tag{42}$$

(E) Constraints:

The constraints for the resource allocation are applied only to the known nodes. As infinite nodes are assumed to have infinite resources, these constraints are not applicable to them that is for $i \geq N_{known} + 1$. Hence the previous equation for bandwidth constraint (Eq. 30) can be modified as:

$$\sum_{g=1}^{G}\sum_{k=1}^{k_g} b_{g_k}\left(x^{gk}_{i_p j_q} + x^{gk}_{j_q i_p}\right) \leq B_{i_p j_q} \text{ if } i \leq N_{known} \text{ OR } j \leq N_{known} \tag{43}$$

For the capacity constraints at the nodes, Eq. 31 can be modified as:

$$\sum_{g=1}^{G}\sum_{k=1}^{K_g}\left(C_{s_{gk}} U_{s_{gk}} y^i_{s_{gk}} + C_{d_{gk}} U_{d_{gk}} y^i_{d_{gk}}\right) + C^i_{internal} + C^i_{through} \leq C_i \quad \forall i \leq N_{known} \tag{44}$$

where

$$U_{s_{gk}} = \begin{cases} 0, & \text{if } \exists k' < k : v_{s_{gk}} \equiv v_{s_{k'g}} \text{or } v_{s_{gk}} \equiv v_{d_{k'g}} \\ 1, & \text{otherwise} \end{cases}$$

$$U_{d_{gk}} = \begin{cases} 0, & \text{if } \exists k' < k : v_{d_{gk}} \equiv v_{s_{k'g}} \text{or } v_{d_{gk}} \equiv v_{d_{k'g}} \\ 1, & \text{otherwise} \end{cases}$$

where 'v' represents virtual node and $\equiv$ means ïs same as". And the parameters $U_{*_{gk}}$ avoids same virtual node to act as source and destination node for different 'k' sub-graphs. Equation similar to Eq .44 can be derived for other resources like memory and storage.

(F) Topology and Embedding Constraints:

Formulation for topology and embedding constraints remains same as defined in the previous work (2.3.5). Hence the equations Eq .32 to Eq .37 remains same and is used in the thesis work as it is.

(G) Goal Function:

The goal is to maximize the profit for the operators own domain and to minimize the resource utilization in the infinite nodes. as the cost of embedding is higher in the infinite nodes. Therefore goal function is formulated as:

$$\begin{aligned} \sum_{g=1}^{G} p_g y_g - & \sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N}\sum_{p=1}^{I_i}\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{q=1}^{I_j} \mu^B_{i_p j_q} b_{g_k} x^{gk}_{i_p j_q} \\ & - (1+\gamma/\beta)\sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N}\sum_{p=1}^{I_i}\sum_{q=1}^{I_i} \mu^B_i b_{g_k} x^{gk}_{i_p i_q} \\ & - \gamma/\alpha \sum_{g=1}^{G}\sum_{k=1}^{K_g}\sum_{i=1}^{N} \mu^B_i b_{g_k} z^i_{g_k} \\ & - \sum_{i=1}^{N}\sum_{g=1}^{G}\sum_{k=1}^{K_g} \mu^C_i \left(C_{s_{gk}} U_{s_{gk}} y^i_{s_{gk}} + C_{d_{gk}} U_{d_{gk}} y^i_{d_{gk}}\right) + C^i_{internal} + C^i_{through} \end{aligned} \tag{45}$$

where $\gamma = \frac{\text{percentage of free links in substrate}}{\text{percentage of free node resources in substrate}}$

The first term ensures that the embedding succeeds with maximized profit for operator's own domain. The variable 'p_g' represents the price for each 'g' VN requests. The rest of the term tries to minimize resource consumption related to the allocation of bandwidth. CPU resources are not considered as they are used in a fixed amount in every request if it were to succeed. The second term tries to minimize the consumption of bandwidth on the link connecting two different physical domains. The bandwidth cost μ_i^B associated with each domain 'i' also determines the solution for each VN request. Similarly the third term tries to minimize amount of internal bandwidth allocated to virtual links inside a physical domain. The "1"from the co-efficient $(1+\gamma/\beta)$ represents inter-node bandwidth used while β in γ/β stands for compute resource used. And γ is used to balance the utilization of links vs nodes while embedding links. The last term is used to minimize the compute resources used while embedding links into nodes. The fourth term tries to minimize the compute resource used while embedding links into nodes. And the last term tries to minimize the compute power used for the virtual nodes while mapping onto physical node.

