UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament de Teoria del Senyal i Comunicacions

Self-managed Resources in Network Virtualisation Environments

PhD Thesis Dissertation

by

Rashid Mijumbi

Submitted to the Universitat Politècnica de Catalunya (UPC) in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Barcelona, September 2014

Supervisor: Prof. Joan Serrat Fernández

Co-Supervisor: Dr. Juan-Luis Gorricho

Abstract

Network virtualisation is a promising technique for dealing with the resistance of the Internet to architectural changes, enabling a novel business model in which infrastructure management is decoupled from service provision. It allows infrastructure providers (InPs) who own substrate networks (SNs) to lease chunks of them out to service providers who then create virtual networks (VNs), which can then be re-leased out or used to provide services to end-users.

However, the different VNs should be initialised, in which case virtual links and nodes must be mapped to substrate nodes and paths respectively. One of the challenges in the initialisation of VNs is the requirement of an efficient sharing of SN resources. Since the profitability of InPs depends on how many VNs are able to be allocated simultaneously onto the SN, the success of network virtualisation will depend, in part, on how efficiently VNs utilise physical network resources. This thesis contributes to efficient resource sharing in network virtualisation by dividing the problem into three sub-problems: (1) mapping virtual nodes and links to substrate nodes and paths i.e. virtual network embedding (VNE), (2) dynamic managing of the resources allocated to VNs throughout their lifetime (DRA), and (3) provisioning of backup resources to ensure survivability of the VNs.

The constrained VNE problem is NP-Hard. As a result, to simplify the solution, many existing approaches propose heuristics that make assumptions (e.g. a SN with infinite resources), some of which would not apply in practical environments. This thesis proposes an improvement in VNE by proposing a one-shot VNE algorithm which is based on column generation (CG). The CG approach starts by solving a restricted version of the problem, and thereafter refines it to obtain a final solution. The objective of a one-shot mapping is to achieve better resource utilisation, while using CG significantly enhances the solution time complexity.

In addition current approaches are static in the sense that after the VNE stage, the resources allocated are not altered for the entire lifetime of the VN. The few proposals that do allow for adjustments in original mappings allocate a fixed amount of node and link resources to VNs throughout their life time. Since network load varies with time due to changing user demands, allocating a fixed amount of resources based on peak load could lead to an inefficient utilisation of overall SN resources, whereby, during periods when some virtual nodes and/or links are lightly loaded, SN resources are still reserved for them, while possibly rejecting new VN requests. The second contribution of this thesis are a set of proposals that ensure that SN resources are efficiently utilised, while at the same making sure that the QoS requirements of VNs are met. For this purpose, we propose self-management algorithms in which the SN uses time-difference machine learning techniques to make autonomous decisions with respect to resource allocation.

Finally, while some scientific research has already studied multi-domain VNE, the available approaches to survivable VNs have focused on the single InP environment. Since in the more practical situation a network virtualisation environment will involve multiple InPs, and because an extension of network survivability approaches from the single to multi domain environments is not trivial, this thesis proposes a distributed and dynamic approach to survivability in VNs. This is achieved by using a multi-agent-system that uses a multi-attribute negotiation protocol and a dynamic pricing model forming InPs coalitions supporting SNs resource backups. The ultimate objective is to ensure that virtual network operators maximise profitability by minimising penalties resulting from QoS violations.

Resumen

La virtualización de redes es una técnica prometedora para afrontar la resistencia de Internet a cambios arquitectónicos, que permite un nuevo modelo de negocio en el que la gestión de la infraestructura está desacoplada del aprovisionamiento del servicio. Esto permite a los proveedores de infraestructuras (InPs), propietarios de la red física subyacente (SN), alquilar segmentos de la misma a los proveedores de servicio, los cuales crearán redes virtuales (VNs), que a su vez pueden ser realquiladas o usadas para proveer el servicio a usuarios finales.

Sin embargo, las diferentes VNs deben inicializarse, mapeando sus nodos y enlaces en los del substrato. Uno de los retos de este proceso de inicialización es el requisito de hacer un uso eficiente de los recursos de la SN. Dado que el beneficio de los InPs depende de cuantas VNs puedan alojarse simultáneamente en la SN, el éxito de la virtualización de redes depende, en parte, de cuan eficiente es el uso de los recursos de red físicos por parte de las VNs. Esta Tesis contribuye a la compartición eficiente de recursos para la virtualización de redes dividiendo el problema en tres sub-problemas: (1) mapeo de nodos y enlaces virtuales sobre nodos y enlaces del substrato (VNE), (2) gestión dinámica de los recursos asignados a las VNs a lo largo de su vida útil (DRA), y (3) aprovisionamiento de recursos de backup para asegurar la supervivencia de las VNs.

La naturaleza del problema VNE lo hace “NP-Hard”. En consecuencia, para simplificar la solución, muchas de las actuales propuestas son heurísticas que parten de unas suposiciones (por ejemplo, SN con recursos ilimitados) de difícil asumir en la práctica. Esta Tesis propone una mejora al problema VNE mediante un algoritmo “one-shot VNE” basado en generación de columnas (CG). La solución CG comienza resolviendo una versión restringida del problema, para después refinarla y obtener la solución final. El objetivo del “one-shot VNE” es mejorar el uso de los recursos, a la vez que con CG se reduce significativamente la complejidad temporal del proceso.

Por otro lado, las propuestas actuales son estáticas, ya que los recursos asignados en la fase VNE no se alteran a lo largo de la vida útil de la VN. Las pocas propuestas que permiten reajustes del mapeado original ubican una cantidad fija de recursos a las VNs. Sin embargo, dado que la carga de red varía con el tiempo, debido a la demanda cambiante de los usuarios, ubicar una cantidad fija de recursos basada en situaciones de pico conduce a un uso ineficiente de los recursos por infrautilización de los mismos en periodos de baja demanda, mientras que en esta situación, al tener los recursos reservados, pueden rechazarse nuevas solicitudes de VNs. La segunda contribución de esta Tesis es un conjunto de propuestas para el uso eficiente de los recursos de la SN, asegurando al mismo tiempo la calidad de servicio de las VNs. Para ello se proponen algoritmos de auto-gestión en los que la SN usa técnicas de aprendizaje de máquinas para materializar decisiones autónomas en la asignación de recursos.

Finalmente, aunque determinadas investigaciones ya han estudiado el problema multi-dominio VNE, las propuestas actuales de supervivencia de redes virtuales se han limitado a un entorno de provisión de infraestructura de un solo InP. Sin embargo, en la práctica, la virtualización de redes comportará un entorno de aprovisionamiento con múltiples InPs, y dado a que la extensión de las soluciones de supervivencia de un entorno único a uno multi-dominio no es trivial, esta Tesis propone una solución distribuida y dinámica a la supervivencia de VNs. Esto se consigue mediante un sistema multi-agente que usa un protocolo de negociación multi-atributo y un modelo dinámico de precios para conformar coaliciones de InPs para proporcionar backups a los recursos de las SNs. El objetivo último es asegurar que los operadores de VNs maximicen su beneficio minimizando la penalización por violación de la QoS.

Resum

La virtualització de xarxes es una tècnica prometedora per afrontar la resistència d'Internet als canvis arquitectònics, que permet un nou model de negoci en el que la gestió de la infraestructura de xarxa es desacobla de la provisió del servei. Això permet als proveïdors de infraestructura (InPs), propietaris de la xarxa física substrat (SN), llogar segments d'aquesta als proveïdors dels serveis, que crearan xarxes virtuals (VNs) que a l'hora poden re-llogar-se o utilitzar-se per donar servei a usuaris finals.

No obstant això, les diferents VNs s'han d'inicialitzar assignant els seus nodes i enllaços als del substrat. Un dels reptes d'aquest procés es el requisit de fer un ús eficient dels recursos de la SN. Donat que el benefici d'un InP depèn del nombre de xarxes virtuals que puguin allotjar-se simultàniament en la SN, l'èxit de la virtualització de xarxes depèn en part de quan eficient es l'ús dels recursos de la xarxa física per part de les VNs. Aquesta Tesi contribueix a la millora de l'eficiència en la compartició de recursos en la virtualització de xarxes dividint el problema en tres sots problemes: (1) assignació de nodes i enllaços virtuals a nodes i enllaços del substrat (VNE), (2) gestió dinàmica dels recursos assignats a les VNs al llarg de la seva vida útil (DRA) i (3) aprovisionament de recursos de backup per assegurar la supervivència de les VNs.

La naturalesa del problema VNE el fa "NP-Hard". En conseqüència, per simplificar la solució, moltes de les propostes son heurístiques que es basen en hipòtesis (per exemple, SN amb recursos il·limitats) de difcil compliment en escenaris reals. Aquesta Tesi proposa una millora al problema VNE mitjançant un algorisme "one-shot VNE" basat en generació de columnes (CG). La solució CG comena resolent una versió restringida del problema, per tot seguit refinar-la i obtenir la solució final. L'objectiu del "one-shot VNE" es aconseguir millorar l'ús dels recursos, mentre que CG redueix significativament la complexitat temporal del procés.

D'altre banda, les solucions actuals son estàtiques, ja que els recursos assignats en la fase VNE no es modifiquen durant tot el temps de vida útil de la VN. Les poques propostes que permeten reajustar l'assignació inicial, es basen en una assignació fixe de recursos a les VNs. No obstant això, degut a que la càrrega de la xarxa varia a conseqüència de la demanda canviant dels usuaris, assignar una quantitat fixe de recursos basada en situacions de càrrega màxima esdevé en ineficiència per infrautilització en períodes de baixa demanda, mentre que en tals períodes de demanda baixa, el tenir recursos reservats, pot originar rebutjos de noves VNs. La segona contribució d'aquesta Tesi es un conjunt de propostes que asseguren l'ús eficient dels recursos de la SN, garantint a la vegada els requeriments de qualitat de servei de totes les VNs. Amb aquesta finalitat es proposen algorismes d'autogestió en els que la SN utilitza tcniques d'aprenentatge de màquines per a materialitzar decisions autònomes en l'assignació dels recursos.

Finalment, malgrat que diversos estudis han tractat ja el problema VNE en entorn multi-domini, les propostes actuals de supervivència de xarxes virtuals s'han limitat a contexts d'aprovisionament per part d'un sol InP. En canvi, a la pràctica, la virtualització de xarxes comportarà un entorn d'aprovisionament multi-domini, i com que l'extensió de solucions de supervivncia d'un sol domini al multi-domini no es trivial, aquesta Tesi proposa una solució distribuïda i dinàmica per a la supervivència de VNs. Això s'aconsegueix amb un sistema multi-agent que utilitza un protocol de negociació multi-atribut i un model dinàmic de preus per formar coalicions d'InPs que proporcionaran backups als recursos de les SNs. L'objectiu últim es assegurar que els operadors de xarxes virtuals maximitzin beneficis minimitzant les penalitzacions per violació de la QoS.

Acknowledgments

First and foremost, I would like to thank God, whose many blessings have made me who I am today. Secondly, this thesis would never have its current form without the help and support of many kind people around me, to only a few of whom it is possible to give particular mention here.

I am forever grateful to Prof. Joan Serrat Fernández for having accepted to be my thesis director. Your commitment, patience, guidance and support have been invaluable to the birth of this thesis, and to my general development as a researcher. You have gone beyond the definition of an advisor to assume a role of an "academic father", always giving your best effort not only to help me progress in my PhD, but also introducing and giving me opportunities to work with other respected people in the network and service management community.

I would like to thank my co-advisor Dr. Juan-Luis Gorricho for your relentless efforts in guiding me from the conception of the PhD research topic, to carrying out the research and finally writing this thesis. Without your support, trust and disposition, this thesis would not have been possible.

I express my sincere gratitude to Prof. Filip De Turck (Ghent University - iMinds), Prof. Raouf Boutaba (University of Waterloo), Prof. Ke Xu (Tsinghua University), Dr Javier Rubio-Loyola (CINVESTAV, Tamaulipas), Prof. Kun Yang (University of Essex), Prof. Ramón Agüero (Universidad de Cantabria) and Prof. Steven Latré (University of Antwerp - iMinds) for agreeing to work with me at different stages during the development of the content in this thesis. I would have never dreamt for my development as a researcher to be guided by better professionals.

Special thanks go to my colleagues and professors: Prof. José-Luis Melús, Prof. Rafael Valle, Prof. Ramon Ferrús, Prof. Carles Puente, Prof. Ulises Cortés, Dr. Jeroen Famaey, Dr. Steven Davy, Dr. Sofie Verbrugge, Antonio Astorga, Ricardo Bagnasco, Mario Flores, Raul-Marcelo Ortiz, Maxim Claeys, Niels Bouten, Bram Naudts, Meng Shen, to mention but a few. Thank you all for your friendship, the fruitful informal talks and your respective contributions to this thesis.

I thank my family who have supported me through out my academic and personal development. Despite the distance and self alienation resulting from pursuing my career goals, I have always felt the love and care. Importantly, I would like to thank my mother who has given me unequivocal support throughout my life, as always, for which my mere expression of “thanks mom” will never suffice.

Living in Barcelona as well as all the many cities I have been blessed to live during these years has been a wonderful experience, full of enriching moments. My friends have always given me the most support, without realising that my progress depended on it. Shamim Nassanga, Peace Gakwaya, Farrah Matovu, Hussein Nsamba, Sowali Mukose, Isa Kisingo, Auraham Camacho, Valeria Zuñiga, Cecilia Zuñiga, Cuquis Garcia, Yanet Ponce, Aurora Ruiz, Susana Gutierrez, Pepe Castro, Gerardo Vazquez, Jose Martinez, Samuel Monterro, Thomas Vanhove, Stefano Petrangeli, Ruggero Bettinardi, Alice Del Genovese, Irune Fernandez, Mabel Fernandez, to mention but a few. You have been my big family, and have supported me at all times, with encouragement to work harder, and with laughter and companionship whenever I needed it. Importantly, for you, Marina Khryukina, words will never fully describe what you mean to me. Thank you very much for everything you sacrificed for us and for all the unforgettable moments. I am forever indebted.

Finally, the research in this thesis has been partially supported by various Spanish national and European projects. Special mention to the Ministry of Economy and Competitiveness (MINECO), Government of Spain, for the scholarship that supported the development of this thesis, and FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

This thesis is dedicated to the memory of my father, ***Umar Muramagi****. I wish he could be here to see me become the kind of person he always wanted me to be. I hope that, wherever he is, he rests in peace looking at this important step in my career!*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

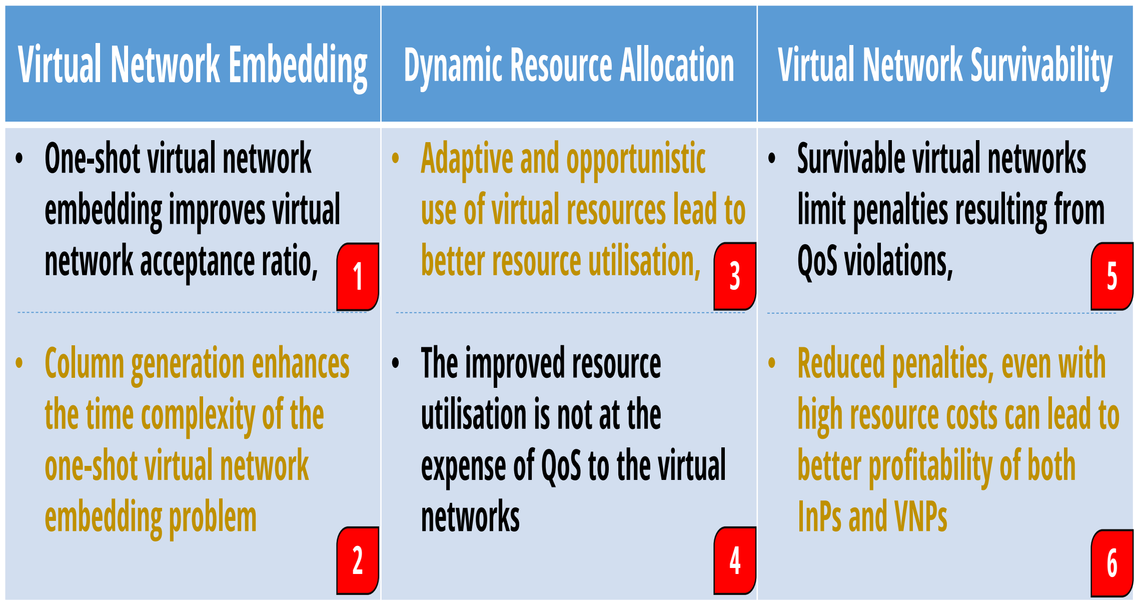
The evolution of the Internet in the last decades has led to a shift from its conception as a mere connectivity network to a content based network [1]. Along with this evolution, Internet users are now concerned not only with being able to communicate, but also getting the right information at the right time and at an affordable price. The expectations and demands of Internet users have risen to levels that are very difficult to achieve with the traditional architectural approach of the same connectivity infrastructure for any type of service offer. Therefore, specialisation of resources and protocol stacks is a must for such diversified service provisioning scenarios. This requires modifications in the current "one size fits all" architecture of the Internet [2]. However, the existence of multiple stakeholders with competing objectives makes it very difficult, if not impossible, for any architectural changes to be made on the Internet. This is the so called ossification of the Internet [3], and can be observed, for example, from the difficulties that have been encountered in the deployment of IP Multicast [4] and IPv6 [5].

Network virtualisation - which is now a subject of various research teams both in academia as well as industry [6] - has been proposed as a Future Internet enabler technique to not only allow for the de-ossification of the current Internet [7] but also to facilitate new and specialised service deployment [3]. It is conceived as a new approach to network design and service delivery aimed at achieving dynamic network and service management, which is important in achieving comparative and competitive advantages in Internet service provision. In a network virtualisation environment [2], multiple service providers are able to create heterogeneous virtual networks to offer customized end-to-end services to end-users by leasing shared resources from one or more infrastructure providers without significant investment in deploying physical infrastructures [8].

1.1 Problem Statement

One of the fundamental requirements in network virtualisation is the assignment of physical network resources to virtual networks [9]. It involves assigning physical node and link resources to virtual nodes and links respectively. The resources can be assigned *offline* (implying that all virtual network resource demands are assumed to be known in advance) or *online*. In addition we can distinguish between *static* and *dynamic* allocation. In static allocation, resources are assigned to a virtual network on creation, and are not adjusted according to changes in the demands of the users, traffic loads, physical resources and infrastructures. Even with the most efficient allocation algorithms, static resource allocation cannot adapt to variations in dynamically changing network environments, and therefore the virtual network becomes resource inefficient in real network conditions [10].

In general, most current approaches [11] have considered the resource assignment approach as only involving the mapping of virtual nodes and links to substrate nodes and paths, also known as the virtual network embedding (VNE) problem [11]. The VNE problem can be represented using a mixed-integer program, and is known to be NP hard [8]. Because of this computational intractability, various approaches to it have been by use of heuristics [12] which make different assumptions (such as assuming availability of unlimited physical resources) so as to simplify the solution. However, even though various constraints and objectives make this problem computationally intractable, the presence of multifarious topologies and possible opportunities to exploit them still leaves enough room for research on customized solutions and better approximation algorithms [2]. In addition, even after the initial mapping step, there



Virtual Network Embedding	Dynamic Resource Allocation	Virtual Network Survivability
• One-shot virtual network embedding improves virtual network acceptance ratio, 1	• Adaptive and opportunistic use of virtual resources lead to better resource utilisation, 3	• Survivable virtual networks limit penalties resulting from QoS violations, 5
• Column generation enhances the time complexity of the one-shot virtual network embedding problem 2	• The improved resource utilisation is not at the expense of QoS to the virtual networks 4	• Reduced penalties, even with high resource costs can lead to better profitability of both InPs and VNPs 6

Figure 1-1: Hypothesis: Each sub-problem with the corresponding expected results

is need to manage the allocated resources so as to ensure efficient resource utilisation. This has so far not received deserved attention in the state of the art [11].

1.2 Hypothesis

The view of this thesis is that an efficient resource management in virtual network environments does not only constitute an efficient initial virtual network embedding step, but also appropriate life cycle management of resources allocated to virtual networks so as to not only ensure that the resources allocated to virtual networks do not remain idle in times of low network load, but also that, in case of failures to substrate network resources, the QoS guarantees to the mapped virtual networks are not violated. Therefore, this thesis divides the resource management problem into three sub-problems: (1) virtual network embedding; (2) dynamic resource allocation; and (3) virtual network survivability; and proposes a solution to each one of them. In Fig. 1-1, the we summarise the hypothesis of each of the sub-problems identified in this thesis. The numbers shown alongside each hypothesis are intended for reference purposes, and will be used through out this thesis.

1.3 Thesis Objectives

This thesis entails development of algorithms for management of resources in network virtualisation environments. The primary objectives of the developed algorithms −beyond the obvious goals of utilisation efficiency, and autonomic allocations of physical resources− are: (1) to carry out virtual network embedding in one-shot; (2) to dynamically adjust resources allocated to virtual networks according to perceived needs; (3) to minimize QoS violations resulting from failures in substrate network resources. To this end, for each the above primary objectives, the following specific points of interest constitute the research carried out in this thesis.

- To investigate the effect −on resource utilisation efficiency− of performing one-shot virtual network embedding, as well as establishing the possible computational time savings achieved by applying a column-generation approach to the solution of the VNE problem,
- To design, and evaluate distributed self-management algorithms −based on one or more machine learning techniques− for the dynamic allocation of resources in network virtualisation environments,
- To evaluate the possible profitability resulting from automated negotiations and dynamic substrate resources pricing by infrastructure providers, aimed at survivability in multi-domain virtual networks.

It is our humble opinion that achieving the above objectives, either fully (as an orchestrated solution) or in part (each of them independently) would constitute a significant contribution to the very important problem of resource management for the Future Internet, and specifically in network virtualisation environments.