The following section presents details on the experiment set up and in house simulation tool used to implement the proposed algorithm. After that, we present details on the results gathered for the partial embedding process.

Simulations were executed on a server by solving MIP formulation using the Gurobi library running in Java on 4 cores. A fixed substrate topology with 4 different operators is created as shown in the figure 19. In this topology, one node is made as a known node (operator's own node) and other nodes as unknown nodes (infinite nodes). Substrate topology has two layers with first layer having inter-operator topology. This layer includes $N_{operator}$ pseudo nodes abstracting the the operators. The second layer is a intra-operator topology consisting of 'N' number of nodes in each pseudo-node or operator domain. Both inter-domain and intra-domain topologies are randomly generated with random resources and random connection between the pseudo nodes. In intra-operator topology, initially each node has random node capacity generated between 50 and 100 units. Link capacity is randomly generated and uniformly distributed between 50 and 100 units. For each operator node (pseudo-node), available node capacity is equal to the summation of the capacity of the node it contains.

In the figure (fig .19) the rectangular boxes represent nodes and the lines represent links connecting different nodes (domains/operators). The node (node Id: 3741663) with dark shade is known node and others (node Ids: 3741664, 3741665, 4281031) are infinite node. Known nodes have finite number of resources, for instance, the known node here has 70 CPU units, 91 memory and 51 storage units. The overall bandwidth capacity of the known node is 320 units. But the infinite nodes have infinite resources with CPU, storage and memory as 100000 units. The external link connecting two pseudo nodes, out of which if one is infinite node, have infinite bandwidth resource. Since we have configured 2 digit numbers for VN resources (CPU, memory, storage), 100000 units is considered as infinite resources.

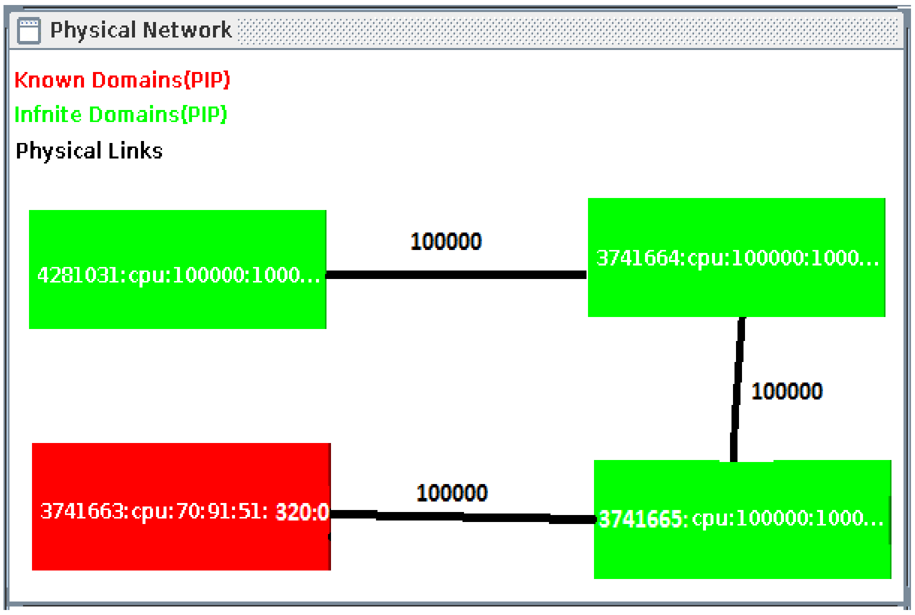

Physical Network
Known Domains(PIP)
Infnite Domains(PIP)
Physical Links
4281031:cpu:100000:1000...
100000
3741664:cpu:100000:1000...
100000
3741663:cpu:70:91:51: 320:0
100000
3741665:cpu:100000:1000...