1.4 Thesis Technological Scope

The scope of this thesis is with in the activities of proposing solutions to each of the three sub-problems identified in Section 1.2. For each of these sub-problems, we pro-

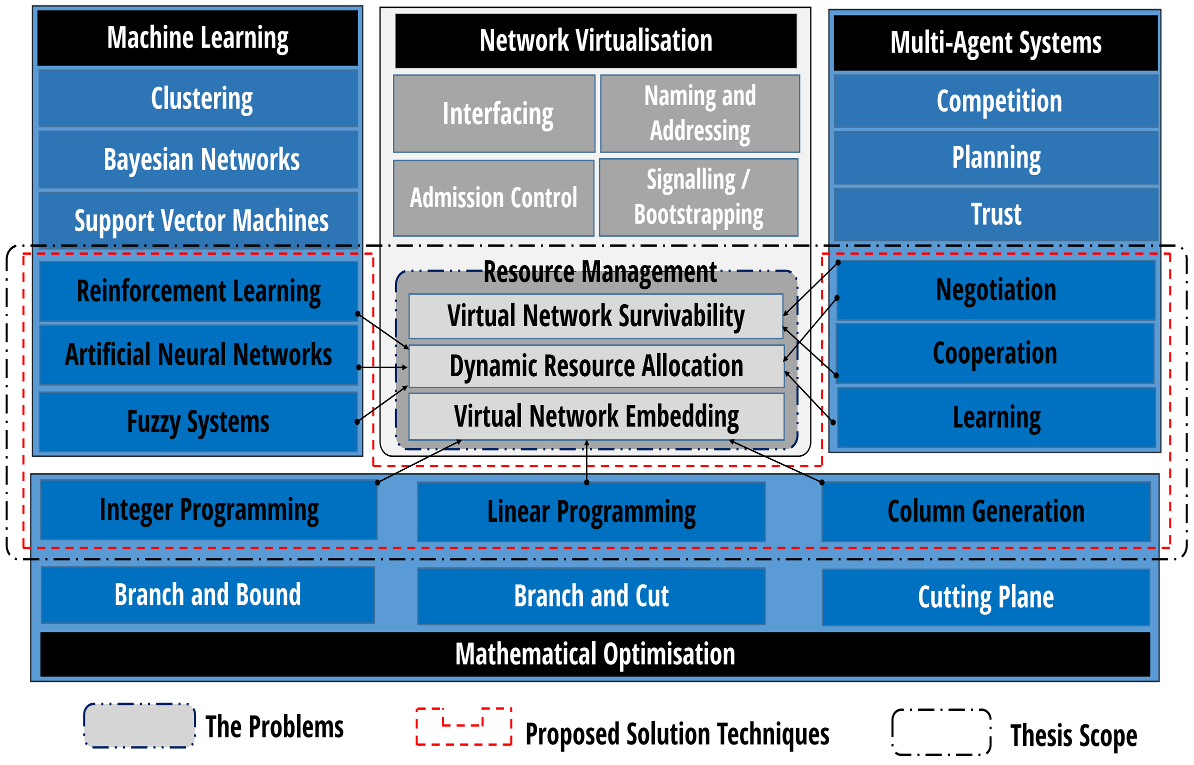


Figure 1-2: Thesis Scope, including the Problems and Solution Techniques

pose a specific solution aimed at achieving the respective sub-objectives as outlined in Section 1.3. As illustrated in Fig. 1-2, we employ tools (integer programming, linear programming and column generation) from mathematical optimisation [13, 14] for the virtual network embedding sub-problem. We then use a combination of machine learning techniques (reinforcement learning [15], artificial neural networks [16] and fuzzy systems [17]) and multi-agent systems [18] (agent cooperation) for dynamic resource allocation. Finally, we apply multi-agent negotiation and cooperation [19] to ensure survivable virtual network embedding. The choice of each of the solution techniques used in this thesis is motivated by the nature and objectives of each of the three sub-problems considered, for example, the one-shot virtual network embedding sub-problem is usually formulated as a mathematical program, which is computationally intractable. The choice of using column generation is due to its (column generation) ability to significantly reduce the computational time of mathematical programs with a large number of variables.

1.5 State of the art

From the scientific point of view, network virtualisation has already received a lot of attention, and as a result, the number of peer-reviewed publications dedicated to it is already quite high. While [2] gives a comprehensive survey of network virtualisation, the work presented in this thesis embraces multiple areas of interest and their application to network virtualisation. The remainder of this section gives a summary of the state of the art for each of the three sub-problems as well as the respective solution techniques, and the major differences between the state-of-the-art and the proposals of this thesis.

1.5.1 Virtual Network Embedding (VNE)

Different approaches to the embedding problem, based on two stages, starting with node mapping and then link mapping, are proposed in [20] and [21]. A coordinated node and link mapping is proposed in [8]. Although the coordination here improves the solution space, the mapping is still performed in two separate stages, hence yielding sub-optimal embedding. A one-shot embedding solution based on a multi-agent system is proposed in [22], assuming unbounded substrate network resources, and all virtual network requests to be known in advance (offline solution). A comprehensive survey of virtual network embedding approaches can be found in [11]. Unlike the proposals in [11], the work in this thesis does not only formulate the VNE as a one-shot mapping problem but also proposes a column generation approach to improve the computation complexity.

1.5.2 Application of Column Generation to VNE

Integer and Linear programming have been applied to a variety of problems in networking. ViNEYard [8] uses mixed integer programming to coordinate node and link mapping in virtual network embedding, while [23] and [24] use mathematical programming for dynamic resource allocation in networks. [25] uses an optimization technique for link mapping (assuming that the virtual nodes have already been mapped to substrate nodes), while [26] incorporates substrate failures in the virtual network embedding problem by formulating the link mapping problem as a path-based multi-commodity flow (MCF) [27] problem. Unlike all these works, the mathematical programming formulation used in this thesis combines both node and link mapping in one stage.

Column generation based formulations for multi-commodity flow based problems are proposed in [28], [29] and [30]. In these formulations the source and end nodes for each flow are known a priori, which reduces the complexity of the problem, compared to the virtual network embedding that we solve in this thesis. Finally, [31] proposes a column generation approach that chooses one from a given set of virtual network embeddings, aimed at maximising substrate network revenue. It is worth noting that this proposal is different both in objective and complexity of problem in a way that it considers that VNE has already been completed and that the embedding configurations are given.

1.5.3 Dynamic Resource Allocation (DRA)

Most existing works on dynamic resource allocation are based on three approaches: control theory, performance dynamics modeling and workload prediction. [32] and [33] are control theoretic approaches, [34] and [35] are based on performance dynamics, while the authors in [36] and [37] use workload prediction. One of the major differences between these works and the work in this thesis is the application domain. Resource allocation in virtual networks presents additional challenges as we have to deal with different resource types (such as bandwidth and queue size) which are not only segmented into many links and nodes, but also require different quality of service guarantees.

With regard to network virtualisation, the authors in [38] propose a dynamic and distributed approach to virtual network embedding, assuming that the virtual nodes are already mapped, while in [39] and [40] the VN embedding problem when the substrate network is dynamically changing is studied. In [20], a solution that considers dynamic requests for embedding/removing virtual networks is presented.

The authors map the constraints of the virtual network to the substrate network by splitting the requirements of one virtual link in more than one substrate link. On the other hand, the proposal in [26] is aimed at network survivability, performing re-embeddings in case of failures in the substrate network. The authors in [41] propose a reactive solution (carried out only when an embedding strategy cannot assign a VN request) which aims at minimising the number of congested substrate links by carrying out link migrations, while [42] proposes algorithms for the problem of efficiently re-configuring and embedding VN requests submitted to a cloud-based data center, requiring that the ISPs submit new requests to modify existing ones, and that only one such request can be handled at a given time. In a related approach, [43] proposes a migration-aware dynamic virtual data center (VDC) embedding framework that also includes VDC scaling as well as dynamic VDC consolidation, while Butt et. al. [44] propose a topology-aware embedding that performs re-embeddings aimed at improving performance of previously embedded VNs. [45] proposes a resource allocation scheme in data centers by considering that the VN link follows a normal distribution, without involving any adjustment to allocated resources after embedding. [46] proposes an opportunistic bandwidth sharing for virtual network mapping that only considers a single link, with several competing requirements for resources. The work in this thesis differs from previous ones in that our resource re-allocations are proactive (not triggered by failed embeddings), autonomous (not triggered by either users or network providers) and do not involve any re-embeddings of already mapped requests. Our proposals also consider a complete network (not a single node or link as in some works), through out its lifetime.

1.5.4 Application of Learning Techniques to DRA

Machine learning techniques have been used in various problems involving dynamic resource management. The authors in [47, 48, 49, 50] propose and simulate dynamic resource allocation in telecommunication networks using reinforcement learning and they show improvements introduced by learning compared to other solutions, while [51] proposes an approach for dynamic resource allocation in Clouds based on maximization of a utility function. Centralised radio resource management architecture for call admission control and multirate transmission control are proposed in [52] and [53] respectively, while [54] proposes a decentralised learning multi-agent system for channel and power algorithm selection based on the construction of behavioral rules. A frequency resource selection approach, based on multi-armed bandit formulation is proposed in [55]. Finally, artificial neural networks and fuzzy systems have been applied to the resource allocation problem in [56] and [57] respectively, while [58] combines both neural networks and fuzzy systems for a joint radio resource management solution.

The major difference between these works and the proposals of this thesis are based on application domain. In a VN environment, the allocated resources are dependent on each other, for example a given virtual link can be mapped on more than one substrate link and the resources allocated to a virtual node may affect the performance of virtual links attached to it, say in terms of increased routing delays. It is also worth noting that the learning environments for the proposals in this thesis involve multiple entities, which is −again− dictated by the nature and possible model of the problem solved in this thesis. Having multiple learning entities in a single system presents several challenges amoung which includes managing possible conflicting actions as well as the need for cooperation to benefit from actions of other agents.

1.5.5 Virtual Network Survivability (VNS)

SVNE [26] incorporates single substrate link failures in VNE, while [59] uses a node migration technique to introduce link survivability. [60] represents each of the substrate nodes of a given substrate network by an agent, allowing the agents to communicate with each other so as to resolve link and node failures. All these proposals are for the single InP case. The major distinction between the proposal in this thesis and the current ones is that we consider survivability in a multi-domain environment, in which case we have to deal with limited information (for example about other InP network topologies), which inevitably calls for negotiation. It is worth noting that extension of a survivable embedding solution from single domain to multiple domains

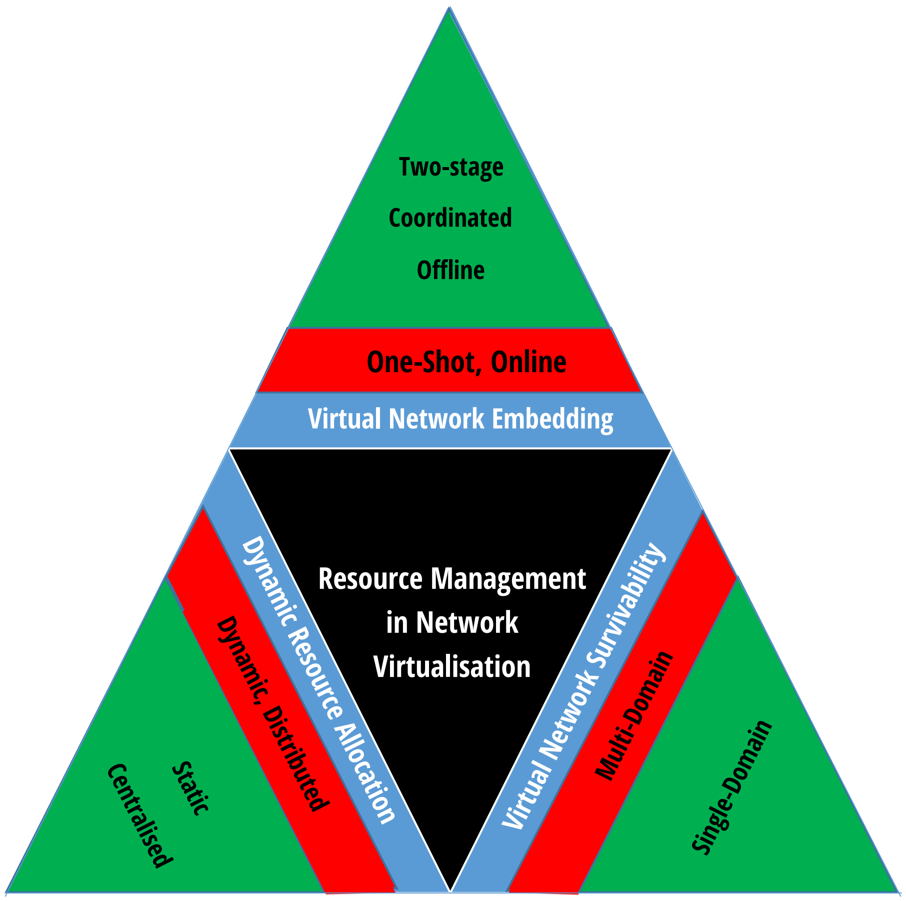

Two-stage
Coordinated
Offline
One-Shot, Online
Virtual Network Embedding
Resource Management in Network Virtualisation
Dynamic Resource Allocation
Dynamic, Distributed
Static
Centralised
Virtual Network Survivability
Multi-Domain
Single-Domain

Figure 1-3: State of Art Approaches, showing Gaps Filled by this Thesis

is not trivial since it involves both intra and inter domain link failures [26]. A broad survey on survivable virtual network embedding can be found in [61].

1.5.6 Application of Autonomic Negotiation to VNS

Negotiation and contracting in multi-provider setups have been studied in [62] and [63] focusing on peer-to-peer networks, while [64] and [65] propose multi-market coalition formations of autonomic management systems in competitive environments, where the authors model service quality by “service skills” for which performance level parameters are defined. In all these works, the services on offer are well and easily defined, such that the focus is on service negotiation. However, the VNE survivability problem is more challenging due to its online nature, capacity constraints, connectivity, and end-node constraints on the links. It therefore requires not only determining the negotiation participants, but also dynamically re-defining the service being negotiated. A detailed survey on automated negotiation is given in [66].

1.5.7 Summary

To summarise, in Fig. 1-3, the general problem of this thesis is represented at the centre (in black), surrounded by the three sub-problems (in light blue) which are the focus of this thesis. For each of the sub-problems, the main characteristics of the state-of-the-art proposals are represented (in green), and aspects where this thesis makes contributions (in red).

1.6 Thesis Contributions

This dissertation contributes to the area of resource management in network virtualisation. Specifically, it introduces novel approaches to virtual network embedding, dynamic resource allocation, and virtual network survivability. The main contributions of this thesis are as follows:

- A near optimal one-shot virtual network embedding approach that improves substrate resource utilisation and virtual network acceptance ratio compared to solutions in the state of the art,
- A column generation-based approach that significantly improves the time complexity of one-shot virtual network embedding compared to an optimal formulation,
- A set of distributed reinforcement learning algorithms that allocate resources to virtual nodes and links dynamically, leading to better substrate resource utilisation,
- A combined artificial neural network (ANN) and reinforcement learning (RL) solution in which RL trains the ANN for dynamic resource allocation in network virtualisation,

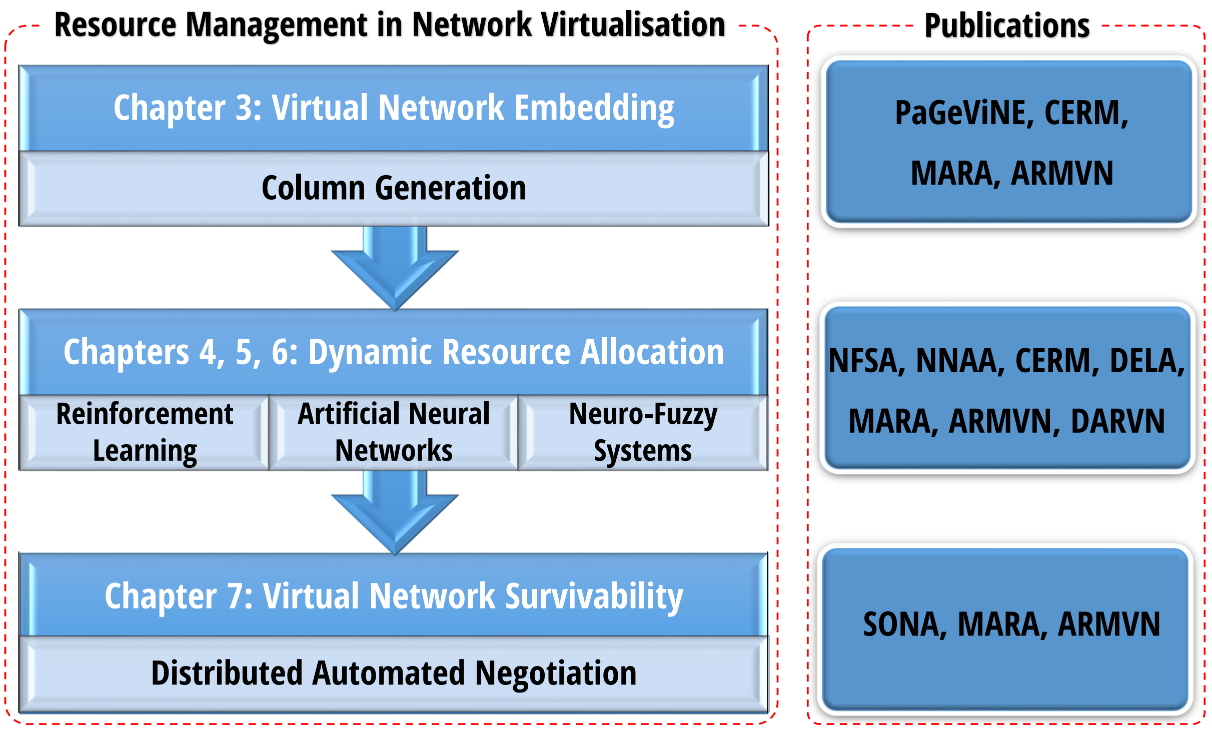

Resource Management in Network Virtualisation
Chapter 3: Virtual Network Embedding
Column Generation
Chapters 4, 5, 6: Dynamic Resource Allocation
Reinforcement Learning
Artificial Neural Networks
Neuro-Fuzzy Systems
Chapter 7: Virtual Network Survivability
Distributed Automated Negotiation
Publications
PaGeViNE, CERM, MARA, ARMVN
NFSA, NNAA, CERM, DELA, MARA, ARMVN, DARVN
SONA, MARA, ARMVN

Figure 1-4: Thesis Contributions, showing relevant publications

- An adaptive neuro-fuzzy system that dynamically learns allocation of substrate resources to virtual networks. A hybrid learning mechanism which uses supervised learning to initialise a rule base and then uses unsupervised learning to adapt the rule base and fuzzy sets of each rule to achieve efficient resource allocation,

- A cooperation scheme that allows the substrate network agents to coordinate their actions so as to avoid conflicts and to share their knowledge so as to enhance their learning speed and improve action selection efficiency,

- A negotiation protocol that ensures virtual network survivability with minimum communication message overhead; VNP and InP negotiation strategies that minimise QoS violation penalties, hence ensuring both VNP and InP profitability; and a dynamic substrate resource pricing model that ensures efficient utilisation of resources.

In summary, to the best of our knowledge, the work presented in this thesis is the first application of column generation to virtual network embedding. It is also a novel contribution of this thesis to apply machine learning techniques to dynamic resource allocation in network virtualisation. Finally, our automated negotiation and pricing proposal is the first foray into network survivability for multi-domain virtual networks. Fig. 1-4 is a graphical summary of the major contributions of this thesis, together with the chapters in the thesis where these contributions are presented, as well as the relevant publications related to the contributions[1].

1.7 Structure of the thesis

The rest of this thesis is arranged as shown in Fig. 1-5. Specifically, **Chapter 2** presents an introduction to network virtualisation, focussing on the common business model, the general architecture of network virtualisation environments, and a brief description of the three main problems that are the subject of this thesis. **Chapter 3** introduces column generation, formulates the one-shot unsplittable flow virtual network embedding problem, and proposes a column generation approach to enhance the computation time complexity of the VNE problem. **Chapters 4, 5,** and **6** propose combinations of machine learning techniques (including reinforcement learning, artificial neural networks, and fuzzy systems) aimed at contributing the dynamic self-management of virtual network resources, with each of the chapters building on the advantages of the previous one, while solving some drawbacks. **Chapter 7** proposes an automated multi-entity negotiation system and a dynamic pricing model both of which are aimed at achieving virtual network survivability. Finally, this thesis is concluded in **Chapter 8**, giving our outlook for future research directions in the area of resource management in network virtualisation.

[1]The full name of each code for the publications is given at the end of the thesis, where the complete list of publications is presented.

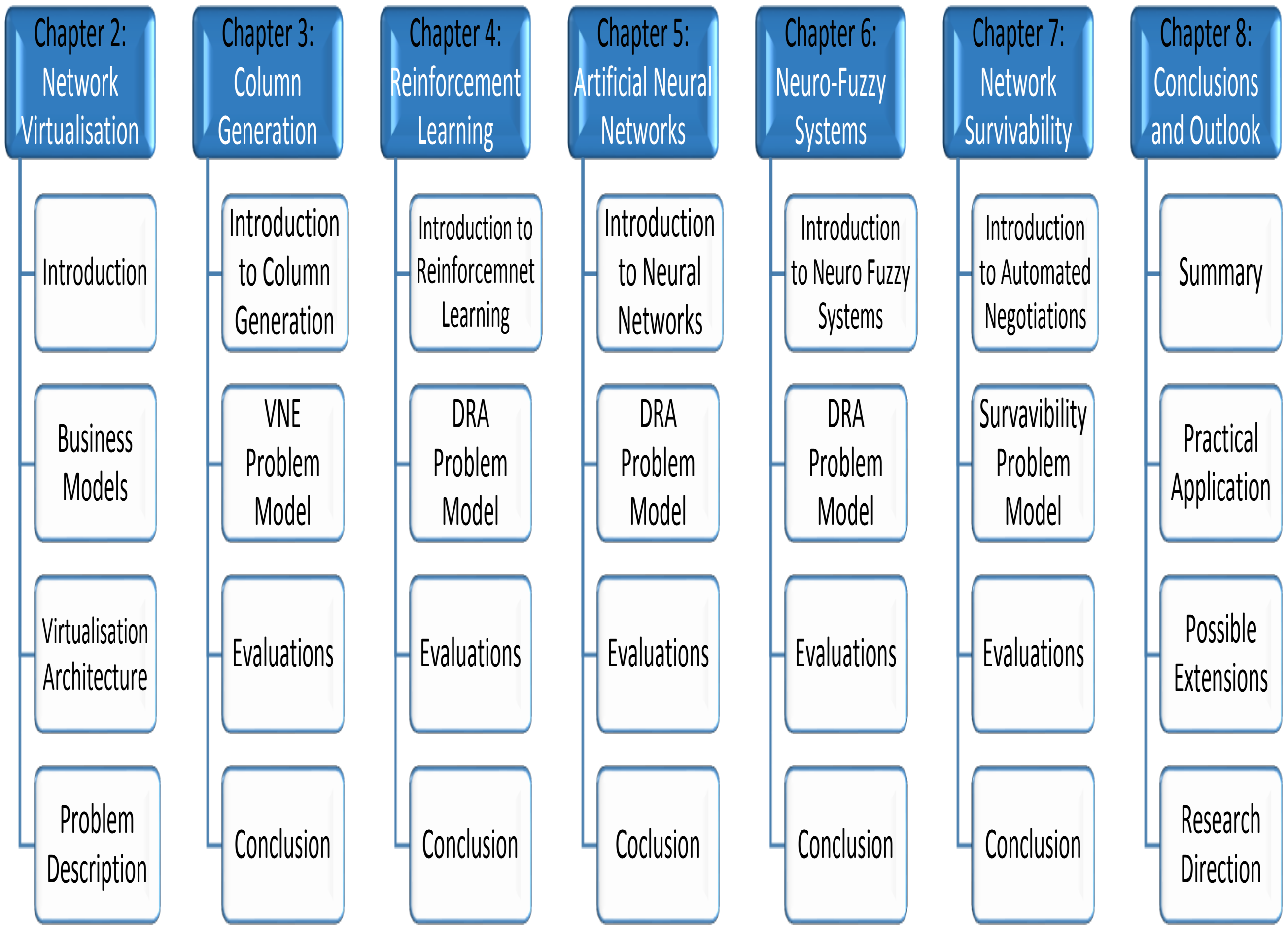

Chapter 2: Network Virtualisation
Introduction
Business Models
Virtualisation Architecture
Problem Description
Chapter 3: Column Generation
Introduction to Column Generation
VNE Problem Model
Evaluations
Conclusion
Chapter 4: Reinforcement Learning
Introduction to Reinforcemnet Learning
DRA Problem Model
Evaluations
Conclusion
Chapter 5: Artificial Neural Networks
Introduction to Neural Networks
DRA Problem Model
Evaluations
Coclusion
Chapter 6: Neuro-Fuzzy Systems
Introduction to Neuro Fuzzy Systems
DRA Problem Model
Evaluations
Conclusion
Chapter 7: Network Survivability
Introduction to Automated Negotiations
Survavibility Problem Model
Evaluations
Conclusion
Chapter 8: Conclusions and Outlook
Summary
Practical Application
Possible Extensions
Research Direction

Figure 1-5: Thesis Structure

Chapter 2

Network Virtualisation

2.1 Introduction

Network virtualisation is the process of combining hardware network resources and software network resources into a single administrative unit - a *virtual network* (VN). The goal of network virtualisation is to provide systems and users with efficient, controlled, and secure sharing of the networking resources [67]. It involves the separation of an infrastructure service from the physical resources on which the service runs. This service (such as node CPU, or link bandwidth) is not described on, identified by, or strictly associated to any physical asset. Instead, the service is described in a data structure, and exists entirely in a software abstraction layer reproducing the service on any physical resource running the virtualisation software [68]. Therefore, the life cycle, identity, location, and configuration attributes of the service exists in software with API interfaces, thereby unlocking the full potential of automated provisioning.