Figure 19: Substrate Topology

VN are created by generating virtual network with four virtual nodes as shown in the figure 20. The capacity requirements of the virtual nodes and links are integer numbers randomly generated and uniformly distributed, respectively in the range 1 to 100xVNF (Virtual Node Factor) units and 1 to 100xVLF (Virtual Link Factor) units, where VNF, VLF $\epsilon(0,1)$ are the simulation parameters that define the ease for incorporating virtual nodes in physical nodes and virtual links in physical links.

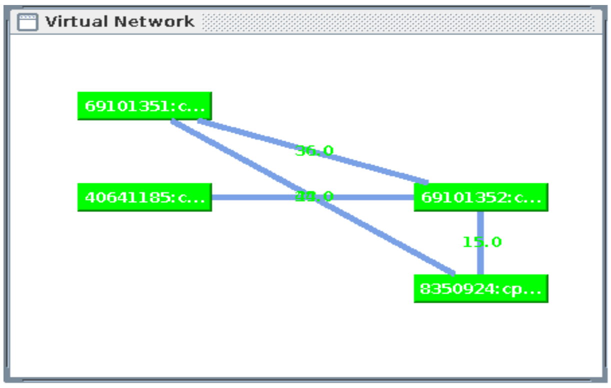
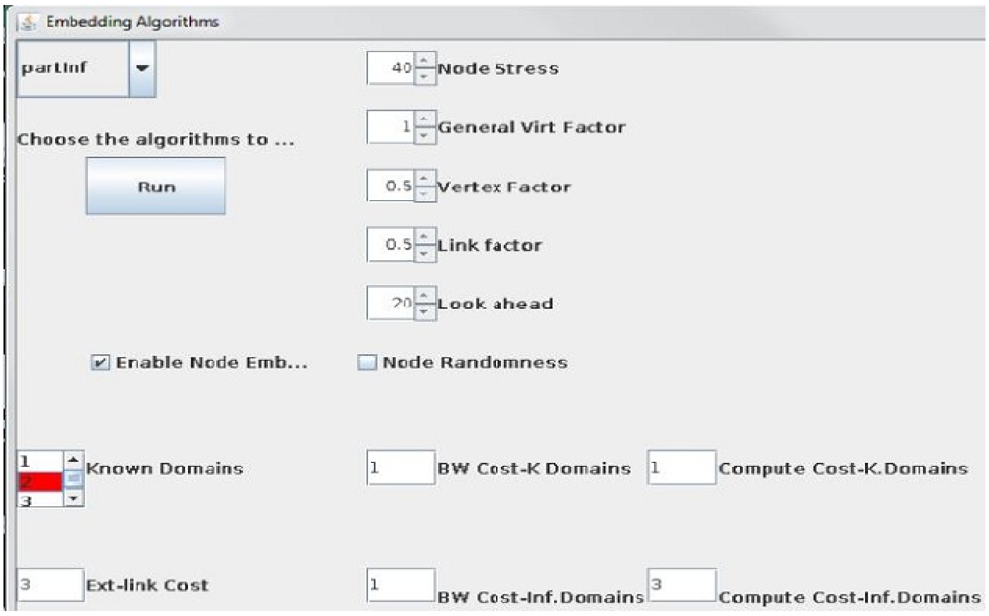


Figure 20: Virtual Network Request Topology

The following section provides the details on configuring the orchestrator part. Most of the variables defined the formulation are assigned values in this configuration part. A simple GUI is designed to accept values for different variables and provide a visual screens with controls to understand embedding of each VN request.


Embedding Algorithms

partInf

Choose the algorithms to ...

Run

40 Node Stress

1 General Virt Factor

0.5 Vertex Factor

0.5 Link factor

20 Look ahead

☑ Enable Node Emb...