The main objective of network virtualisation is to achieve abstraction of network resources, in such a way that the resources can be shared with an aim of achieving efficiency [69]. Each VN in a network virtualisation environment (NVE) is made up of virtual nodes and virtual links, and each substrate network (SN) is made up of substrate nodes and substrate links. Each virtual node is hosted on a substrate node, and multiple virtual nodes can share one substrate node. In the same way, each virtual link is hosted over a substrate path (one or more substrate links), and any given substrate link can host one or more virtual links. Each virtual network constructs its own architecture and assigns its own protocol, which is customized according to its special services and user requirements [70]. Service providers can deploy and manage customised end-to-end services on those virtual networks for the end users by effectively sharing and utilising underlying network resources leased from multiple infrastructure providers [71].

The effect of network virtualisation on the Future Internet architecture has created a debate between architectural purists and pluralists [72]. While the purists propose that network virtualisation is a means of evaluating incremental changes to the architecture, the pluralists consider it as an important part of the architecture itself, so as not only to solve the ossification impasse, but also allows for reliable, flexible and dynamic management of the Internet. This chapter introduces the business model as well as architecture of network virtualisation based on the pluralists approach. The three sub-problems; virtual network embedding, dynamic resource allocation and virtual network survivability, considered in this thesis are also briefly described in this Chapter.

2.2 Network Virtualisation Business Models

Virtualisation is achieved by decoupling the roles of the traditional Internet service providers (ISPs) into two independent entities: infrastructure providers (InPs), who deploy and manage the physical network resources - also known as substrate networks (SNs), and service providers (SPs), who create virtual networks (VNs) by aggregating resources from one or more InPs and offer end-to-end network services [2].

The most basic business model in a network virtualisation environment consists of three players; an infrastructure provider, a service provider and end users. As shown in Fig. 2-1, the model involves a splitting of the role of the traditional Internet service provider [73] into two roles; a service provider and an infrastructure provider. It is worth noting that some models such as [71] include a brokerage role that can act as a mediator between any of the three roles shown in Fig. 2-1. In fact, [74] proposes two

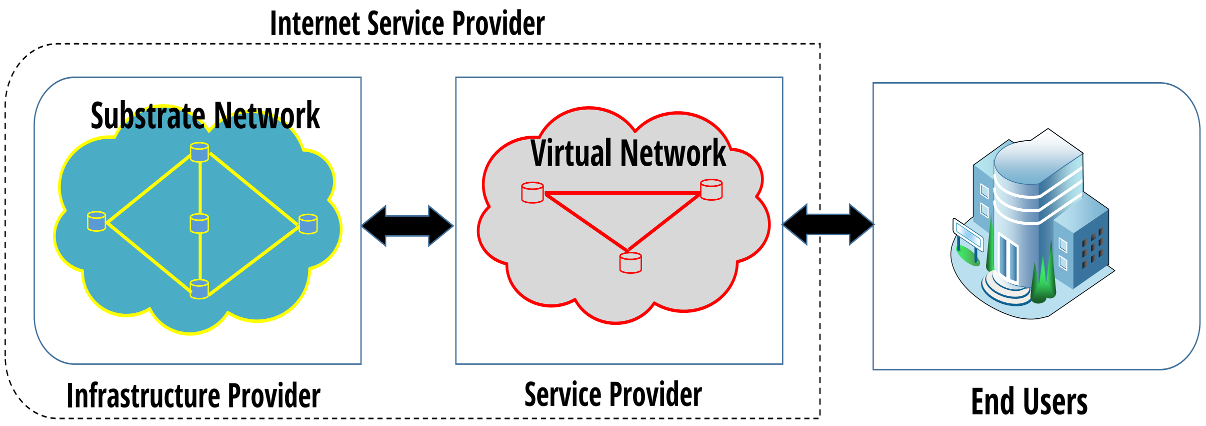


Figure 2-1: Network Virtualisation Business Roles and Players

additional roles - a virtual network operator (VNO) and a virtual network provider (VNP). The roles of the players in Fig. 2-1 are detailed in the rest of the section.

2.2.1 Infrastructure Provider

Infrastructure providers deploy and manage physical resources in form of substrate networks. These resources are then leased, through programmable interfaces, to one or more service providers. They also determine which resource requests (from service providers) are accepted, and how the physical resources are allocated to service providers. This way, InPs are able to influence the profitability resulting from their physical resources. If a given InP is not able to provide resources fully or in part to a given service provider, negotiations and hence coalitions can be formed with other InPs so as to provision multi-domain virtual networks [75].

2.2.2 Service Provider

Service Providers lease physical resources from one or more infrastructure providers, which they use to create virtual networks. They can then deploy customised protocol stacks onto these virtual networks and hence provide customised services to end users. In a more general case, service providers may also sub-lease the resources allocated to them to other service providers, in which case the former would appear as infrastructure providers.

2.2.3 End User

End users are the final consumers of the services provided by service providers. They are similar to the end users in the existing Internet, except that the existence of multiple virtual networks from competing service providers enables them to choose from a wide range of services [2]. End users may connect to multiple service providers for different services.

2.3 Network Virtualisation Architecture

Fig. 2-2 shows a conceptual representation of a network virtualisation environment architecture. The architecture is made up of two general entities; substrate networks and virtual networks. In the figure, three virtual networks are created on top of and share the physical resources of the substrate network. Each node in the virtual networks is hosted or mapped on a physical node, and each link in the virtual networks is mapped to one or more links in the physical networks. While in the figure we represent three virtual networks sharing resources from a single substrate network, it is in general possible that a given virtual network may use resources from more than one substrate network. Users of any of the virtual network should seamlessly connect via the substrate network to access the host resources, which could be in general the Internet for such applications as browsing and email, or even specialized services such as web servers, content servers or even storage databases. A given virtual network can also provide service to another virtual network, and generally a user can subscribe to more than one virtual network [2]. The substrate networks are made up substrate nodes and substrate links, while virtual networks are composed of virtual nodes and virtual links. In the following subsections, we define these components [11, 71, 76, 77]

2.3.1 Substrate Network

A substrate network is a combination of physical active and passive network elements (network nodes and network links). It consists of substrate nodes and substrate links

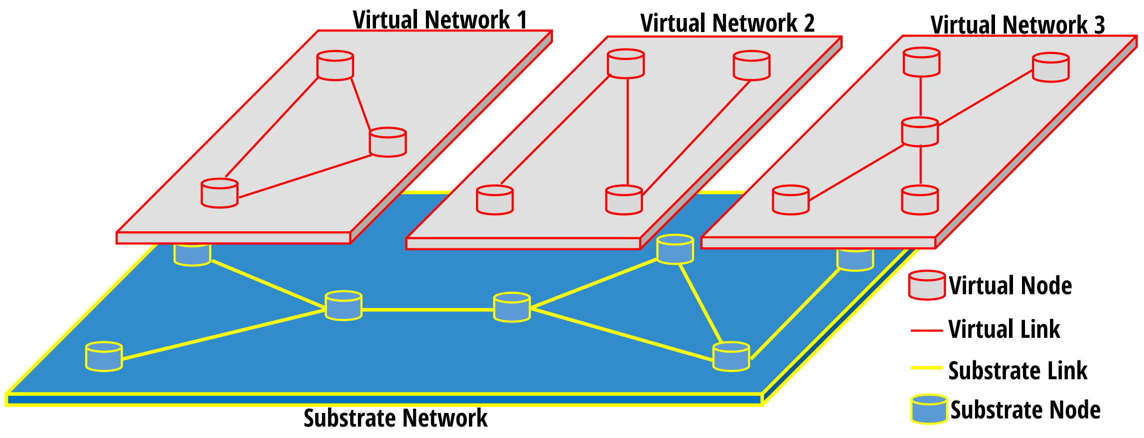


Figure 2-2: Network Virtualisation Architecture

that form a connected network topology of the infrastructure.

2.3.2 Substrate Nodes

A substrate node is a physical, active electronic device that is attached to a substrate network, and is capable of sending, receiving, or forwarding information over a communications channel. It is either a physical host or a physical router. A host acts as a packet source or a sink, while a router performs packet forwarding according to the protocols of the substrate network. A substrate node may host one or more virtual nodes from different virtual networks. Substrate nodes are usually defined with parameters such as location, CPU, queue size, e.t.c.

2.3.3 Substrate Links

A substrate link is a physical communications channel that connects two substrate nodes. Each substrate link can host one or more virtual links from one or more virtual networks. Each substrate link has a set of attributes that characterise it, for example, bandwidth (data rate), delay, packet loss. If the substrate network supports substrate path splitting [20] then a substrate link may only support a proportion of the total demand of any given virtual link.

2.3.4 Virtual Network

A virtual network is a combination of active and passive network elements (network nodes and network links) on top of a substrate network. It consists of virtual nodes and virtual links that form a connected network topology using the infrastructure of the underlying physical network. As shown in Fig. 2-2, several independent virtual networks can exist in parallel on top of a physical network.

2.3.5 Virtual Links

A virtual link is a logical interconnection of two virtual nodes, appearing to them as a direct physical link with dynamically changing properties. Each virtual link in the virtual network may span over one or more connected physical links i.e. a path in the underlying physical topology. In general, a virtual link may consist of multiple substrate paths, which can be used to increase the capacity or reliability (survivability) of the virtual link [26].

2.3.6 Virtual Nodes

A virtual node is a software component with either hosting or routing functionality, for example an operating system encapsulated in a virtual machine. Virtual nodes are interconnected through virtual links, forming a virtual network topology. In addition to having characteristics similar to those of substrate nodes, virtual nodes may have a restriction on how far from their required location they can be located [8].

2.4 Resource Management in NVEs

While many aspects of network virtualisation have received attention from the research community, a few remain unexplored till today, and many others, although touched, can be improved [2]. Resource management in network virtualisation environments (NVEs) is one of the areas that still require attention from the research community [2], as it affects the substrate resources utilisation efficiency, as well as

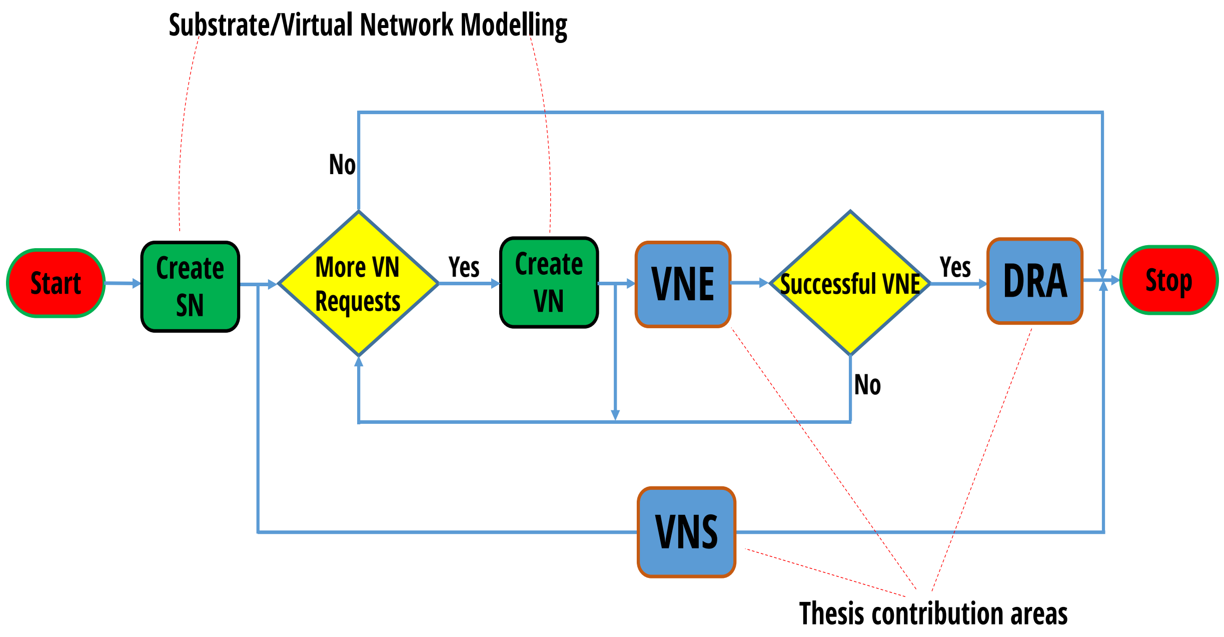


Figure 2-3: Resource Management in Network Virtualisation

quality of service guarantees both of which would directly affect the profitability of InPs and SPs, and hence the success of network virtualisation. As stated in Chapter 1 this thesis decomposes the resource management problem into three sub-problems; virtual network embedding (VNE), dynamic resource allocation (DRA), and virtual network survivability (VNS). While virtual network embedding is already well studied [11], this thesis proposes improvements to its solution compared to state of the art approaches. However, dynamic resource allocation and virtual network survivability are currently un-explored aspects of resource management in network virtualisation. The overall virtualisation resource management process flow considered in this thesis is shown in Fig. 2-3. It begins by describing the resource capacities of a substrate network, as well as the resource requirements and constraints of virtual networks. A virtual network embedding is then attempted. If the VNE is successful, then a dynamic resource allocation process is initiated for the embedded VN, and continues through out its lifetime. At the same time, and in parallel, a virtual network survivability process ensures that in case of substrate resource failures, quality of service guarantees for the virtual networks are not negatively affected. It is worth noting the while DRA manages resources through out the life of a given successfully embedded

VN, VNS continuously runs throughout the life of the substrate network. In the following subsections, the substrate and virtual network models used in this thesis, as well as the three contribution areas shown in Fig. 2-3 are introduced.

2.4.1 Virtual Network Modelling

The allocation of SN resources to a given VN is initiated by a SP specifying resource requirements for both virtual nodes and links to the InP. The specification of VN resource requirements is usually represented by a weighted undirected graph denoted by $G_v = (N_v, L_v)$, where N_v and L_v represent the sets of virtual nodes and links respectively. Each virtual link $l_{ij} \in L_v$ connecting the virtual nodes i and j has a maximum delay D_{ij} and bandwidth B_{ij}, while each virtual node $i \in N_v$ has a proposed location $L_i(x, y)$, a constraint on the maximum deviation $\Delta P_i(\Delta x, \Delta y)$ from its proposed location, which specifies the maximum allowed deviation for each of the x and y coordinates of node i, and a queue size Q_i, which is a measure of the maximum number of packets (or Bytes) a given node can have in its buffer before dropping packets.

It is worth noting that in general, the virtual nodes/links may have other constraints/requirements. For instance, ViNEYard [8] considers node CPU instead of node queue size. This thesis uses such node/link constraints interchangeably, and as dictated by a given sub-problem. As an example, for the VNE sub-problem, we use the node CPU instead of node queue size[1] since its the parameter that has been considered in most VNE proposals [11], while we use node queue size for DRA since in this sub-problem we are interested in evaluating the packet drop ratio.

2.4.2 Substrate Network Modelling

Similarly, a substrate network can be modelled as an undirected graph denoted by $G_s = (N_s, L_s)$, where N_s and L_s represent the sets of substrate nodes and links, respectively. Each substrate link $l_{uv} \in L_s$ connecting the substrate nodes u and v has

[1]This change is only in problem representation as one variable is replaced by another, and hence does not change the problem formulation or solution complexity.

a delay D_{uv} and a bandwidth B_{uv}, while each substrate node[2] $u \in N_s$ has queue size Q_u and a location $L_u(x, y)$.

2.4.3 Virtual Network Embedding

The VNE problem involves the mapping[3] of each virtual node $i \in N_v$ to one of the possible substrate nodes with in the set $\Upsilon(i)$. $\Upsilon(i)$ is defined as a set of all substrate nodes $u \in N_s$ that have enough *available* queue size and are located within the maximum allowed deviation $\Delta P_i(\Delta x, \Delta y)$ of the virtual node i. For a successful VNE, each virtual node must be mapped and any given substrate node can map at most one virtual node from the same request. Similarly, all the virtual links have to be mapped to one or more substrate links connecting the nodes to which the virtual nodes at its ends have been mapped. Each of the substrate links must have a sufficient data rate (bandwidth) to support the virtual link, e.g. a virtual link with a bandwidth requirement of 15Mbps cannot be mapped on a substrate path containing a substrate link whose available (unused) bandwidth is 14Mbps (unless the substrate network supports path splitting, and that the extra 1Mbps is carried by another parallel substrate path.). In addition, the total delay of all the substrate links used to map a given virtual link must not exceed the maximum delay specified by the virtual link. In Fig. 2-4, we show an example of two virtual networks being mapped onto a substrate network. The resource requirements for each virtual node or link is also shown. The values in the substrate network are the total loading of any given physical node or link. As can be noted from Fig. 2-4, one substrate node can host more than one virtual node (e.g. node A). A substrate link can also host more than one virtual link (e.g. link AB), and a given virtual link can span more than one substrate link (e.g link RP).

The VNE problem, with fixed constraints on virtual nodes and links, reduces to the multi-way separator problem which is known to be NP-Hard [8, 78]. Even when

[2]Throughout this thesis, while referring to nodes, the letters i and j are used to refer to virtual nodes while the letters u and v are used to represent substrate nodes.

[3]The terms mapping and embedding are used synonymously in this thesis.

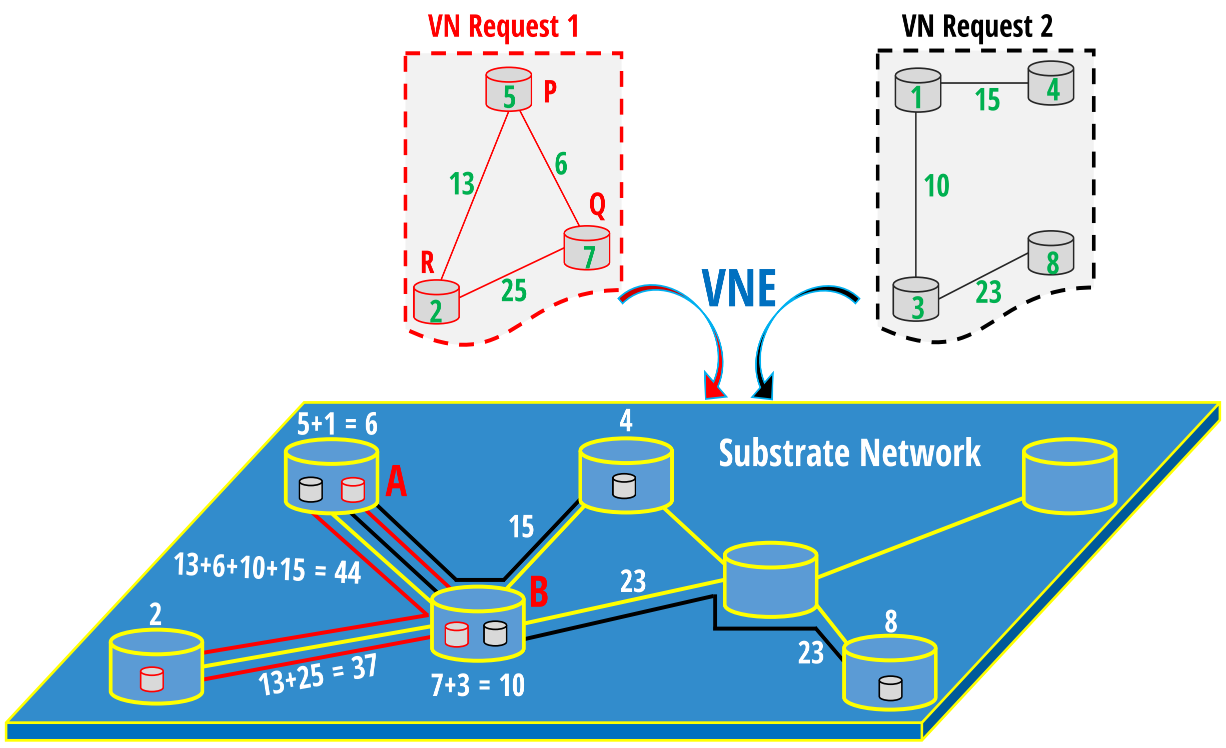


Figure 2-4: Virtual Network Embedding: Two VNs mapped onto a SN

all nodes have already been mapped, the problem of performing link mapping for unsplittable flows [79] is still NP-Hard [80]. Therefore, most approaches to the VNE problem have been through use of heuristics. In this context, and mainly aimed at simplifying the problem, several variations of the VNE problem can be defined. We give some of them below:

(a) Single Domain[21, 20, 8]: The virtual network is assumed to be completely mapped by a single substrate network,

(b) Multi-domain [75, 81] : A given VN is mapped across multiple substrate networks, owned by different InPs,

(c) Offline [21]: Assumes that all VN requests and demands are known in advance,

(d) Online [82]: VN requests arrive one at a time, and each is mapped without knowledge of future requests,

(e) Static [83]: Both VN and SN as well as initial mappings are not changed throughout the lifetime of the VN,

(f) Dynamic [20, 41, 84]: Considers changes in both VN and SN to either perform re-mappings or resource scaling,

(g) Two-stage [85, 86, 87]: Performs node mapping and link mapping in two disjoint steps,

(h) Coordinated [8, 88]: Performs node and link mapping in two coordinated steps,

(i) One-Shot [89, 90, 91, 92]: Performs both node and link mapping in one step,

(j) Splittable flows [8]: Assume that the substrate network supports the splitting of virtual network flows,

(k) Un splittable flows [20]: Considers that the total bandwidth of any given virtual link demand must be mapped wholly (or not at all) by any given substrate link.

Virtual network embedding problems that are formulated with any of the characteristics in (c, e, g, h, j and l) are generally much easier to solve, but are in general not realistic, for example, assuming that the substrate network has unlimited resources, or assuming that all virtual network requests are known a priori. In addition, performing the embedding in two separate steps can lead to blocking or rejecting of resource requests at the link mapping stage and hence a sub-optimal substrate resource utilisation. Even when the two embedding steps are coordinated, the embeddings are still sub-optimal. If the embedding is performed in one step, the embedding efficiency is significantly improved. However, the computational intractability prohibits finding optimal solutions even in this case. In Chapter 3, we propose a one-shot, unsplittable flow embedding approach that achieves a better resource utilisation while at the same time achieving a significant improvement in time complexity by use of column generation.