☐ Node Randomness

1 / 2 / 3 Known Domains

1 BW Cost-K Domains

1 Compute Cost-K.Domains

3 Ext-link Cost

1 BW Cost-Inf.Domains

3 Compute Cost-Inf.Domains

Figure 21: Configuration GUI

The cost factors explained in the goal function Eq .45 can be set using the GUI inputs. GUI (figure .21) allows to assign different values for compute cost and bandwidth cost in known nodes and infinite nodes. The cost factors μ_i^B and μ_i^C are set keeping the cost for known nodes cheaper compared to the costs for resources in infinite nodes. The configuration GUI for each orchestrator provides option to select more than one known nodes leaving the remaining as infinite nodes by default. The VN requests are generated simultaneously and fed to the embedding algorithm which process the request one by one. After solving the MIP problem, the embedding algorithm embeds the requests characterized by the variable $y_g = 1$ by reducing the capacity of the substrate. Every time a VN request is embedded successfully, the resource capacity of the underlying physical domains get decreased by the amount of resources allocated to that VN request. For every VN request, the algorithm while embedding resources in the first level (pseudo-node level) also performs embedding in the domain level (inside domains) simultaneously.

Table 1: Cost Configuration Table

Configuration		
Variable	Description	Value assigned
p_g	Price of g-th VN request	1000 to 1500
$\mu^B_{i_p j_q}$	External link bandwidth cost	3
μ^B_i	Internal link bandwidth cost for domain 'i'	Known nodes=1, infinite nodes=1
μ^C_i	Compute cost for each domain 'i'	Known nodes=3, infinite nodes=5
α	Switching Alpha	100
β	Switching Beta	100

The values for the corresponding cost factors used in the experiment is tabulated (Table 1). The input textboxes in the GUI corresponds to the rows in the table. For the price 'p_g', value is assigned while generating VN request graph. For β and γ value is assigned for in the code. Total number of pseudo-nodes, number of nodes inside each pseudo-node and number of nodes in virtual network request can be changed in the simulation code. The dropdown field "Known domains" in the GUI accepts multiple input and represents number of known domains. It is possible to select any of the pseudo-nodes in the network. The field "BW Cost-K Domains" accepts input for the variable μ^B_i (internal link bandwidth cost), but only for selected known nodes. Similarly the field "BW Cost-Inf. Domains" accepts input for the variable μ^B_i (internal link bandwidth cost), but only for infinite nodes. For the variable μ^C_i values in the fields "Compute Cost-K. Domains" and "Compute Cost-Inf. Domains" are used for known nodes and infinite nodes respectively. For external links connecting two pseudo-nodes, value from the textbox Ëxt-link cost" is used. By clicking the run button with all the values assigned, the algorithm starts mapping virtual nodes while displaying visual graphs for physical network and VN request. Simulation also provides visuals of mapping virtual nodes and link on to physical network.

The figure below (Figure .22) depicts embedding process with a VN request containing four virtual nodes with node Ids 13541991, 13541992, 83121213 and 83121214. The right-hand side frame shows embedding of these four virtual nodes on to the physical network containing five different domains (pseudo nodes). For the simulation we have selected one pseudo node as known domain (red/dark coloured rectangle). The known domain (node Id: 3741663) has finite number of resources (CPU:270 units, storage:45 units and memory:12) and belongs to the operator where the algorithm is running. Other pseudo nodes except the node: 3741663 is considered as infinite nodes and the resources are set to high numbers (CPU:100000 units, storage:100000 units and memory:100000). Now in this scenario, the orchestrator inputs the virtual network request by splitting them edge-wise sub-graphs. These sub-graphs are embedded one by one according to the formulation we have presented. In the figure, we can see the result of the algorithm showing, the virtual nodes 83121213 and 83121214 are embedded on the known node (3741663) and two other virtual nodes 13541991, 13541992 are not embedded on the known node. The small rectangular boxes overlapping on bigger rectangles represents virtual nodes that are mapped on to the physical domains.

This fulfils our first requirement of partial embedding. Now our algorithm is able to decide how many virtual nodes and links can be embedded on the known node and how many cannot, from a single VN request. There are other scenarios where the whole VN request can be embedded in one single known node and whole VN request fails to embed on the known node. Most of the previous algorithms did not allow partial embedding,stating whole VN request fails if any of its nodes and links cannot be embedded. The embedding solution here depends mainly on the embedding cost of different pseudo nodes. This brings in three cases where:

1. Embedding cost in Known nodes are cheaper than embedding in infinite nodes: In this case the orchestrator embeds most of the virtual networks in known nodes. After some time VN request gets split between the domains based on the objective set in the goal function. The resources of infinite nodes are used when the known nodes does not have sufficient amount of resources to serve.

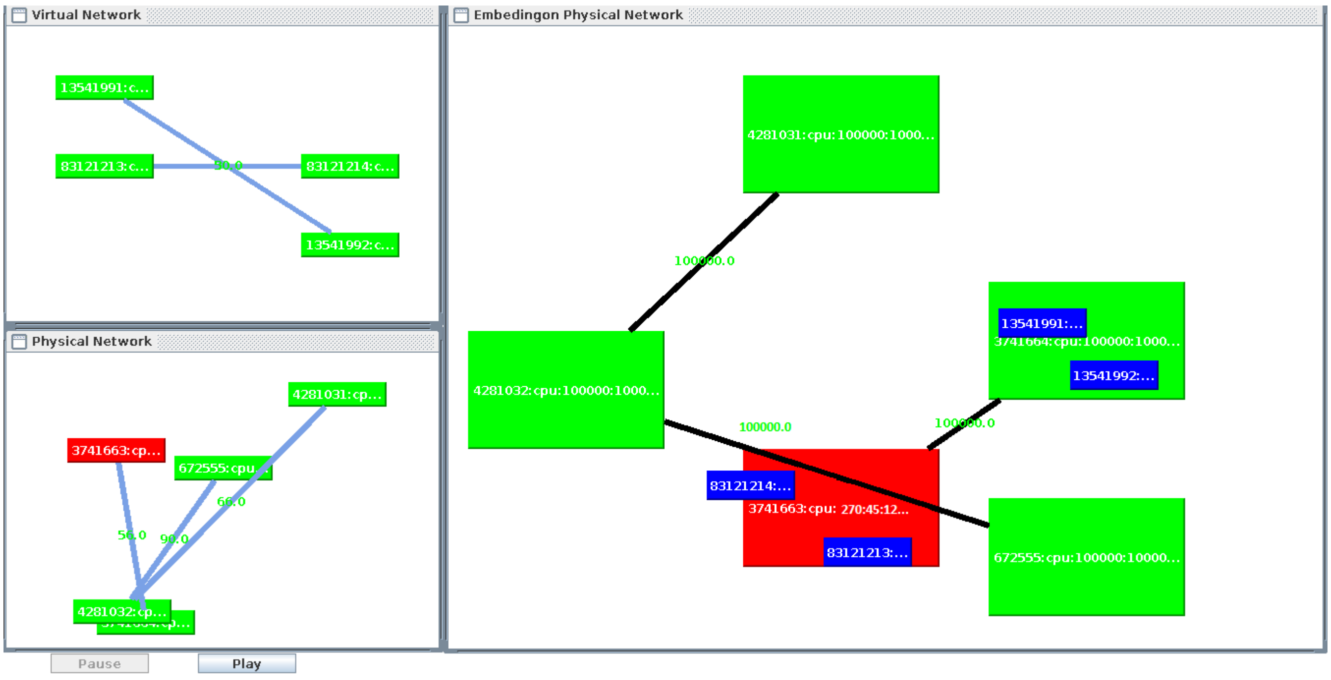


Figure 22: Partial Embedding

2. Embedding cost in Known nodes are costlier than embedding in infinite nodes: In this case all of the VN requests are embedded only in infinite nodes. None of the request fails because infinite nodes have infinite resource which can serve any number of VN request with any type of resource demands.

3. Cost of embedding in Known nodes and infinite nodes are same: Here VN request is embedded on any domain which has sufficient amount of resources for the VN request.