2.4.4 Dynamic Resource Allocation

Through the VN request, a service provider defines the required virtual node and link parameters such as node queue size and link bandwidth. However, as the resources allocated to nodes and links are meant for use by end users, and because Internet traffic is not uniform, reserving a fixed amount of resources for virtual nodes and links throughout their lifetime could lead to inefficient resource utilisation and hence limit the revenue of infrastructure providers, especially if other VN requests are rejected while reserving resources for VNs that are lightly loaded. Therefore, dynamic resource allocation as proposed in this thesis involves monitoring the actual usage of resources allocated to virtual networks, and making opportunistic use of these resources based on perceived need for them. The opportunistic use of resources involves carefully taking advantage of unused virtual node and link resources to ensure that VN requests are not rejected when resources reserved to already embedded requests are idle. This should however be performed carefully to ensure that quality of service parameters such as packet drop ratio and delay for the VNs are not affected. In Figs. 2-5 and 2-6 we illustrate the difference between a static resource allocation scheme, and a dynamic one. In the static resource allocation in Fig. 2-5 both the nodes (P and Q) and the link (PQ) keep their resource allocation at 100% of their initial demand irrespective of resource utilisation, while in the dynamic resource allocation, the resource allocation to each node and link is dynamically adapted to the perceived demand for these resources. Chapters 4, 5, 6 propose self-management approaches to dynamic resource allocation in virtual network environments.

2.4.5 Virtual Network Survivability

In practice, physical networks do not remain operational at all times [93], hence making the provisioning of resources for backups and/or restorations an inevitable part of any survivable network resource management approach. Survivability in network virtualisation [61] involves consideration that substrate links and nodes can fail, and in ensuring that the virtual nodes or links mapped onto the failed substrate resources are

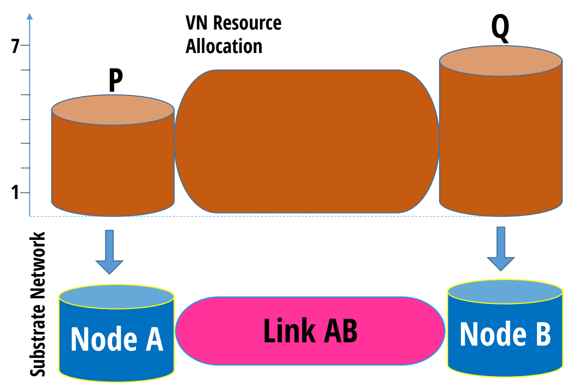


Figure 2-5: Static Resource Allocation

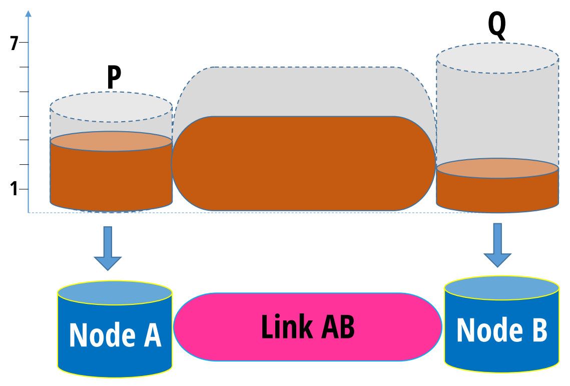


Figure 2-6: Dynamic Resource Allocation

not disrupted. This is usually achieved either by backing secondary resources (proactive survivable virtual network embedding) before failures have actually occurred or provisioning the resources upon substrate resource failures (reactive survivable virtual network embedding) [94]. While proactive virtual network embedding avoids the delays and possible data loss that may be encountered if resources have to be provisioned upon failures, reserving some physical resources for un foreseen failures could result into inefficient resource utilisation for the substrate network.

Since network link failures occur about 10 times more than node failures [95], and given that about 70% of unplanned link failures are single link failures [93], most approaches to survivability in virtualisation have concentrated on survivability for single substrate link failures [26, 61]. It should however be noted that any node failure can be considered as a failure of links adjacent to the node [26], and as such, proposals that consider single substrate link failures can be extended to cover multiple link failures, and hence node failures.

In Fig. 2-7, we show an example of a virtual network embedding that considers survivability of substrate links by provisioning backup substrate paths for each virtual link. In the figure, the virtual nodes P, Q and R are mapped onto substrate nodes D, E and B respectively. As an example, the virtual link RP has an original mapping (primary mapping) on substrate path BAD, and has a provisioned backup substrate path (secondary mapping) on substrate path BCD. Therefore, in case any of the

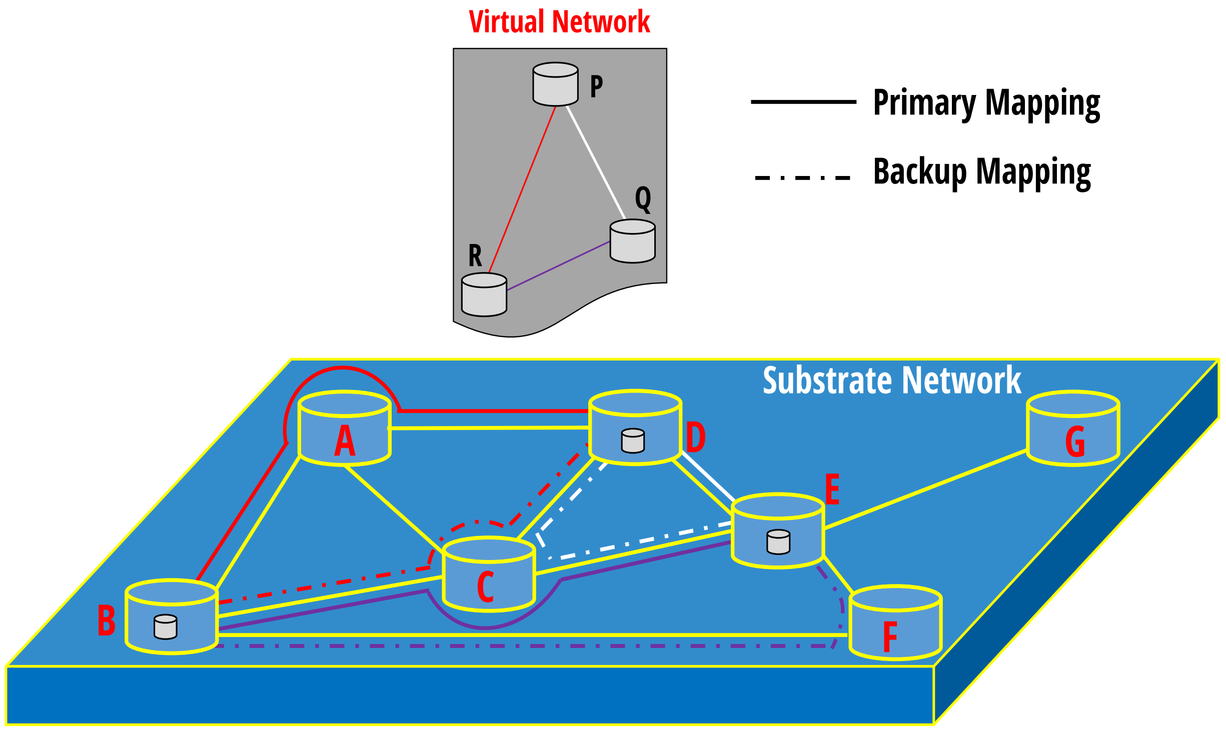


Figure 2-7: Survivable Virtual Network Embedding: Backup Link Provisioning

substrate links on the path BAD fails, the link RP will be migrated to BCD. As it can be noted, in order to ensure that the backup path is not affected by the failure, it is important to ensure that each of the substrate links on the backup path is disjoint of the primary substrate path. Chapter 7 of this thesis presents a proposal for considering survivability in multi-domain virtual networks which is based on automated negotiations.

2.5 Conclusion

This Chapter has introduced network virtualisation, its business model, and a network virtualisation architecture. We have also introduced the three major problems that are the subject of this thesis. In the next Chapter, the first of these sub-problems will be solved using column generation.

Chapter 3

Column Generation-based VNE

3.1 Introduction

The VNE problem can be formulated as a mathematical program [2]. In this case, the objective is usually to minimise or maximise a given aspect of the substrate to virtual network resource allocation, such as minimising substrate network load imbalances [2], maximising InP profits [96] and minimising energy losses [25, 97]. In this Chapter, we start by giving a brief introduction to both mathematical programming and column generation. We then formulate the one-shot unsplittable virtual network embedding problem based on substrate network paths rather than links, and thereafter propose a solution to it using column generation. The Chapter is concluded by evaluating our proposal as well as discussing the results.

3.2 Mathematical Programming

Mathematical programming, and especially linear programming, is one of the best developed and most used branches of operational research [98]. It involves the use of mathematical models, particularly optimizing models [99], to assist in taking decisions. The objective is usually to achieve optimum allocation of limited resources among competing activities, under a set of constraints imposed by the nature of the problem being studied. These constraints could reflect financial, technological, marketing, organisational, or many other considerations. In broad terms, mathematical programming can be defined as a mathematical representation aimed at programming or planning the best possible allocation of scarce resources. A mathematical program is made up of four main components [100]:

- **Variables** (also known as decision variables): These represent things that can be adjusted or controlled, for example the bandwidth of a substrate link, the CPU demand of a virtual node, e.t.c. Usually, the aim of a mathematical program is to find the values of the variables that provide the best value of the objective function.

- **Objective function**: This is a mathematical expression that combines the variables to express the goal of the mathematical program. It may represent a an average resource allocation or InP profit,. It is usually required to maximise or minimise this function.

- **Constraints**: These are mathematical expressions that combine the variables to express limits on the possible solutions. For example, they may express the fact that the maximum number of virtual links that can be mapped on a substrate link must not have total bandwidth demand that exceeds the total free bandwidth capacity of the substrate link.

- **Variable bounds**: Only rarely are variables in a mathematical problem permitted to take on any value from negative infinity to positive infinity. Instead, the variables usually have bounds, for example, 0 and 1 may bound a variable that indicates whether or not a given virtual node is mapped onto a substrate node.

When the mathematical representation uses linear functions exclusively (i.e. when all the mathematical expressions for the objective function and the constraints are linear), we have a linear-programming model [98]. In the next section, we formulate a linear program and use it to explain the concepts of duality and hence column generation.

3.3 Formulation of Linear Programs

A linear program is the most common formulation of an optimization/mathematiocal problem. It involves a *minimisation* or *maximisation* of an objective function over some domain. The objective function is linear, and the domain, or feasible set, is defined by linear constraints [101]. Equations (3.1) - (3.4) show a generic example of a linear program [102].

$$\textbf{minimise} \quad \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j t_j \tag{3.1}$$

$$\textbf{subject to} \quad e_j t_j + \sum_{i=1}^{m} a_{ij} x_i \geq g_j \quad , \quad 1 \leq j \leq n \tag{3.2}$$

$$f_i x_i + \sum_{j=1}^{n} b_{ij} t_j \geq h_i \quad , \quad 1 \leq i \leq m \tag{3.3}$$

$$x_i \geq 0, \ t_j \geq 0 \quad , \quad 1 \leq i \leq m, \ 1 \leq j \leq n \tag{3.4}$$

As can be observed in the formulation (3.1) - (3.4), the variables are x and t, (3.1) is the objective function, (3.2) and (3.3) are constraints and (3.4) are variable bounds. Therefore, the linear program above has $m + n$ constraints, and all its variables are positive.

3.3.1 Duality

Any given linear programming problem is referred to as a primal problem, and every primal problem has an associated linear program called the *dual problem* [101]. The dual problem provides an upper bound to the optimal value of the primal problem [14, 103]. The fundamental idea behind duality is that every feasible solution for the primal problem gives a bound on the optimal value of the objective function of the corresponding dual problem [104]. The duality theorem states that the objective function value of the dual at any feasible solution is always greater than or equal to the objective function value of the primal at any feasible solution [98]. In order to derive a dual program from the primal in (3.1) - (3.4), the following seven steps are used [101].

1. If necessary, rewrite the objective as a minimisation.

   Since our primal is already a minimisation problem, we skip this step

2. Rewrite each inequality constraint as a "*less than* or *equal*", and rearrange each constraint so that the right-hand side is 0.

   After this step, the linear program looks as shown in (3.5) - (3.7)

$$\textbf{minimise} \quad \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j t_j \tag{3.5}$$

$$\textbf{subject to} \quad g_j - e_j t_j - \sum_{i=1}^{m} a_{ij} x_i \leq 0 \quad , \quad 1 \leq j \leq n \tag{3.6}$$

$$h_i - f_i x_i - \sum_{j=1}^{n} b_{ij} t_j \leq 0 \quad , \quad 1 \leq i \leq m \tag{3.7}$$

3. Define a non-negative dual variable for each inequality constraint, and an unrestricted dual variable for each equality constraint.

   To constraints (3.2), *we associate n dual variables $y_j \geq 0$, and to constraints* (3.3), *we associate m dual variables $s_i \geq 0$.*

4. For each constraint, eliminate the constraint and add the term (*dual variable*)×(*left- hand side of constraint*) to the objective. Maximise the result over the dual variables.

$$\textbf{maximise} \quad \Big(\sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j t_j\Big) \tag{3.8}$$

$$y_j\Big(g_j - e_j t_j - \sum_{i=1}^{m} a_{ij} x_i\Big) \quad , \quad 1 \leq j \leq n \tag{3.9}$$

$$s_i\Big(h_i - f_i x_i - \sum_{j=1}^{n} b_{ij} t_j\Big) \quad , \quad 1 \leq i \leq m \tag{3.10}$$

5. We now have an objective with several terms of the form (dual variable)*(expression with primal variables), plus remaining terms involving only primal variables.

Rewrite the objective so that it consists of several terms of the form (*primal variable*)×(*expression with dual variables*), plus remaining terms involving only dual variables.

Doing this results into the objective in (3.11) - (3.13) *below*

$$\textbf{maximise} \quad \Big(\sum_{i=1}^{m} h_i s_i + \sum_{j=1}^{n} g_j y_j\Big) \tag{3.11}$$

$$t_j\Big(d_j - e_j y_j - \sum_{i=1}^{m} b_{ij} s_i\Big) \quad , \quad 1 \leq j \leq n \tag{3.12}$$

$$x_i\Big(c_i - f_i s_i - \sum_{j=1}^{n} a_{ij} y_j\Big) \quad , \quad 1 \leq i \leq m \tag{3.13}$$

6. Remove each term of the form (*primal variable*)×(*expression with dual variables*) and replace with a constraint of the form:

   - *expression* ≥ 0, if the primal variable is non-negative.
   - *expression* ≤ 0, if the primal variable is non-positive.
   - *expression* $= 0$, if the primal variable is unrestricted.

   Doing this results into the objective in (3.14) - (3.16) *below*

$$\textbf{maximise} \quad \Big(\sum_{i=1}^{m} h_i s_i + \sum_{j=1}^{n} g_j y_j\Big) \tag{3.14}$$

$$\Big(e_j y_j + \sum_{i=1}^{m} b_{ij} s_i \leq d_j\Big) \quad , \quad 1 \leq j \leq n \tag{3.15}$$

$$\Big(f_i s_i + \sum_{j=1}^{n} a_{ij} y_j \leq c_i\Big) \quad , \quad 1 \leq i \leq m \tag{3.16}$$

7. If the linear program in step 1 was rewritten as a minimisation, rewrite the result of the previous step as a minimisation; otherwise, do nothing.

Therefore, the final dual program is shown in (3.17) - (3.20).

$$\textbf{maximise} \quad \sum_{i=1}^{m} h_i s_i + \sum_{j=1}^{n} g_j y_j \tag{3.17}$$

$$\textbf{subject to} \quad \sum_{i=1}^{m} b_{ij} s_i + e_j y_j \leq d_j \quad , \quad 1 \leq j \leq n \tag{3.18}$$

$$f_i s_i + \sum_{j=1}^{n} a_{ij} y_j \leq c_i \quad , \quad 1 \leq i \leq m \tag{3.19}$$

$$y_j \geq 0, \ s_i \geq 0 \quad , \quad 1 \leq j \leq n, \ 1 \leq i \leq m \tag{3.20}$$

3.4 Column Generation

Many linear programs are too large to consider all their variables explicitly. Since most of the variables will be non-basic and assume a value of zero in the optimal solution, only a subset of variables need to be considered in theory when solving the problem [102]. Column generation takes advantage of this idea to generate only the variables which have the potential to improve the objective function, i.e. to find variables with negative reduced cost (assuming without loss of generality that the problem is a minimization problem). In order to use a column generation approach, the problem being solved is split into two problems: the primal problem and the dual problem. In Fig. 3-1, we represent the interaction between the primal and dual problems in column generation.

The main idea is to solve a restricted version of the program (the restricted primal problem) - which contains only a subset of the variables, and then (through the use of the dual problem) add more variables as needed [105]. Therefore, we start by solving a restricted primal problem, and from its solution, we are able to obtain dual prices for each of the constraints in the primal problem. This information is then utilised in the objective function of the dual problem. The dual problem is then solved. If the objective value of the dual problem is negative, a variable with negative reduced cost has been identified. This variable is then added to the primal problem, and the primal problem is solved again. Re-solving the primal problem generates a new set of

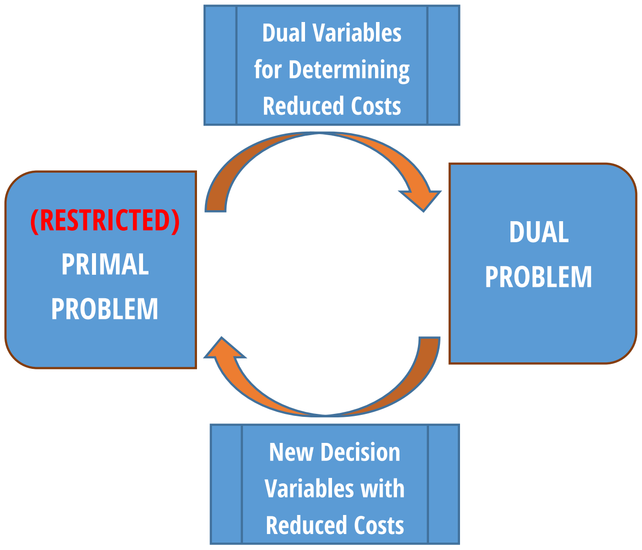

Dual Variables for Determining Reduced Costs
(RESTRICTED) PRIMAL PROBLEM
DUAL PROBLEM
New Decision Variables with Reduced Costs

Figure 3-1: Interaction between Primal and Dual Problems in Column Generation

dual values, and the process is repeated until no negative reduced cost variables are identified. When the dual problem returns a solution with non-negative reduced cost, we can conclude that the solution to the primal problem is optimal [106]. In order to have the initial restricted set of variables, it is required to have an initial *feasible* solution to the primal problem.

3.5 Column Generation-based VNE

In this section, we propose a column generation-based approach for a one-shot virtual network embedding. The one shot virtual network embedding problem involves performing both node and link mapping at the same time. We start by formulating the one shot VNE as a mathematical program, in which VNs arrive one a time (online) and hence the formulated optimisation problem involves the embedding of a single VN at any given time. The proposed VNE involves creation of paths in the substrate network. As a result, the variables considered in the mathematical program are substrate network paths, rather than individual links. For this reason, the column generation approach proposed involves generation of new paths. Therefore, in the rest of this thesis, the phrases *column generation* and *path generation* are used interchangeably.

3.5.1 Substrate Network Augmentation

We start by creating an augmented network [8], with each virtual node i connected to each of the substrate nodes in its possible node set $\Upsilon(i)$ by a *meta link* [8] $l_{iu} \in L_x$, where L_x is the set of all meta links. Then the aim is to establish a single path p_{uv}^{ij} from each virtual node i to all other virtual nodes j to which it is connected. The path p_{uv}^{ij} is made of two meta links, l_{iu} and l_{jv}, and a sub-path in the substrate network connecting the substrate nodes u and v. This sub-path may be made up of one or more substrate network links.

In Fig. 3-2, we show a representation of an instance of the problem. In the figure, XYZ are nodes of a virtual network, while ABCDEFG are nodes of a substrate network. As an example, for virtual link XZ, one possible path could be XABEZ, and is represented as p_{ae}^{xz}. The path p_{ae}^{xz} is a sequence of links in the augmented network that start from one end of the virtual link to the other. Therefore, in order to embed the virtual link XZ, we need to determine the three components of the path, which $-$ for this example $-$ are the two meta links XA and EZ, and the substrate network path ABE composed of two links, AB and BE. The components XA and EZ can be determined from a virtual to substrate node mapping, while ABE from a link mapping approach such as shortest path [107]. In particular, this path example would mean that the virtual node X is mapped onto substrate node A, the virtual node Z is mapped onto substrate node E and that the virtual link XZ is mapped onto the substrate network path ABE. One difficulty illustrated in this example comes from the fact that if, for example, we choose the path XABEZ for virtual link XZ, then the virtual link XY can only be mapped on a path that includes meta link XA and not XC. This would in turn require that Y be mapped onto C, otherwise we would have a suboptimal solution in which the virtual link XY uses resources from two substrate links (AC & CG) instead of a single link (CG). Hence, the determination of these paths should not be carried sequentially and independently. As previously mentioned, our aim is to find the best possible path for each of the virtual links subject to the

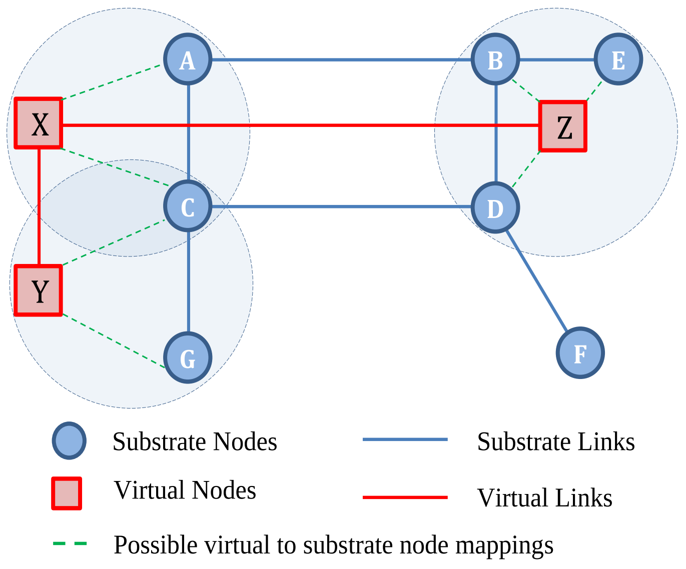


Figure 3-2: Embedding a virtual network onto a substrate network

mapping requirements described in 2.4.3

3.5.2 LP−P: Path based Formulation −*Primal*

We formulate the virtual network embedding problem as a commodity flow problem[27], where virtual links are flows that should be carried by the substrate network. However, unlike most commodity flow formulations, in our case, the source node i and terminal node j for each flow also need to be determined.