Results: In our experiment we assigned low embedding costs to known domains and high embedding costs to infinite domains (table .1), so that operator's own domain makes high profit by leasing its resources to more number of virtual networks. The results gathered in the experiment shows that, in the beginning our algorithm starts allocating known node's resources to VN requests till there is sufficient amount of resource to embed. After that the resources from infinite nodes are used to serve the upcoming VN requests. We run the simulation with 100 virtual network requests with each request having different topology compared to others. Same number of virtual nodes are used for every VN request. Only difference is the connection between the virtual nodes and the resource demand they have. The following results depicts average resource allocation for same set of 100 VN request for 10 different physical network topologies keeping the embedding costs and other parameters same.

The figure (fig .23) shows the allocation of CPU resource for each VN request. We mainly considered CPU demand and bandwidth demand as VN resource demands. As mentioned earlier each virtual node has CPU demand in the range of 1 to 100xVNF CPU units where VNF is taken as 0.5. Because of the embedding cost in known nodes being cheaper than infinite nodes, the algorithm starts using maximum resource in the known nodes. Also because of higher embedding cost, resource from infinite nodes are only used when there is no sufficient amount resource for a VN. This can be observed from the figure where the entire resource demands from the first few VN requests are served only in known nodes (Red in colour). That is, 100% of the resource demands from VNs are allocated in the known node. After some time, this percentage gradually decreases because of insufficient resource in known domain. Then the resource from infinite resource are used for next set of VN requests. For instance, after some 25 VN requests, more than 50% of the resource demand is served by infinite nodes. And when the VN request is nearer to 90 only 20% of resource demand is served by knnown nodes and remaining 80% of the resource is allocated in infinite nodes. It can be seen that for most of the VN requests, resource are allocated both in known nodes and infinite nodes which is a clear indication of "Partial Embedding ".

Similar results are obtained for bandwidth allocation for each virtual link of a VN request. When all the virtual nodes are embedded in known nodes, bandwidth for all the virtual links of this VN request is allocated in the same known node. But in case of partial embedding, more bandwidth is allocated altogether in known and infinite nodes than the actual bandwidth demand by the VN request. This is because of a scenario where a 2-node sub-graph is

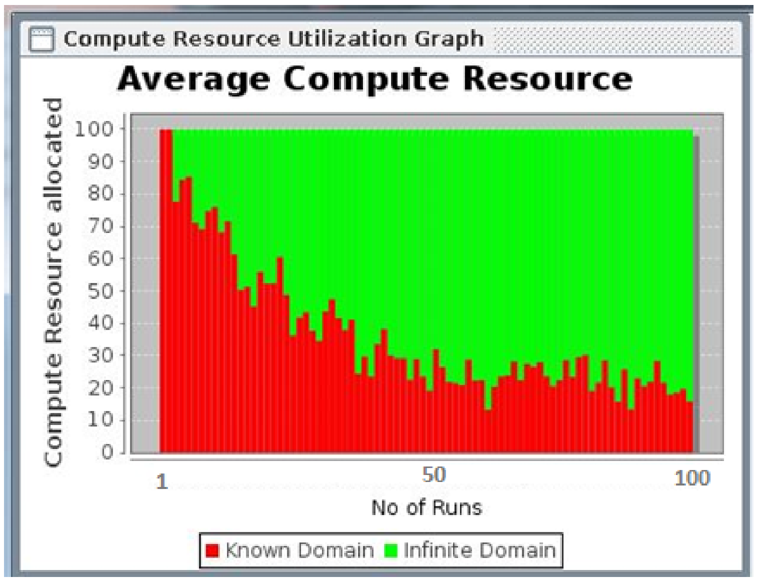


Figure 23: Average CPU Allocation

embedded, one virtual node mapped in known node and one in infinite node. In this case the single link connecting these two nodes need to be embedded in both known nodes and infinite nodes. For instance, consider a 2-node graph with virtual link bandwidth demand of 10 units. When one virtual node is embedded in known domain, a virtual link of bandwidth 10 units is also embedded in that known node. For the second virtual node embedded in the infinite node, again a virtual link with 10 bandwidth units is mapped in this infinite node. So two virtual links each of 10 bandwidth units are embedded. The figure (fig .24) below depicts bandwidth allocation for each VN request in both known nodes and infinite nodes.