Variable and Parameter definitions: In this formulation, we define a nonnegative binary variable $f_{uv}^{ij} = [0, D_{ij}]$ which represents the unsplittable flow of a virtual link $l_{ij} \in L_v$ on a simple substrate path $p_{uv}^{ij} \in P$. The indices u, v, i and j define a path $(i - u - v - j)$ in the augmented substrate network. As described in 3.5.1, these paths are made up of three components: two meta-links iu and jv, and a substrate network path from u to v. The variable f_{uv}^{ij} is binary in that it can only take on values 0 and D_{ij}, where D_{ij} is the demand of virtual link $l_{ij} \in L_v$. We define P as a set of all the possible substrate paths, P_{uv} as the set of all paths that use the substrate link $l_{uv} \in L_s$ and P^{ij} as the set of all paths that can support the flow for virtual link $l_{ij} \in L_v$. We also define $\chi_u^i = [0,1]$ as a binary variable equal to 1 if the virtual node i is mapped onto the substrate node u and 0 otherwise. As mentioned in 3.5.1, it is important to note that variables χ_u^i and χ_v^j directly determine the existence or otherwise of meta links iu and jv for the path p_{uv}^{ij} since the meta links are dependent on the respective node mappings. For example, if $\chi_u^i == 0$ then the virtual node i is not mapped onto substrate node u, implying that the meta link from i to u is non existent, and so is the path p_{uv}^{ij}. Let A_{uv} be the available bandwidth capacity on the substrate link l_{uv}, and A_u be the available computation capacity on node u.

Objective: The objective of the mathematical formulation (3.21)−(3.29) is to balance the resource usage of the substrate network, by favouring the selection of those resources with comparatively higher available capacity. Balancing the loading of the substrate network has two advantages; first, it distributes the mapping of a given VN request over multiple substrate network resources which avoids a single VN being majorly affected by single or regional failures in the substrate network, hence ensuring better VN survivability. In addition, since the problem we consider in this thesis is online, we do not know in advance the required node locations for future VN requests. Balancing the loading of the substrate network ensures that at any given point, each substrate node/link has the same utilised capacity on average. This avoids situations where a VN request would be rejected due to one or more of its nodes not being able to be mapped because substrate nodes in their respective possible node sets $\Upsilon(i)$ have less resources than other parts of the substrate network.

$$\textbf{Minimise} \sum_{l_{ij}\in L_v} \sum_{p_{uv}^{ij}\in P} \frac{1}{A_{uv}} f_{uv}^{ij} + \sum_{i\in N_v} \sum_{u\in\Upsilon(i)} \frac{1}{A_u} \chi_u^i \tag{3.21}$$

subject to

$$\sum_{u\in\Upsilon(i)} \chi_u^i = 1 \qquad \forall i \in N_v \tag{3.22}$$

$$\sum_{i\in N_v} \chi_u^i \leq 1 \qquad \forall u \in N_s \tag{3.23}$$

$$\sum_{p_{uv}^{ij} \in P^{ij}} f_{uv}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v \tag{3.24}$$

$$\sum_{p_{uv}^{ij} \in P_{uv}} f_{uv}^{ij} \leq A_{uv} \qquad \forall l_{uv} \in L_s \tag{3.25}$$

$$f_{uv}^{ij} - D_{ij}\chi_u^i \leq 0 \qquad \forall p_{uv}^{ij} \in P \tag{3.26}$$

$$f_{uv}^{ij} - D_{ij}\chi_v^j \leq 0 \qquad \forall p_{uv}^{ij} \in P \tag{3.27}$$

$$f_{uv}^{ij} = [0, D_{ij}] \qquad \forall p_{uv}^{ij} \in P \tag{3.28}$$

$$\chi_u^i = [0,1] \qquad \forall i \in N_v, \forall u \in N_s \tag{3.29}$$

The first term in the objective (3.21) is for link mapping, while the second term is for node mapping. Each of these terms are divided by the respective capacities to ensure that the substrate resources with more free resources are preferred. Constraint (3.22) ensures that each virtual node is mapped to a substrate node, while (3.23) ensures that any substrate node may be used at most once for a given mapping request. Constraints (3.24) and (3.25) represent the virtual link demand requirements and substrate link capacity constraints respectively. Specifically, (3.24) states that the flow f_{uv}^{ij} on path p_{uv}^{ij} should carry the total demand of the virtual link ij, while (3.25) states that the flow f_{uv}^{ij} on path p_{uv}^{ij} should be atmost equal to the capacity of each substrate link on that path. From constraint (3.26), if $\chi_u^i == 0$ then $f_{uv}^{ij} = 0$. If $\chi_u^i == 1$ then $f_{uv}^{ij} = [0, D_{ij}]$. This is also true for (3.27). These constraints ensure that virtual links and virtual nodes are mapped at the same time, i.e., a flow f_{uv}^{ij} – using the path p_{uv}^{ij} starting with meta link iu and ending with meta link jv – is only non-zero if the virtual node i is mapped onto substrate node u and j is mapped onto v. Together, (3.26) and (3.27) ensure that a flow f_{uv}^{ij} is only non zero if both the two end links iu AND jv exist. Finally, (3.28) and (3.29) are the variable domain bounds.

The formulation in (3.21)−(3.29) is intractable for two reasons; first, the restrictions that variables χ_u^i and f_{uv}^{ij} only take on binary values ((3.28) and (3.29)), and then the fact that the number of possible paths p_{uv}^{ij} (and hence the number of variables f_{uv}^{ij}) is very large (exponential) even for moderately sized networks. Therefore, solving the problem in its current form is impractical. There are three possibilities to solving the problem;

1. a relaxation to the constraints on variables χ_u^i and f_{uv}^{ij} to take on continuous values,
2. restricting the number of input variables f_{uv}^{ij} (by restricting the number of paths p_{uv}^{ij}).
3. a combination of both the first two approaches.

For the VNE problem as formulated in (3.21)−(3.29), a relaxation would require careful consideration to avoid violating the requirements that both nodes and links are mapped in one shot (since the variables χ_u^i would no longer be able to restrict the mapping of virtual nodes to particular substrate nodes), as well splitting the flows of the virtual links across multiple links. Therefore, we take the second approach, and employ path generation, which allows for the use of only a sufficiently meaningful number of paths, and adding more paths as needed until a final solution is obtained.

3.6 Proposed Path Generation Approach

The path generation approach taken in this thesis is shown in Fig 3-3. We start by creating an initial set of paths (P_1) using a two stage node and link mapping. We then use these paths to solve a dual problem, and use the pricing problems (shortest path problems) to determine a set of paths (P_2) to add to the initial solution i.e. paths that can improve the initial solution. These paths are then used to solve a restricted primal problem to obtain a final solution. It can be noted that our proposal avoids the usual iteration required in a path generation approach where the primal and

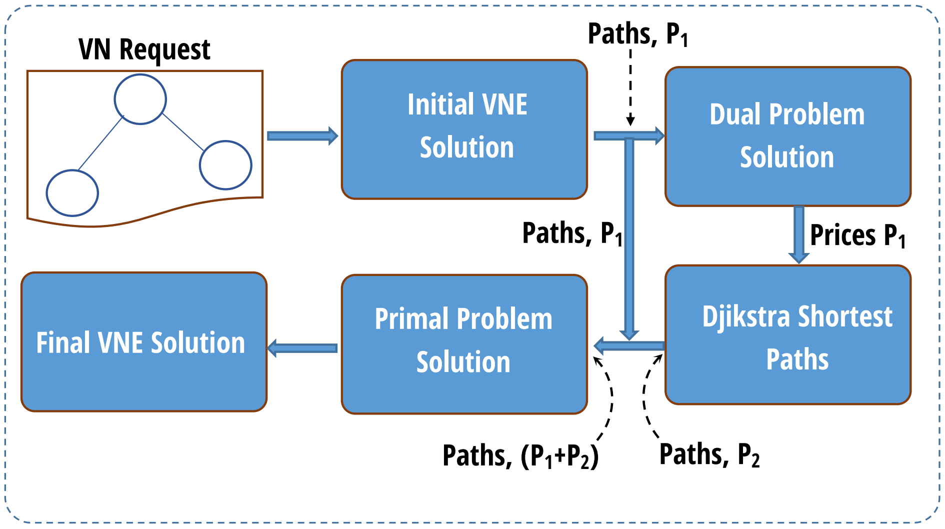


Figure 3-3: Path Generation-based Virtual Network Embedding

dual problems are solved sequentially, many times, instead preferring only to perform a single iteration. In the next subsections, we propose a method for determining the initial set of paths, derive the pricing problems, and then describe the overall algorithm proposed in this thesis.

3.6.1 Initial Solution

An initial solution (Init−Sol) is determined as a set of paths P_1 in the augmented substrate network, with each path $p_{uv}^{ij} \in P_1$ able to support the flow f_{uv}^{ij} of virtual link l_{ij}. Each of these paths must be able to meet the virtual network mapping conditions as formulated in the primal problem ((3.22) - (3.29)) . Revisiting the example in Fig. 3-2, since we have two virtual links, an initial solution for such a case would have two paths, one for each virtual link. Examples of these paths could be XABEZ and XACY for virtual links XZ and XY respectively. In order to determine such a path, say for virtual link XZ, the approach proposed is as follows: we start by performing a node mapping, which for this example, would map virtual nodes X and Z onto substrate nodes A and E respectively. This step gives us the meta links XA and EZ. In this subsection, we propose a novel node mapping solution LP−N for determining XA and EZ. The next step involves determining the path ABE in the substrate network.

This is done by using *Dijkstra's algorithm*[108], with the constraint that each link on the path should have enough capacity to support the virtual link under consideration. The complete path is determined by joining meta links XA and EZ to the respective ends of ABE.

LP−N: Node Mapping

LP−N is based on mathematical programming and is formulated in such a way that mapping of any given virtual node is biased towards those substrate nodes in its set of possible nodes $\Upsilon(i)$, which has a high weighted average available capacity of the connected links.

Objective: Two objectives are considered in this formulation: The first is to keep the computation time of the solution as low as possible by including only the possible virtual node to substrate node combinations. Secondly, we minimize the possibility of failure at the link mapping stage, by making the node mapping *aware* of the link mapping stage through the use of weights W_i and W_u.

Variable definition: As defined before ((3.21) - (3.29)), χ_u^i is a binary variable equal to 1 when the virtual node i is mapped onto substrate node u and 0 otherwise.

$$\textbf{minimise} \sum_{i \in N_v} \sum_{u \in \Upsilon(i)} \frac{W_i}{W_u} \chi_u^i \tag{3.30}$$

subject to:

$$\sum_{u \in \Upsilon(i)} \chi_u^i = 1 \quad \forall i \in N_v \tag{3.31}$$

$$\sum_{i \in N_v} \chi_u^i \leq 1 \quad \forall u \in N_s \tag{3.32}$$

$$\chi_u^i = [0,1] \qquad \forall i \in N_v, \forall u \in N_s \tag{3.33}$$

Constraints (3.31), (3.32) and (3.33) are similar to (3.22), (3.23) and (3.29) respectively. The weights W_i and W_u are determined for each virtual and substrate node respectively. W_u is weighted average of the available capacities of all the substrate links connected to u. Similarly, W_i is weighted average of the demand of all the virtual links connected to i. To illustrate the idea behind these weighted averages, consider Fig. 3-4, which is a subset of the topology represented in Fig. 3-2. The values beside each link represent the available link bandwidths (blue in Fig. 3-2) and link demands (red in Fig. 3-2) for substrate and virtual links respectively. As an example, considering the virtual node X,

$$W_X = 20 \times \left(\frac{20}{20+10}\right) + 10 \times \left(\frac{10}{20+10}\right) = 16.67.$$

In the same way for substrate node C,

$$W_C = 50 \times \left(\frac{50}{50+60+70}\right) + 60 \times \left(\frac{60}{50+60+70}\right) + 70 \times \left(\frac{70}{50+60+70}\right) = 61.11$$

The reason for using this ratio as a weight is to ensure that those substrate nodes that are connected to many substrate links with higher available resources are usually preferred, and that in case two or more virtual nodes have a given substrate node in their possible node set (such as X and Y in Fig. 3-2), then the substrate node would always be allocated to that virtual node with the highest weighted average link demand. This achieves some level of coordination between the node mapping and link mapping phases and thereby reduces the probability of rejecting link mapping requests.

We note that there could be instances where the weighted averages lead to selecting substrate nodes with less good links, especially when the links have widely differing residual capacities. For example, a node connected to two links with residual capacities 80 and 10 respectively will have a $W_1 = 72$, while a node connected to two

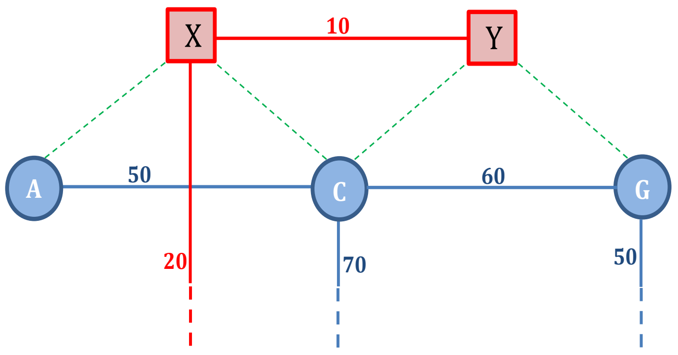


Figure 3-4: Node-Link Weighted Averages

links with residual capacities 60 and 70 respectively will have a $W_2 = 65$. In this case, the first node will be selected yet the second node *could* be a better choice. One simple solution to handle such scenario is to use the sum of two averages: the weighted average and a simple average. However, it is worth mentioning that in our approach network embedding is done in such a way that the average loads of substrate network nodes and links are balanced, this way, avoiding scenarios where some node and/or links have widely differing residual capacity. The procedure, **Init-Sol**, for determining the initial solution is shown in Algorithm 1.

3.6.2 Pricing Problem

To determine which paths should be added to the initial set so as to improve the solution, we need to solve the pricing problems for LP$-$P. In order to identify the pricing problems we first formulate the dual problem LP$-$D for the primal problem LP$-$P.

Dual Variables definitions: For a primal that minimises an objective, deriving the corresponding dual involves maximising an objective, and defining a non-negative dual variable for each inequality constraint, and an unrestricted dual variable for each equality constraint. Therefore, we define six dual variables as follows: λ_i for the virtual node constraints (3.22), μ_{ij} for the virtual links demand constraints in (3.24), $\eta_u >= 0$ substrate node constraints in (3.23), $\gamma_{uv} >= 0$ substrate links available

Algorithm 1 Init−Sol $(G_v(N_v, L_v), G_s(N_s, L_s))$

```
1: for $i \in N_v$ do
2:     $Determine\, Candidate\, Node\, Set,\ \Upsilon(i)$
3:     if $\Upsilon(i) = \emptyset$ then
4:         $Reject\ Request$
5:         end
6:     end if
7:     $Calculate\ W_i$
8: end for
9: for $u \in N_s$ do
10:    $Calculate\ W_u$
11: end for
12: $Solve$ : LP-N
13: for $l_{ij} \in L_v$ do
14:    for $u \in \Upsilon(i)$ do
15:        if $\chi_u^i = 1$ then
16:            Meta Link 1: $l_1 = iu$
17:            Start Node, $s = u$
18:        end if
19:    end for
20:    for $v \in \Upsilon(j)$ do
21:        if $\chi_v^j = 1$ then
22:            Meta Link 2: $l_2 = jv$
23:            End Node, $t = v$
24:        end if
25:    end for
26:    LinkMapping: $p_s = Dijkstra\Big(s, t, G_s(N_s, L_s)\Big)$
27:    Create Path: $p_{uv}^{ij} = l_1 + p_s + l_2$
28:    Add $p_{uv}^{ij}$ to $P'$
29: end for
```

capacity constraints in (3.25), $\sigma_{iu} >= 0$ for simultaneous node and link mapping constraint constraint (3.26) and $\tau_{jv} >= 0$ for constraint (3.27). Since most results of duality for linear programs do extend to integer programming[109], the following dual formulation is based on the steps described in Section 3.3.1.

The **objective** of the dual formulation (3.34)−(3.36) is to obtain a mathematical program that produces a maximised value as close as possible to that of its original primal program for any instance of the variables. Therefore, the dual of the primal formulation in (3.21)−(3.27) is:

$$\textbf{maximise} \sum_{i \in N_v} \lambda_i + \sum_{l_{ij} \in L_v} D_{ij}\mu_{ij} - \sum_{u \in N_s} \eta_u - \sum_{l_{uv} \in L_s} A_{uv}\gamma_{uv} \tag{3.34}$$

subject to

$$\lambda_i + \sum_{p_{uv}^{ij} \in P} (\sigma_{iu} + \tau_{jv}) - \eta_u \leq \frac{1}{A_u} \qquad \forall l_{iu} \in L_x \tag{3.35}$$

$$\mu_{ij} - \sigma_{iu} - \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} - \tau_{jv} \leq \sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} \qquad \forall p_{uv}^{ij} \in P \tag{3.36}$$

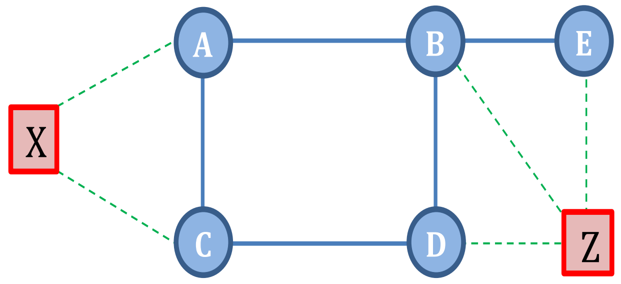
The pricing problems are shown in (3.35) and (3.36). From (3.35), the pricing condition for substrate nodes can be determined as:

$$\lambda_i + \sum_{p_{uv}^{ij} \in P} (\sigma_{iu} + \tau_{jv}) > \frac{1}{A_u} + \eta_u$$

However, since the variables χ_u^i are much fewer compared to f_{uv}^{ij}, we include all the possible substrate nodes for each virtual node in the restricted primal problem. This eliminates the need for node pricing and we are left to deal with only the link pricing problem (3.36):

$$\mu_{ij} > \sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} + (\sigma_{iu} + \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} + \tau_{jv})$$

This pricing problem can be solved using the shortest path algorithm. Any path $p_{uv}^{ij} = S_{iu} + (l_{uv} \in P_{uv}) + T_{jv}$ in the augmented substrate network whose length with respect to the dual variables (this means that the costs of the substrate links $l_{uv} \in P_{uv}$ are γ_{uv}, those of meta links S_{iu} are σ_{iu} and those of T_{jv} are τ_{jv}) is smaller than μ_{ij} satisfies the inequality above, and *has the potential* to improve the solution. However, a change in path for any given virtual link could necessitate a change in the mapping of one of its end nodes, which would change the prices and feasibility of mappings for other virtual links connected to it. For example, in Fig. 3-2, if the virtual node X is



(A←X, B←Z), (A←X, E←Z), (A←X, D←Z)

(C←X, B←Z), (C←X, E←Z), (C←X, D←Z)

We use A←X to mean that node virtual node X is mapped onto substrate node A

Figure 3-5: Possible substrate node combinations for virtual link XZ

mapped onto substrate node C, all the paths for both links XZ and XY go through C. If the path for say XZ is changed to go through A, it would either mean that the path for XY should also be changed to go through A, otherwise this path cannot be used to give a feasible and improved solution. Therefore, addition of paths individually for each virtual link does not guarantee that each of the added paths would still lead to a feasible solution, and for as long as the added path cannot yield a *feasible* solution, this path cannot lead to improvement in the solution of the restricted primal problem. In this case, there would be no guarantee that the pricing problems can be solved in polynomial time, as it could require quite a number of iterations before enough paths are added to actually improve the solution.

In our proposal, instead of adding individual paths for each virtual link in each iteration of the path generation algorithm, we include all the possible shortest path combinations after solving the formulation (3.34) – (3.36). We use Fig. 3-5, which is extracted from Fig. 3-2, to illustrate this for the case of virtual link XZ. Since the node X has two possible substrate nodes and virtual node Z has three possible substrate nodes, then the possible combinations for these nodes are 6. In our pricing

Algorithm 2 Final−Sol($G_v(N_v, L_v), G_s(N_s, L_s)$)

```
 1: Create Augmented Substrate Network
 2: Initial Paths Set: $P_1 \leftarrow$ *Solve* **Init−Sol**
 3: *Solve* **LP−D**($P_1$)
 4: for $l_{ij} \in L_v$ do
 5:    for $u \in \Upsilon(i)$ do
 6:       for $v \in \Upsilon(j)$ do
 7:          $P_2 = GetShortestPath(i, u, v, j)$
 8:          $P = (P_1 + P_2)$
 9:       end for
10:    end for
11: end for
12: *Solve* **LP−P**($P$)
```

solution, we determine the shortest path − based on the weights

$$\sum_{l_{uv}, l_{iu} \in p_{uv}^{ij}} \frac{1}{A_{uv}} + \left(\sigma_{iu} + \sum_{l_{uv} \in p_{uv}^{ij}} \gamma_{uv} + \tau_{jv} \right)$$

for each of these 6 possible end node combinations. This is done for all the virtual links, and all the corresponding paths are added to the restricted primal problem. However, the number of paths added for each pricing iteration would be too big to handle if many iterations are carried out. Even the Dijkstra algorithm takes quite some time to find the shortest paths. For this reason, we perform only one round for the substrate paths and use the resulting shortest paths based on the dual problem to solve LP−P to obtain the final solution. As we show in the simulation results, the solution obtained is near optimal. The procedure, **Final−Sol**, for determining the final solution is shown in Algorithm 2.

3.7 Performance Evaluation

3.7.1 Simulation Setup

To evaluate the performance of our proposed approach, we implemented a discrete event simulator in Java, which uses the tool Brite [110] to generate substrate and

Table 3.1: Brite Network Topology Generation Parameters

Parameter	**Substrate Network**	**Virtual Network**
Name (Model)	Router Waxman	Router Waxman
Number of nodes (N)	100 and 20	[15-25] and [3-10]
Size of main plane (HS)	500	500
Size of inner plane (LS)	500	500
Node Placement	Random	Random
GrowthType	Incremental	Incremental
Neighbouring Nodes	3	2
alpha (Waxman Parameter)	0.15	0.15
beta (Waxman Parameter)	0.2	0.2
BWDist	Uniform	Uniform

virtual network topologies. We used the tool ILOG CPLEX 12.4 [111] to solve the mathematical programs. Simulations were run on Windows 8 Pro running on a 4.00GB RAM, 3.00GHz Processor Machine. Both substrate and virtual networks were generated on a 500×500 grid. The CPU and bandwidth capacities of substrate nodes and links are uniformly distributed between 50 and 100 units respectively. The CPU demand for virtual network nodes is uniformly distributed between 2 and 10 units while the bandwidth demand of the links is uniformly distributed between 10 and 20 units. The parameters used in Brite to generate network topologies are shown in Table I. The parameters α and β are Waxman-specific exponents, such that, $0 < \alpha \leq 1, 0 < \beta \leq 1, (\alpha, \beta) \in \mathbb{R}$. α represents the maximal link probability while β is used to control the length of the edges. High values of alpha lead to graphs with higher edge densities while high values of beta lead to a higher ratio of long edges to short ones. The values used in this thesis are the default values in the brite router Waxman model used in [110]. Each virtual node is allowed to be located within a uniformly distributed distance between 100 and 150 units of its requested location. For embedding quality evaluations, two possible sets of network sizes have been used.