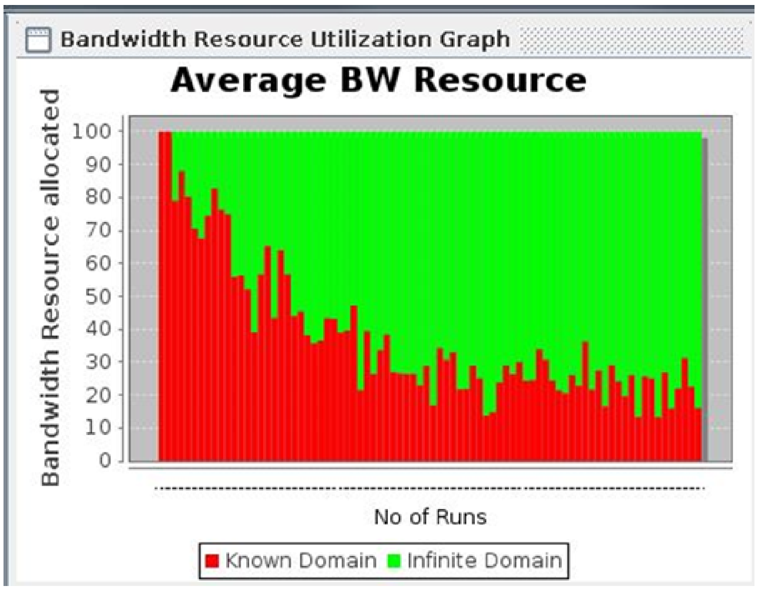


Figure 24: Average Bandwidth Allocation

Another graph below (fig .25) summarizes resource allocation and how the resources are consumed in known nodes for every VN request. This graph is to show the resource status only in known nodes and resource status of infinite nodes are not considered. The graph has four plots with different colours representing type of resource allocated and consumed. The line with blue colour represents CPU allocation in the known nodes. It clearly shows for the first 2-3 VN requests 100% of resource demand is served by known nodes. With the increase in incoming VN requests, this blue line gradually decreases. The blue line plot is another way of visualising the graph in fig .23. The brown line plot represents bandwidth allocated for every VN request. Same as in the CPU allocation, bandwidth allocation in known nodes also decrease with increase in VN requests.