Table 3.2: Performance Quality Evaluation Algorithms

Code	**Mapping Method**
GNMSP	Greedy Node Mapping and Shortest Path for links[20]
CNMMCF	Coordinated Node and MCF for Link Mapping[8]
VNA-1	One Short Node Mapping[112]
PaGeViNE	Path Generation based Virtual Network Embedding
ViNE-OPT	Link based Optimal Virtual Network Embedding

One involves a substrate network with 100 nodes and virtual networks with number of nodes varied uniformly between 15 and 25, while the other has a substrate network with 20 nodes and virtual networks with number of nodes varied uniformly between 3 and 10. The need for different network sizes will be explained in a later subsection. For these simulations, we assumed Poisson arrivals at an average rate of 1 per 3 time units. The average service time of the requests is 60 time units and assumed to follow a negative exponential distribution. The experiments are performed for 1500 arrivals. For the time complexity evaluation, the number of nodes for the substrate network is gradually increased from 20 to 100.

3.7.2 Performance Metrics

Solution Quality

Three performance indicators − Acceptance ratio, Node utilization and Link utilization − are used for quality evaluation. The acceptance ratio gives a measure of the number of virtual network requests accepted compared to the total requests. We define the average node utilization as the average proportion of the total substrate node capacity that is under use at any given time. In the same way, we define average link utilization as the average proportion of the total substrate link capacity that is under use at any given time.

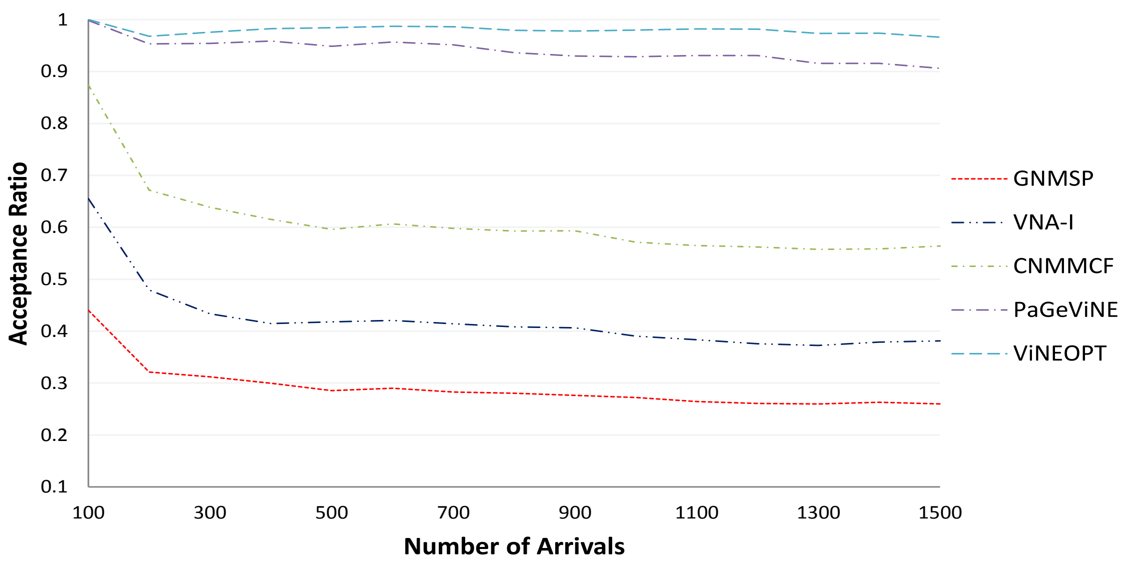


Figure 3-6: Average Acceptance Ratio - 20 SN Nodes

Solution Complexity

We define the time complexity of a given solution as the average time to complete the computation.

3.7.3 Comparisons

We compare the performance of our solution with closely related solutions. In particular, three representative solutions from the literature are chosen. The first performs node and link mapping separately [20]; the second coordinates the two steps [8]; and the third performs a one shot mapping [112]. These solutions were slightly modified to fit into our formulation of the problem. Specifically, unsplittable flows, constraints on substrate network capacities and constraints on virtual node locations were applied. We also implemented a baseline link-based formulation of the optimal one shot mapping (see Appendix C). We identify and name the compared solutions in table 6.4.

Since ViNEOPT requires a very long time (in excess of 1 hour for a single embedding involving a substrate network of 60 nodes and a virtual network of 10 nodes) to perform an embedding, simulations evaluating this algorithm have been restricted

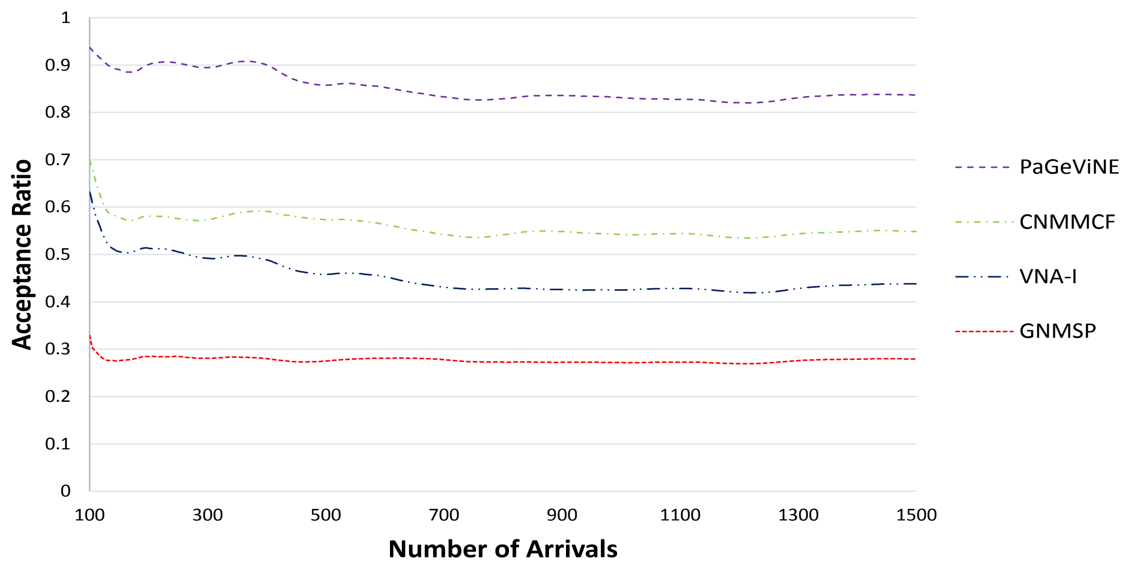


Figure 3-7: Average Acceptance Ratio - 100 SN Nodes

to substrate networks with 20 nodes and virtual networks with nodes from $3 - 10$. However, an extra simulation for acceptance ratio using larger sized networks has be performed so as to reflect more practical network sizes. This simulation excludes ViNEOPT.

3.7.4 Results

From the graphs in Fig. 3-6 it is evident that PaGeViNE achieves an average acceptance ratio close to that obtained by the optimal solution ViNE-OPT. In addition Fig. 3-6 and Fig. 3-7 show that PaGeViNE outperforms state-of-the-art solutions in terms of average acceptance ratio. These two figures also confirm that the embedding efficiency of PaGeViNE is not affected by increasing the size of substrate and virtual networks. It is also evident from the graphs in figures 3-8 and 3-9 that PaGeViNE achieves a better utilization ratio for substrate node and link resources compared to other solutions. However, we note that CNMMCF has a link utilization ratio that is comparatively close to that of PaGeViNE. The fact that CNMMCF is underperforming PaGeViNE with respect to the average acceptance ratio and resource utilization can be attributed to the fact that CNMMCF is using more resources at the link mapping stage since it performs node and link mappings separately. For VNA-1, while

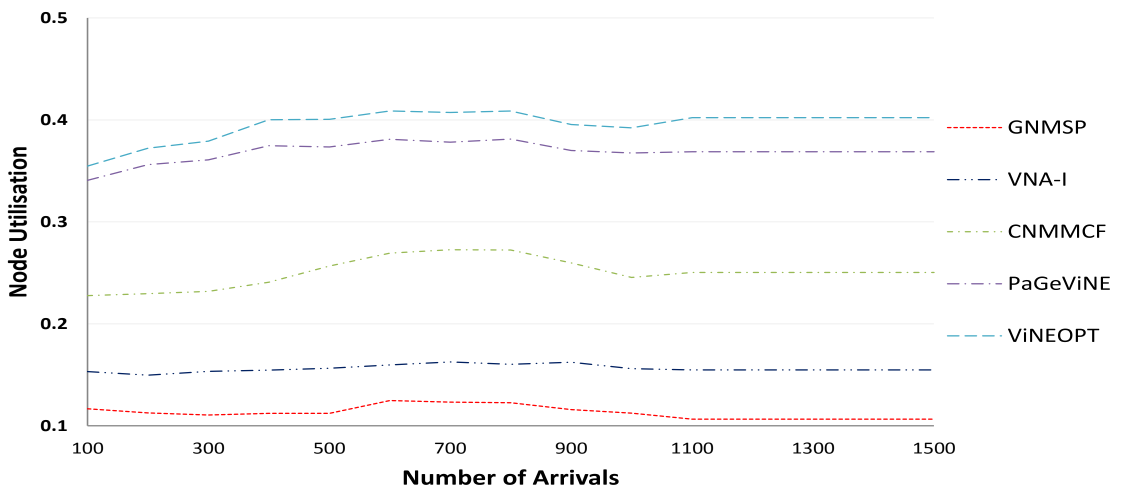


Figure 3-8: Average Node Utilisation

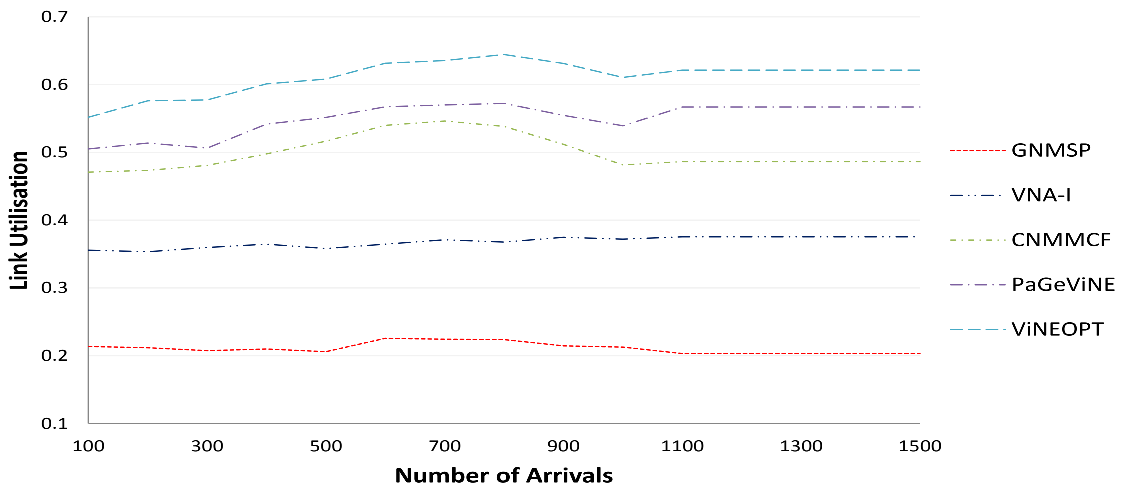


Figure 3-9: Average Link Utilisation

the node and link mapping is done in one shot, they are carried out sequentially, considering specific clusters of the substrate network each time. It is therefore expected that the results would not be as good as those achieved by a global solution based on mathematical programming. With respect to time complexity, the graphs in Fig. 3-10 show that the running times of GNMSP and VNA-1 are comparatively lower than those of PaGeViNE. Once again, this can be explained by the fact that these two solutions do not solve a mathematical program as PaGeViNE does. We also note that the computation time of PaGeViNE is slightly higher than that of CNMMCF. This can be attributed to the fact that PaGeViNE solves three mathematical pro-

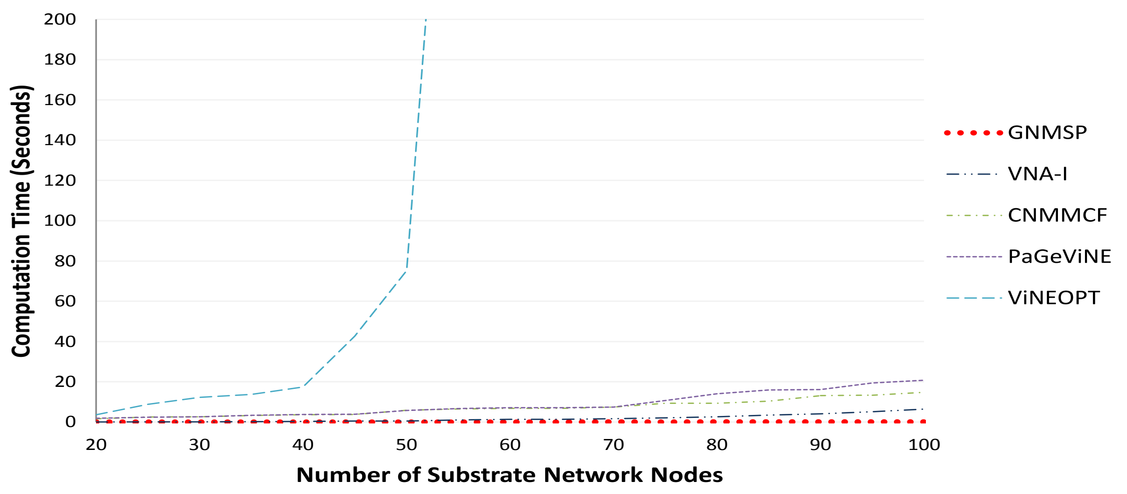


Figure 3-10: Average Computatation Time

grams, while CNMMCF solves only two. Moreover, it is expected that solving the problem in one shot requires more computation than solving it in two stages, since some of the mathematical programs solved in PaGeViNE are binary. With regard to ViNE-OPT we see that the computation time quickly grows exponentially. In fact, ViNE-OPT could not find a solution even after 1 hour for 60 substrate nodes[1].

We therefore note a significant improvement in time complexity of PaGeViNE compared to ViNE-OPT, while ensuring that the embedding quality is not greatly diminished. We are however mindful of the fact that we can reduce the computation time even more, if we use better ways of adding paths after the LP−D solution, than adding all possible shortest paths. We intend to investigate this further in future work, specifically by seeking better methods of performing the pricing for the dual variables for instance using more advanced search techniques.

3.7.5 Time Complexity

The mathematical formulation (3.21)−(3.29) involves solving a binary program. This problem is NP-hard in the general case, and only exponential algorithms are known to solve it in practice [113, 114]. Our approach is to reduce the number of input

[1]Once again, this is why the simulations for acceptance ratio were split into one with 20 substrate network nodes and another with 100 substrate nodes.

variables to the program using path generation. While a significant improvement in computation time is achieved compared to the optimal solution, the computation time is still slightly higher than that of CNMMCF and VNA-1 (see Fig. 3-10). Therefore, even though in practice there are *high performance* tools (such as [111]) for solving binary programs, more work can still be done for instance seeking a relaxation to the program which permits to solve it in polynomial time.

3.8 Conclusion

In this Chapter, we have proposed a near optimal one shot virtual network embedding solution that is based on column generation.

We started by defining a method that yields an initial solution whose objective was to keep the computation time of the solution as low as possible by including only the possible virtual to substrate node combinations, and to minimize the possibility of failure at the link mapping stage, by making the node mapping aware of the link mapping stage through the use of weights that bias the choice of virtual to substrate node mappings.

The VNE problem was then formulated as a MCF problem where each virtual link was represented as a flow to be mapped onto a substrate path. The dual of the problem was then derived, and the corresponding pricing problems determined. To ensure a faster solution, instead of performing iterative improvements in the initial solution, as would be in a typical column generation approach, this Chapter proposed a mechanism of achieving a good solution in only one iteration.

Through simulations, which include three models of state-of-the-art alternative proposals, we conclude that our approach has a comparative advantage over previous approaches in terms of solution quality. That is, we get a higher acceptance ratio of virtual networks and a higher level of resource utilisation in the substrate network. Specifically, compared with the best state-of-the-art approach, our proposal improves the acceptance ratio by about 35% while being only about 5% short of the optimal solution. In addition, the average node and link resource utilisations are improved by about 12% and 10% respectively. These improvements would directly translate into better profitability for infrastructure providers.

Further more, our approach significantly reduces the time complexity with respect to the optimal solution for the same problem conditions. Numerically, for a substrate network of 50 nodes, our proposal improves the computation time by about 1500% compared to the optimal solution, while remaining comparable with the state-of-art approaches. After 50 nodes, the optimal solution diverges exponentially, but our proposal remains comparable to the state-of-art approaches, encountering only a 25% deviation from current approaches for a substrate network of 100 nodes. Therefore, the results in this Chapter confirm hypothesises 1 and 2 as outlined in Section 1.2.

However, the mathematical programs formulated and solved in this Chapter can still be relaxed and heuristics that ensure that the virtual network embedding constraints are satisfied devised. This will be an objective of future research activities. In particular, we will investigate the feasibility of applying artificial intelligence techniques such as particle swarm optimisation [115] and tabu search [116] to the enhancement of our proposal.

Chapter 4

Reinforcement Learning-based Dynamic Resource Allocation

4.1 Introduction

The virtual network embedding proposal in Chapter 3 performs static embeddings in that it does not consider the possibility of re-mapping or adjusting resource allocation to one or more virtual networks. Specifically, it allocates a fixed amount of resources to the virtual nodes and links for the entire lifetime of the virtual network. Since user traffic is non-uniform (not static), fixed resource allocation could lead to an inefficient utilisation of overall network resources, especially if an infrastructure provider rejects requests to embed new virtual networks while reserving the resources for virtual networks that are not currently using them (are lightly loaded). It would therefore be important to dynamically allocate resources to virtual nodes and links taking into consideration actual need for these resources at any time.

While most existing works on dynamic resource allocation are based on control theory [32, 33], performance dynamics modeling [34, 35] and workload prediction [36, 37], it has been shown from artificial intelligence communities that distributed systems of autonomous learning entities can efficiently and dynamically allocate resources in different application domains [117, 118, 119].

In this Chapter, instead of allocating a fixed amount of resources to a given virtual network throughout its lifetime, we dynamically and opportunistically allocate resources to virtual nodes and links depending on their perceived needs. The opportunistic use of resources involves carefully taking advantage of unused virtual node and link resources to ensure that new virtual network requests are not rejected when resources reserved to already embedded virtual networks are idle. To this end, we use a demand-driven dynamic approach that allocates resources to virtual nodes and links using reinforcement learning (RL) [15].

The rest of the Chapter is organised as follows: We introduce reinforcement learning in Section 4.2, followed by a definition of the dynamic resource allocation problem in the context of network virtualisation in Section 4.3. The proposed reinforcement learning approach is then presented in Section 4.4 and evaluated and discussed in Section 4.5. We conclude the Chapter in Section 4.6.

4.2 Reinforcement Learning

Reinforcement Learning (RL) is a technique from artificial intelligence [120] in which an *agent* placed in an *environment* performs actions from which it gets numerical rewards. Fig. 4-1 shows an interaction between an agent and an environment in a typical RL scenario. For each learning episode [15], the agent perceives the current *state* of the environment and takes an *action*. The action leads to a change in the state of the environment (state transition), and the desirability of this change is communicated to the agent through a scalar *reward*, which is an evaluation of the desirability of the agents' action. In the next subsections, the three major components of a RL agent as shown in Fig. 4-1 are defined.

4.2.1 Learning Algorithm

There are many learning algorithms in RL, all falling with in three broad learning methods, which are: dynamic programming, Monte Carlo, and time difference methods [15].

Dynamic programming [121] uses the concepts of a dynamic system's state and

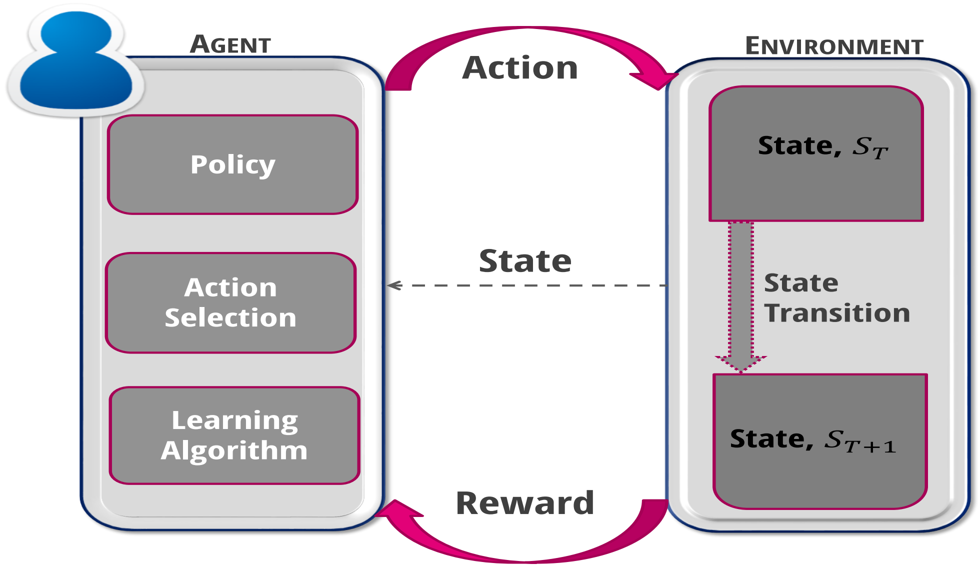


Figure 4-1: Agent-Environment interactions in Reinforcement Learning

of a value function to define a functional equation, which represents a controller for the system. It requires the knowledge of the probability of transiting from one state to another, and assumes that the measurements available on the system's state are detailed enough so that the the controller can avoid reasoning about how to collect information about the state [122]. Problems with these characteristics are best described in the framework of Markovian Decision Processes (MDPs) [123]. Therefore, the dynamic programming approach is to transform the problem of finding a good controller into the problem of finding a good value function. However, Dynamic programming suffers from the curse of dimensionality, meaning that its computational requirements grow exponentially with the number of state variables [15, 124]. On the other hand, Monte Carlo [125] and time difference (TD) [126] methods are primarily concerned with how an agent ought to take actions in an environment so as to minimise the notion of long-term cost, that is, so as to obtain the optimal policy, when the state transition probabilities are not known in advance. Specifically, if state transition probability is not known, but a sample transition model of states, actions and costs can be built, Monte Carlo methods can be applied to solve the problem, while, if the only way to collect information about the environment is to interact with it, TD methods are used. TD methods combine elements of dynamic programming and Monte Carlo ideas, they learn directly from experience which is a characteristic of Monte Carlo methods and they gradually update prior estimate values, which is common of dynamic programming [127].

Since for the problem of dynamic resource allocation in network virtualisation we do not have an estimate of the state transition probabilities, and because information about states, actions and rewards can only be obtained *over time* through interactions with both substrate and virtual networks, the natural reinforcement method of choice should come from TD methods. And since the most common TD learning algorithm is Q-learning [15, 128], the remainder of this Chapter will design and evaluate a Q-learning based algorithm for the opportunistic allocation of resources in network virtualisation.

Q-Learning

This is a time difference [15] learning algorithm that gradually builds information about the best actions to take in each possible state. This is achieved by finding a *policy* that maximises some long-term measure of reinforcement. Therefore, the main objective of a Q-learning agent is to maximise the overall reward it achieves throughout the learning period. In general, the objective of a learning algorithm is to learn an optimal *policy*. In algorithm 3, the generic psuedocode of a Q-learning agent is given.

4.2.2 Policy

A policy defines the learning agent's way of behaving at a given time. It is a mapping from each environment state to actions to be taken when in that state [15]. In Q-learning, a policy is composed of a *Q-value*, $Q(s, a)$ for each of the possible state-action combinations. $Q(s, a)$ is a measure of the desirability of each action, a while in the state, s. The learning process therefore involves continuously updating these values until they guide the agent to taking the best action while in any of the possible states [15].

Policy Update

While using the Q-learning algorithm, after every learning episode, an agent updates its Q-values using the Q-learning rule in (4.1).

$$Q(s_p, a_p) \leftarrow (1-\alpha)Q(s_p, a_p) + \alpha \left\{ r_p + \lambda \max_{a \in \mathcal{A}} Q(s_n, a) \right\} \tag{4.1}$$

where $Q(s_p, a_p)$ is the new value of state s_p corresponding to action a_p, r_p is the reward obtained from taking the action a_p while in state s_p and s_n is the next state resulting from taking the action a_p while in state s_p, implying that $Q(s_n, a)$ is the value associated with the action a of the state s_n. The parameters $0 \leq \alpha \leq 1$ and $0 \leq \lambda \leq 1$ are referred to as learning rate and discount factor respectively.

Learning rate: The learning rate (α) determines the extent to which newly acquired information overrides old information. If $\alpha = 0$ the agent does not learn anything, while a value of 1 for α would make the agent consider only the most recent information.

Discount factor: The discount factor (λ) models the importance that is attached to future rewards in comparison with immediate reward. Therefore, $\lambda = 0$ would make the agent short-sighted in that it would only give importance to current rewards, while a value close to 1 would make it strive for high rewards in the long-term, even if this means getting negative rewards in the short run.

4.2.3 Action Selection

A reinforcement learning agent usually has one of two possible strategies with regard to how actions are selected; (1) *exploit* the knowledge that it has found for the current state s by taking the action a that maximises $Q(s, a)$ i.e. take the action that it already knows is the best in that state, or (2) *explore* by selecting a different action from the one that it currently thinks is best, with the objective of trying to learn if there is an action better than what it currently thinks is best.

Balancing the ratio of exploration and/or exploitation is a great challenge in RL since it influences the convergence (learning) time and the quality of learned policies.

On one hand, too much exploration prevents the agent from maximising the short-term reward because selected exploration actions may yield negative reward from the environment. But on the other hand, exploiting uncertain environment knowledge prevents agents from maximising the long-term reward since selected actions may not be optimal [129]. Whether an agent explores or exploits its knowledge can generally be determined by its *action selection* criterion. The two most common action selection criteria are ϵ-greedy and softmax [15].

ϵ-greedy action selection

In ϵ-greedy, a greedy action (i.e. action with the highest Q-value) is selected most of the time, and – using a small probability, ϵ – a random action is chosen once in a while. This ensures that after many learning episodes, all the possible actions will be tried a high number of times, leading to an optimal policy. Although ϵ-greedy action selection is an effective and popular means of balancing exploration and exploitation in reinforcement learning, one drawback is that when it explores, it chooses equally among all actions. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action. In tasks where the worst actions are very bad, this may be unsatisfactory. The obvious solution is to vary the action probabilities as a graded function of their estimated values [15].

Softmax action selection

Softmax differs from ϵ-greedy in the way the random action is selected. A weight is assigned to each of the actions depending on their estimated values. A random action is selected based on the weight associated with it, ensuring that worst actions are unlikely to be chosen. When using softmax, an agent takes a random action a while in state s with a probability $\mathcal{P}(a|s)$ which is commonly defined by a *Gibbs* or *Boltzmann* distribution [130] as shown in equation (4.2).

$$\mathcal{P}(a|s) = \frac{\exp\{Q(s,a)/\tau\}}{\sum\limits_{\hat{a}\neq a} \exp\{Q(s,\hat{a})/\tau\}} \tag{4.2}$$

Algorithm 3 $Q - Learning(States, Actions, \lambda, \alpha)$

```
1:  Initialise: $States, Actions, \lambda, \alpha$
2:  Initialise $q - values$, $Q(s, a)$, arbitrarily
3:  repeat
4:      for Each Learning Episode do
5:          Choose $a_p$ from $s$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
6:          Take action $a_p$, observe $r_p$, $s_n$
7:          $Q(s_p, a_p) \leftarrow (1 - \alpha)Q(s_p, a_p) + \alpha \{r_p + \lambda \max_{a \in \mathcal{A}} Q(s_n, a)\}$
8:          $s_p \leftarrow s_n$
9:      end for
10: until Learning ends
```

where τ is a positive parameter called the temperature. High temperatures ($\tau \to \infty$) cause the actions to be almost equiprobable, while lower temperatures ($\tau \to 0^+$) cause the probability to be dependent on the Q-values.

Both methods (ϵ-greedy and softmax) have only one parameter that must be set [15]. Whether softmax action selection or ϵ-greedy action selection is better is unclear and usually depends on the task being considered.

4.2.4 Multi Agent Systems

When more than one agent interacts with each other, the resulting system is called a multiagent system [18]. A multiagent system (MAS) can be defined as a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators [131]. Depending on the application, the interaction between the agents in a MAS can either be cooperative or competitive. As stated by [132], multiagent systems are ideal for problems that require autonomous decision making capabilities. Specifically, multiagent systems are rapidly finding applications in a variety of domains, including robotics, distributed control, telecommunications, and economics [131]. Needless to mention, dynamic resource allocation in network virtualisation can be classified both under distributed control and telecommunications.

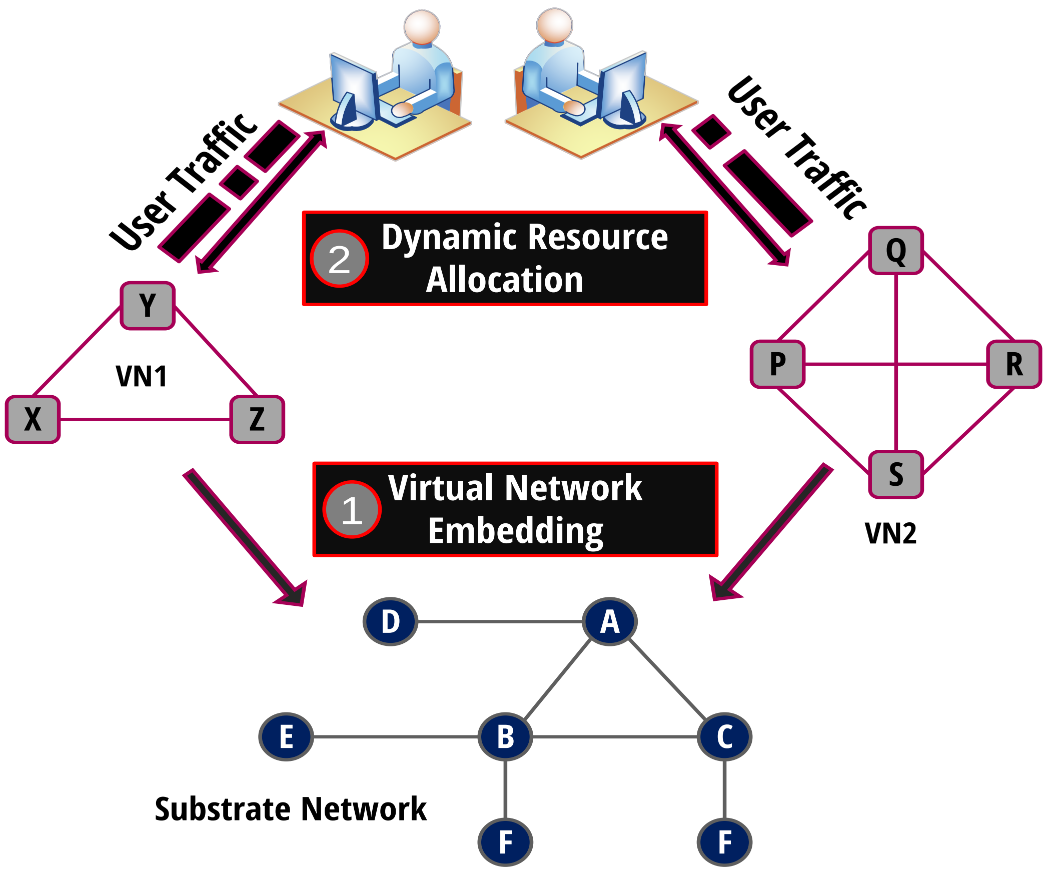


Figure 4-2: Dynamic Resource Allocation in Network Virtualisation

4.3 Problem Description: DRA in NVEs

The dynamic resource allocation problem considers that virtual network resource management does not stop at the embedding of virtual nodes and links to substrate nodes and paths. Instead, as illustrated in Fig. 4-2, after a successful virtual network embedding, there should be a lifecycle management of resources allocated/reserved for the mapped VN, aimed at ensuring efficient utilisation of overall SN resources. Our consideration is that SPs reserve resources to be used for transmitting user traffic, and therefore, after successful mapping of a given VN, user traffic in form of packets is transmitted over the VN. Actual usage of allocated resource is then monitored and based on the level of utilisation, we dynamically and opportunistically adjust allocated resources. The opportunistic use of resources involves carefully taking advantage of unused virtual node and link resources to ensure that new VN requests are not

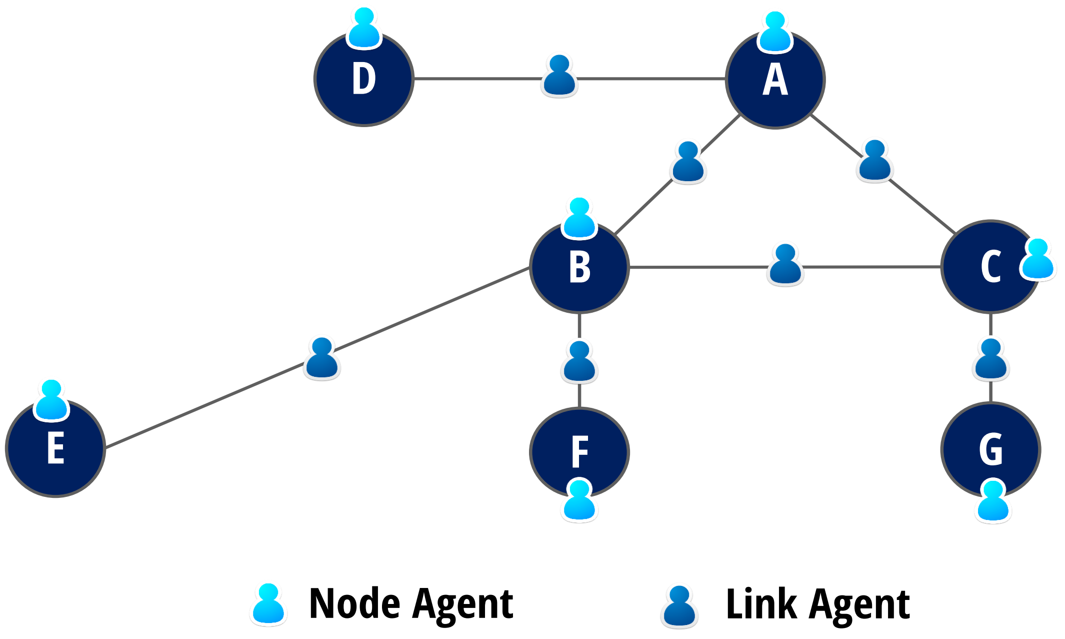


Figure 4-3: Substrate Network Modeling

rejected when resources reserved to already mapped VNs are idle. It is however a delicate trade-off which also ensures that the VNs always have enough resources to guarantee that QoS parameters are kept as established in the corresponding SLAs (or VN request specifications).

4.4 RL-based Dynamic Resource Allocation

As already mentioned, virtual network embedding allocates resources to virtual nodes and links based on the specification in the VN requests. Stopping at the embedding stage would result in a static allocation in which a fixed amount of substrate network resources is reserved for each virtual link and node irrespective of actual utilisation. The approach proposed in this Chapter is to dynamically adjust the resource allocation using RL. To this end, we represent the substrate network as a multiagent system shown in Fig. 4-3, in which each substrate node and link is represented by a node agent $n_a \in \mathcal{N}_a$ and a link agent $l_a \in \mathcal{L}_a$, where $\mathcal{N}_a$ and $\mathcal{L}_a$ are the sets of node agents and link agents respectively. The node agents manage node queue sizes while the link agents manage link bandwidths. The agents dynamically adjust the re-

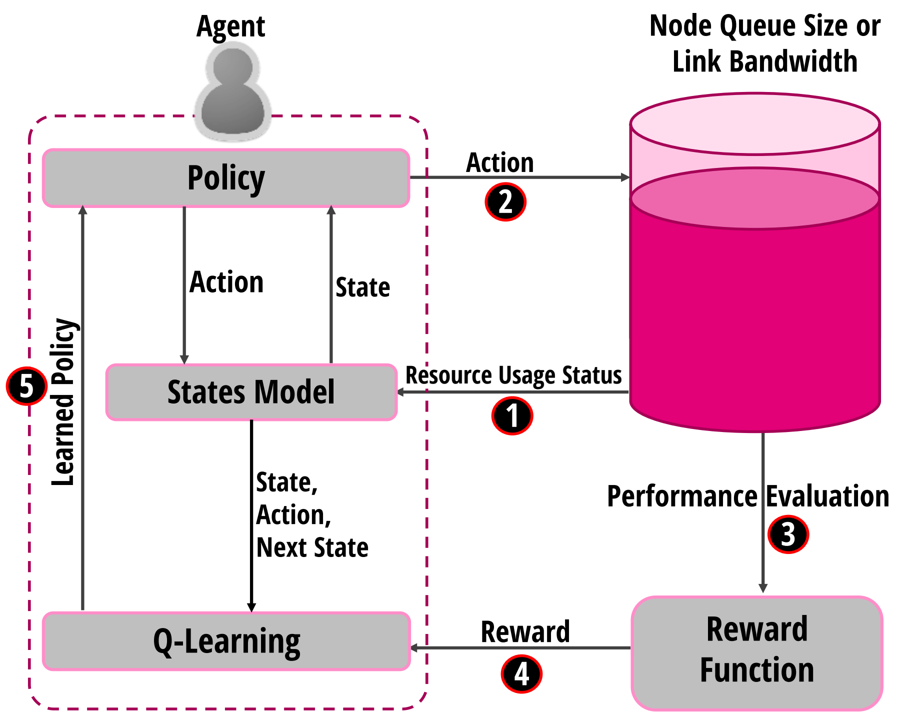
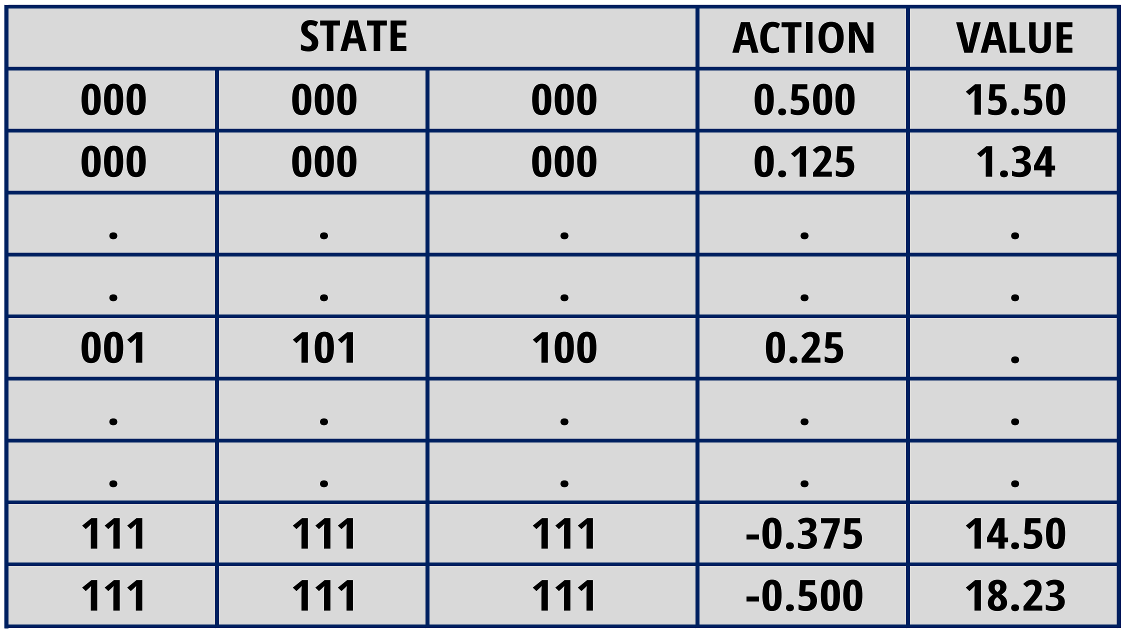


Figure 4-4: Learning System Modelling: Case of a single substrate node/link

sources allocated to virtual nodes and links, ensuring that resources are not left under utilised, and that enough resources are available to serve user requests. We consider that each $n_a \in \mathcal{N}_a$ has information about the substrate node resource availability as well as the resource allocation and utilisation of all virtual nodes mapped onto the substrate node. In the same way, we expect that each $l_a \in \mathcal{L}_a$ has information about substrate link bandwidth as well as the allocation and utilisation of these resources by all virtual links mapped to it. In case a given virtual link is mapped onto more than one substrate link, then each of the $l_a \in \mathcal{L}_a$ agents coordinate to ensure that their allocations do not conflict.

While RL is well studied, its application to dynamic resource allocation in network virtualisation is not trivial. In particular, it requires that all aspects of the RL scenario represented in Fig. 4-1 are modelled in the context of a network virtualisation environment. In Fig. 4-4, we show the proposed RL model for a given substrate node/link. As can be noted, the learning is made up of five steps. The agent starts by getting a resource usage status, which could have information such as the percentage



STATE			ACTION	VALUE
000	000	000	0.500	15.50
000	000	000	0.125	1.34
.	.	.	.	.
.	.	.	.	.
001	101	100	0.25	.
.	.	.	.	.
.	.	.	.	.
111	111	111	-0.375	14.50
111	111	111	-0.500	18.23

Figure 4-5: Example of Agent Policy: Look-up Table

of substrate node/link resources available and the ratio of total virtual node/link demand currently allocated. With this information, the agent takes an action, which could involve increasing or reducing the amount of resource allocated to the virtual node/link. The virtual node/link is then monitored to evaluate its performance e.g. in form of link delays (in case of virtual links) or packet drops (in case of virtual nodes). This evaluation is communicated to the agent in form of a reward, and based on this, the agent adjusts its policy so as to ensure that its future resource allocation actions are better. Each of the components of the model in 4-4 is detailed in subsequent subsections.

4.4.1 Policy

The policy is implemented by means of a lookup table which, for each state, maintains an updated evaluation (in form of Q-values) of all the possible actions. Since we have 9 possible actions and 512 possible states (as explained in the next two paragraphs), the size of our policy is $9 \times 512 = 4608$ state-action values. An example of some entries for the policy used in this Chapter is shown in Fig. 4-5.

Table 4.1: Variable States

Code	Percentage Value
000	$0 < \text{Variable} \leq 12.5$
001	$12.5 < \text{Variable} \leq 25$
010	$25 < \text{Variable} \leq 37.5$
011	$37.5 < \text{Variable} \leq 50$
100	$50 < \text{Variable} \leq 67.5$
101	$67.5 < \text{Variable} \leq 75$
110	$75 < \text{Variable} \leq 87.5$
111	$87.5 < \text{Variable} \leq 100$

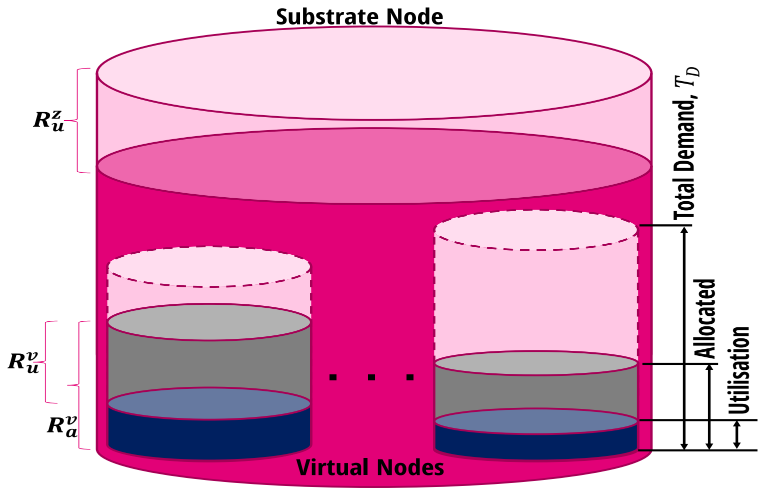


Figure 4-6: Learning Model: Illustration of Agent States (Case of a Node)

States: The state of any agent is a vector $\mathbf{S}$ with each term $s \in \mathcal{S}$ representing the state of one of the virtual links/nodes mapped onto it. The states in this work are discrete. We consider that the total resource demand of each virtual node or link can be divided into at least 8 resource chunks, each representing 12.5% of its total resource demand. For example, a virtual node could be allocated 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5% and 100% of its total demand. It is important to remark that these re-allocations are performed after a successful embedding. Therefore, all embeddings are performed based on the total demand of any given virtual node or link.

Table 4.2: Action Definitions

Action	**Description**
A_0	Decrease allocated resources by 50.0 percent
A_1	Decrease allocated resources by 37.5 percent
A_2	Decrease allocated resources by 25.0 percent
A_3	Decrease allocated resources by 12.5 percent
A_4	Maintain Currently allocated resources
A_5	Increase allocated resources by 12.5 percent
A_6	Increase allocated resources by 25.0 percent
A_7	Increase allocated resources by 37.5 percent
A_8	Increase allocated resources by 50.0 percent

We model the state of any virtual resource (node queue size or link bandwidth) v hosted on a substrate resource z, by a 3-tuple, $s = \left(R_a^v,\ R_u^v,\ R_u^z\right)$, where R_a^v is the percentage of the virtual resource demand currently allocated to it, R_u^v is the percentage of allocated resources currently unused, and R_u^z is the percentage of total substrate resources currently unused. Each of the 3 variables is allowed to take up 8 different states, each made up of 3 bits, e.g., [010]. These values are based on the relationship between a current value and a benchmark, for example, if a virtual node is allocated between 37.5% and 50.0% of its total demand, then $R_a^v = 011$. The complete set of these variables is shown in Table 4.1, which is valid for R_a^v, R_u^v and R_u^z. Therefore, each term of the state vector has 9 bits e.g. $(001, 100, 111)$, implying that we have $n = 2^9 = 512$ possible states. In Fig. 4-6, we represent a substrate node hosting several virtual nodes, and illustrate how each of the variables R_a^v, R_u^z and R_u^v are determined for any given virtual node.