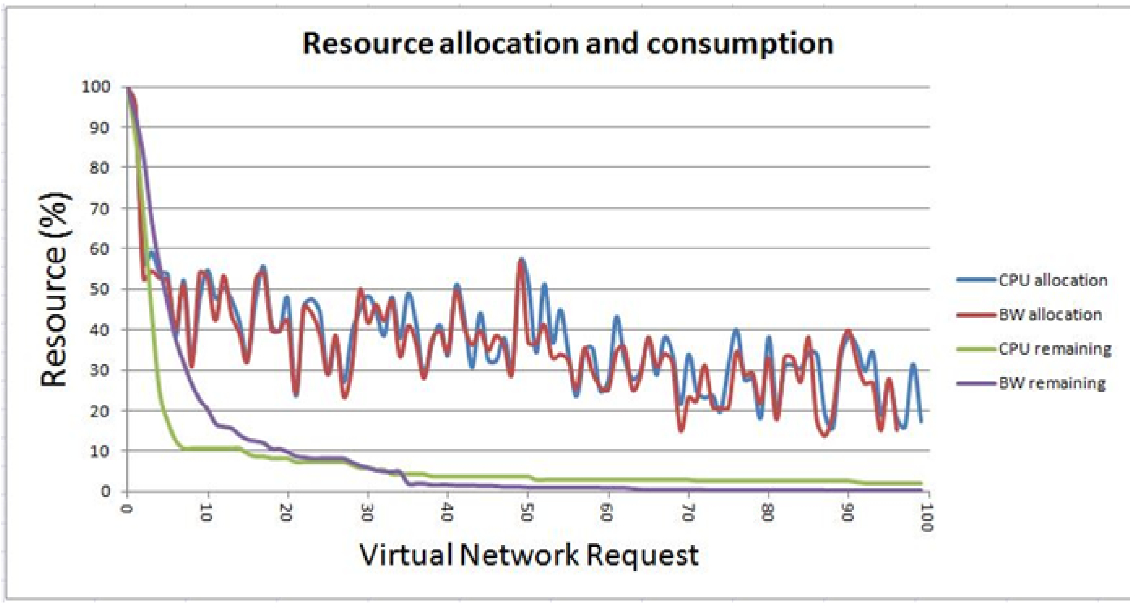


Figure 25: Summary of Resource Allocation and Consumption in Known Nodes

The remaining two plots represents resource availability in known nodes after serving each VN requests. At first the available resource was full (100%) till the first VN request is served. The green line plot is for showing remaining CPU capacity of the known node and purple line plot for remaining bandwidth capacity of a known node. It can be observed that maximum CPU and bandwidth resources are consumed for first few VN requests leaving the resource capacity to almost 10% of the resource the known node had, before embedding any VN request. From the VN request 20 to 80 resource from both known node and infinite nodes are utilized, hence the the remaining resource decrese gradually in known node. The resources of the known node is utilized till it almost becomes 0%. At this stage, algorithm starts looking at infinite node for upcoming VN requests.

3.2 Requirement II:

Starting from partial vn from algo output- nodes links scenarios (few nodes , all nodes of a request), creating partial VN in vnffg format, inserting in db with db script, db pics. why two tables, Pub/sub system as per patent—-till next page.. after that my work of pub sub and pub sub notif. selecting the peer pip all till page 10.

References

[1] K. K. Ashiq Khan, Wolfgang Kellerer and M. Yabusaki, Network Sharing in the Next Mobile Network: TCO Reduction, Management Flexibility, and Operational Independence. DOCOMO Communications Laboratories Europe GmbH.

[2] N. M. M. K. Chowdhury and R. Boutaba, Network Virtualization: State of the Art and Research Challenges. University of Waterloo.

[3] S. S. A. K. David Perez-Caparros, Ishan Vaishnavi, An Architecture for Creating and Managing Virtual Networks. DOCOMO Eurolabs, Munich, Germany.

[4] N. M. M. K. Chowdhury and R. Boutaba, Virtual Network Embedding: A Survey.

[5] J. Lu, Jing; Turner, Efficient Mapping of Virtual Networks onto a Shared Substrate. WUCSE, 2006.

[6] J. R. M. C. Minlan Yu, Yung Yi, Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. Princeton University Princeton, NJ.

[7] T. a. t. n. a. D. Andersen, Unpublished Manuscript, http://www.cs.cmu.edu/âĹijdga/papers/andersen-assign. 2002.

[8] J. Fan and M. Ammar, Dynamic topology configuration in service overlay networks - a study of reconfiguration policies,. in Proceedings of IEEE INFOCOM,, 2006.

[9] M. R. R. N. M. Mosharaf Kabir Chowdhury, Raouf Boutaba, Virtual Network Embedding with Coordinated Node and Link Mapping. University of Waterloo Waterloo, Canada.

[10] W. B. A. D. Z. Ines Houidi, Wajdi Louati, Virtual network provisioning across multiple substrate. Telecom SubParis, France.

[11] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," in 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS), pp. 61–70, May 2014.

[12] I. V. Z. D. A. H. S. B. D. S. R. Guerzoni, R. Trivisonno, A Novel Approach to Virtual Networks Embedding for SDN Management and Orchestration. Huawei European Research Center Riesstrasse 25, Munich, Germany.

[13] I. Vaishnavi, R. Guerzoni, and R. Trivisonno, "Recursive, hierarchical embedding of virtual infrastructure in multi-domain substrates," in Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–9, April 2015.