Actions: The output of each agent is a vector $\mathbf{A}$ indicating an action $a \in \mathcal{A}$ for each of the virtual nodes/links mapped onto it. An agent can choose to increase or decrease the resources (queue size or bandwidth) allocated to any virtual node or link respectively. Specifically, as shown in Table 4.2, at any point each agent can choose 1 of the 9 possible actions, $a \in \mathcal{A} = (A_0, A_1, \ldots, A_8)$ each of which leads to a discrete

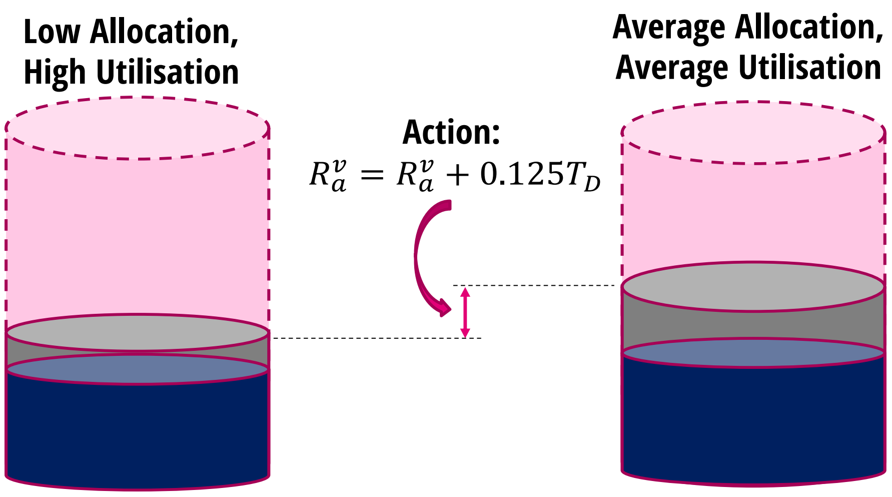

Low Allocation, High Utilisation
Average Allocation, Average Utilisation
Action:
$R_a^v = R_a^v + 0.125T_D$

Figure 4-7: Learning Model: Illustration of Agent Actions

change in resource allocation.[1] In Fig. 4-7, we illustrate the effect of agent actions by showing an initial state of a virtual resource with a low resource allocation and a high level of utilisation, and the effect resulting from the agents' decision to increase the resource allocation by 12.5% of the total virtual resource demand, T_D

States Model

The states model mimics the behaviour of the environment. When provided with a given status of the substrate and virtual networks resource allocation and utilisation levels i.e. the values R_a^v, R_u^v, and R_u^z, a states model returns a state $s \in \mathcal{S}$. In the same way, when provided with a given state $s_p = \left(R_{a_p}^v, R_{u_p}^v, R_{u_p}^z\right)$ and an action a_p, the states model provides the next state $s_n = \left(R_{a_n}^v, R_{u_n}^v, R_{u_n}^z\right)$. It is in general a model of the substrate and virtual network resources and how the different possible actions affect the allocation of substrate resources to virtual networks.

4.4.2 Reward Function

When an agent takes an action, the networks are monitored, recording the link delays, packet drops and virtual and substrate network resource utilisation so as to determine

[1]In all cases, the percentage change is with respect to the total demand of the virtual node or link.

a reward. Specifically, the reward resulting from a learning episode of any agent is a vector $\mathbf{R}$ in which each term $r(v)$ corresponds to the reward of an allocation to the virtual resource[2] v, and is dependent on the percentage resource allocation R_a^v, the percentage resource utilisation $\hat{R}_u$ i.e. $\hat{R}_u = 1 - R_u^v$, the link delay $\hat{D}_{ij}$ in case of $l_a \in \mathcal{L}_a$ and the the number of dropped packets $\hat{P}_i$ in the case of $n_a \in \mathcal{N}_a$.

$$r(v) = \begin{cases} -100 & \text{if } R_a^v \leq 0.25 \\ \nu \hat{R}_u - \left(\kappa \hat{D}_{ij} + \eta \hat{P}_i\right) & \textit{otherwise} \end{cases}$$

Where ν, κ and η are constants aimed at adjusting the influence of the variables $\hat{R}_u$, $\hat{D}_{ij}$ and $\hat{P}_i$ to the overall reward. In this Chapter, the values $\nu = 100$, $\kappa = 1000$ and $\eta = 10$ are used. These values have been determined through simulations, for example, by noting that the values of $\hat{P}_i$ are about 100 times more than those of $\hat{D}_{ij}$ (See Figs. 4-15 and 4-17). We therefore aim at scaling them to comparable magnitudes so that they can have the same effect on $r(v)$. $\hat{D}_{ij}$ and $\hat{P}_i$ are measures of the performances of link agents and node agents respectively. Therefore, for $n_a \in \mathcal{N}_a$, $\hat{D}_{ij} = 0$ while $\hat{P}_i = 0$ for $l_a \in \mathcal{L}_a$. The objective of the reward function is to encourage high virtual resource utilisation while punishing $n_a \in \mathcal{N}_a$ for dropping packets and $l_a \in \mathcal{L}_a$ for having a high delay. We also assign a punitive reward of -100 to resource allocations below 25% to ensure that this is the minimum allocation to a virtual resource and therefore avoid adverse effects to QoS in cases of fast changes from very low to high VN loading.

4.4.3 Q-Learning

In this subsection, we propose a decentralised Q-learning based algorithm to iteratively approximate the state-action values, and then use these values to *select actions* for the allocation of substrate resources to the virtual nodes and links. As shown in algorithm 4 the learning algorithm is made up of three major steps; policy initialisation, policy update and action selection. It is worth noting that this algorithm is

[2]We use the term virtual resource to mean either a virtual node queue or virtual link bandwidth.

Algorithm 4 *Agent Learning Algorithm*

```
1: POLICY INITIALISATION:
2: for $s \in \mathcal{S}, a \in \mathcal{A}$ do
3:     Initialize the Q-table values $Q(s, a)$
4: end for
5: Determine current state $s_c$
6: previous state, $s_p = s_c$, previous action, $a_p = A_0$, next state, $s_n$.
7: repeat
8:     Wait(Learning Interval)
9:     POLICY UPDATE:
10:    Read $s_p$, $a_p$, $s_n$
11:    Observe Virtual Network Performance and Determine reward for previous action $r_p$.
12:    Update the Q-Table using the equation (4.1)
13:    ACTION SELECTION:
14:    Determine current state, $s_c$.
15:    Choose an action, $a_c \in \mathcal{A}$, for that state using a given action selection criterion
16:    Take the action, $a_c$ and determine next state $s$.
17:    Set $s_p = s_c$, $a_p = a_c$, $s_n = s$
18: until Learning is stopped
```

slightly different from the "conventional" Q-learning algorithm [15] because instead of getting a reward immediately, in our case the reward of a given learning episode are used just before the following episode after a performance evaluation has been made. We briefly describe each of these steps in the following paragraphs.

Policy Initialisation

Before learning can start, we need to initialise the learning policy. One possible approach is to assign random or constant values to all states and actions. However, since Q-learning requires all state-action pairs to be visited *at least once* so as to reach optimality, using random or constant initial values may lead to a slow convergence especially for a policy with many state-action values like we have in our approach. The idea is to start with a Q-table with values that more easily represent the expected actions of the agents. We therefore propose an initialisation approach, given by (4.3), that improves the rate of convergence.

$$Q(s,a) = \frac{a}{\Psi} \times (s - 255) \tag{4.3}$$

Where Ψ is a constant aimed at scaling the $Q(s,a)$ values to the required ranges. The formula in equation (4.3) is based on observing that the free substrate and virtual resources increase as we move from state $(000, 000, 000)$ to $(111, 111, 111)$. Therefore, the rationale behind equation (4.3) is to generally bias the agents to increase resource allocation to the virtual network whenever it finds itself in a state closer to $(000, 000, 000)$ and reduce the allocation while in states closer to $(111, 111, 111)$. To this end, we represent each of the states $s \in \mathcal{S}$ with integers $[0, 511]$ and all the actions $a \in \mathcal{A}$ with integers $[0, 8]$. We then divide the total state space into two; such that while in states $[0 - 255]$ the agents in general allocate more resources to the virtual network and then allocate less while in the states $[256 - 512]$. In Fig. 4-8, we show the different possible combinations with their respective values. As shown in the figure, for the same state $(000, 000, 000)$, action A_0 has a Q-value of 0 while action A_8 has -20.4. This initialization strategy has been proven useful as it will be shown in Section 4.5.2.

Policy Update

The idea of learning is to gradually improve the policy until an optimal or near optimal policy is reached. This is achieved by updating the policy table after every learning episode. In our work, the policy table (see Fig. 4-5) is updated using the Q-learning equation (4.1).

Action Selection

An agent can select one out of the 9 possible actions. Since the suitability of any of the two action selection methods described in Section 4.2.3 depends on the nature of the task, in this thesis, we evaluate both of them with respect to our specific learning task, and their respective performances are discussed in Section 4.5.2.

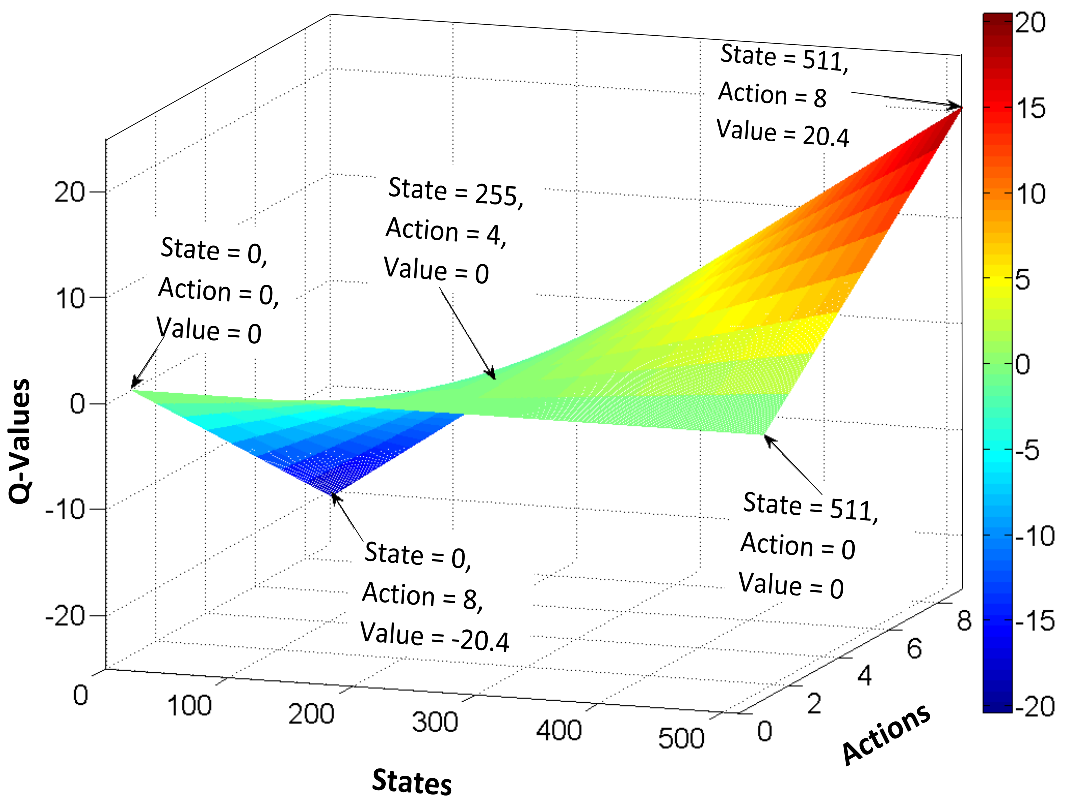


Figure 4-8: Proposed Policy Initialisation Function

4.4.4 Time Complexity of the Proposed Learning Algorithm

We now formally analyse the time complexity of Algorithm 4. The initialisation step in Line 2 requires initialisation of the learning policy and can be solved in $O(|N_{s-a-v}|)$, where N_{s-a-v} is the number of state-action-values (4608 in this proposal). Lines 5, 14 and 16 may each require iteration through all possible states in the worst case and can therefore be solved in $O(|\mathcal{S}|)$. Finally, the "for" loop in Line 15 runs in time $O(|\mathcal{A}|)$. Therefore, each episode of the proposed algorithm can be solved in linear time determined by the policy size.

4.4.5 Cooperation between Agents

Since a virtual link can be mapped to more than one substrate link, the agents $l_a \in \mathcal{L}_a$ that support the given virtual link must cooperate to avoid conflicting resource allocations. We accomplish this by allowing the agents to exchange messages. We consider that each agent $l_a \in \mathcal{L}_a$ maintains a record of other agents $l'_a \in \mathcal{L}_a$ with which it is managing the resources of a given virtual link. This set of collaborating agents changes dynamically for each agent as new virtual networks are embedded and old ones leave. To ensure that the agents $l_a \in \mathcal{L}_a$ do not perform conflicting actions, only one of them learns at any given time. This is achieved by starting the learning processes of each agent at different times on their creation and thereafter performing learning at regular intervals. After each learning episode, if an agent $l_a \in \mathcal{L}_a$ needs to change an allocation, and the virtual link under consideration is mapped onto more than one substrate link, a message is sent to all the other affected substrate link agents $l'_a \in \mathcal{L}_a$ with information about the proposed allocation. This allows for a synchronised allocation of virtual link resources i.e. all agents controlling a given virtual link take similar actions. This is reasonable since all agents belong to the same organisation (the SN) and learn the same policy; as they cannot have conflicting objectives. It would however be interesting to consider a more advanced cooperation protocol that allows for possibilities of agents accepting or rejecting proposals of other agents, which would be ideal in heterogeneous environments where the agents belong to different organisations and hence have different objectives.

4.4.6 Scalabity of Proposed Learning Algorithm

It is worth noting that in general, a virtual link is mapped to $2-3$ substrate links. This means that at any point, a given agent only needs to send update messages to about $1-2$ other agents[3]. We consider that this number of update messages is manageable, and would not congest the network. In addition, since the communicating agents represent substrate links that are part of a simple substrate paths, they should be connected to each other, and hence the update messages are restricted to small regions even for big network sizes.

[3]This is determined by observing VNE results from CPLEX using a one-shot mathematical program to perform the embedding.

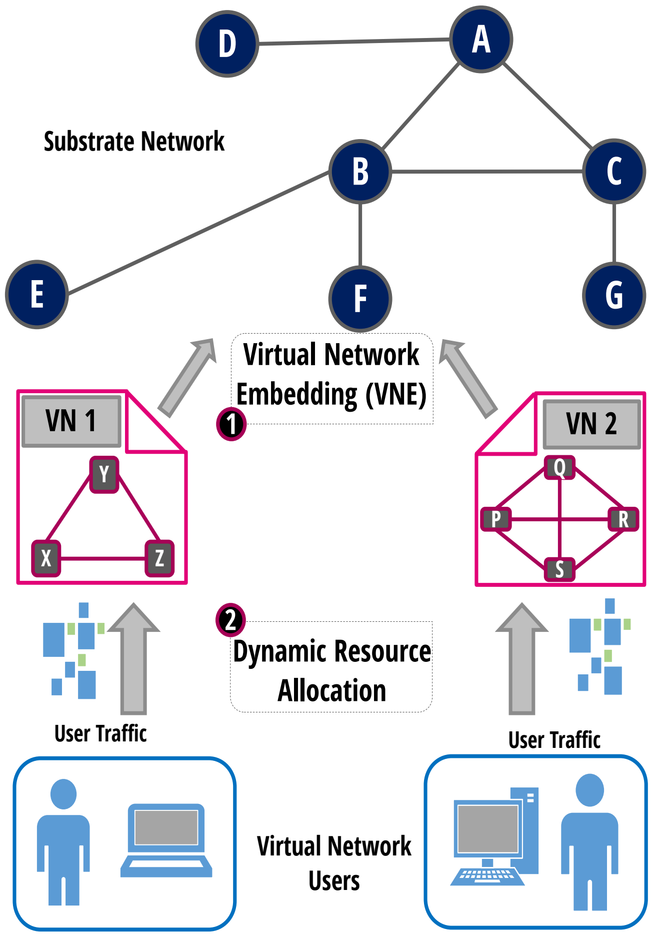


Figure 4-9: Simulation Setup

4.5 Performance Evaluation

4.5.1 Simulation Setup

To evaluate the performance of the proposed approach, Fig. 4-9 shows the setup of the simulation that was used. Specifically, we added a network virtualisation module to NS3 [133]. Table 4.3 shows the NS3 parameters used in our simulations. The setup is such that every time a virtual network request is accepted by the substrate network, the virtual network topology is created in NS3, and a traffic application starts transferring packets over the virtual network. The traffic used in this simulation is based on real traffic traces from CAIDA anonymised Internet traces [134]. This data set contains anonymised passive traffic traces from CAIDA's equinix-chicago

Table 4.3: NS3 Parameters used in Simulation

Parameter	**Value**
Queue Type	Drop Tail
Queue drop Mode	Bytes
Maximum Queue Size	6,553,500 Bytes
Maximum Packets Per VN	3500 Packets
Number of VNs	1024
Network Mask	255.255.224.0
IP Adress Range	10.0.0.0 – 10.255.224.0
Network Protocol	IPv4
Transport Protocol	TCP
Packet MTU	1518 Bytes
Packet Error Rate	0.000001 per Byte
Error distribution	Uniform (0, 1)
Port	8080

and equinix-sanjose monitors on high-speed Internet backbone links, and is mainly used for research on the characteristics of Internet traffic, including flow volume and duration [134]. The trace source was collected on 20th December 2012 and contains over 3.5Million packets. We divide these packets amoung 1000 virtual networks, so that each virtual network receives about 3500 packets. These traces are used to obtain packet sizes and time between packet arrivals for each VN. As the source and destination of the packets are anonymised, for each packet in a given VN, we generate a source and destination IP address in NS-3 using a uniform distribution.

The substrate and virtual network topologies are generated using Brite [110] with settings shown in Table 4.4. Simulations were run on an Ubuntu 12.04 LTS Virtual Machine with 4.00GB RAM and 3.00GHz CPU specifications. Both substrate and virtual networks were generated on a 250×250 grid. The queue size and bandwidth capacities of substrate nodes and links as well as the demands of virtual networks

Table 4.4: Brite Network Topology Generation Parameters

Parameter	**Substrate Network**	**Virtual Network**
Name (Model)	Router Waxman	Router Waxman
Size of main plane (HS)	250	250
Size of inner plane (LS)	250	250
Node Placement	Random	Random
GrowthType	Incremental	Incremental
Neighbouring Nodes	3	2
alpha (Waxman Parameter)	0.15	0.15
beta (Waxman Parameter)	0.2	0.2
BWDist	Uniform	Uniform
Minimum BW (BWMin)	2×10^6 bps	1×10^6 bps
Maximum Dev. (BWMax)	8×10^6 bps	1×10^6 bps

are all uniformly distributed between values shown in Table 4.5. Link delays are as determined by Brite. Each virtual node is allowed to be located within a uniformly distributed distance $7.5 \leq x \leq 15$ of its requested location, measured in grid units. We assumed that virtual network requests arrive following a Poisson distribution with an average rate of 1 per minute. The average service time of each virtual network is 60 minutes and is assumed to follow a negative exponential distribution.

4.5.2 Initial Evaluations

The initial evaluations are aimed at determining the appropriate action selection method for our task, as well as the effectiveness of the proposed policy initialisation scheme. Both of these evaluations are based on a comparison of agent actions with optimal actions. We define an optimal action for an agent as that action that would lead to a resource allocation equal to what the network is actually using. The deviations in these evaluations are therefore with reference to actual resource usage in a similar network that is not performing dynamic allocations.

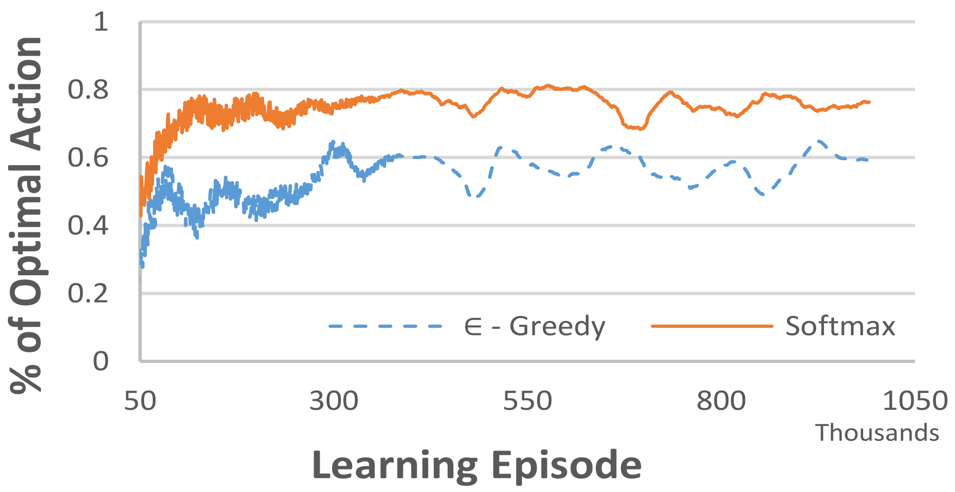


Figure 4-10: Performance Comparison of ϵ-greedy and Softmax

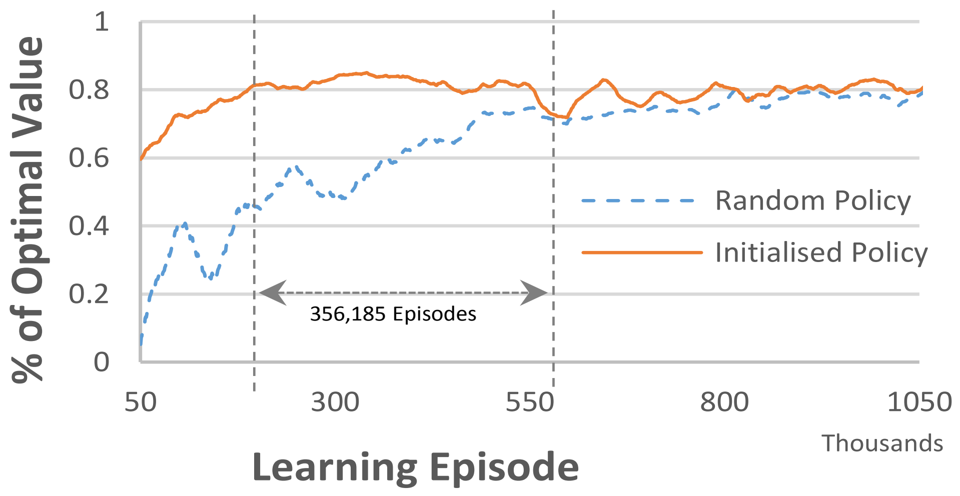


Figure 4-11: Effect of biased policy initialisation

Fig. 4-10 compares the performance of the action selection methods ϵ-greedy and softmax. It is evident that for this task, softmax performs better than ϵ-greedy. The difference in performance can be attributed to the fact that for ϵ-greedy, when random actions are chosen, the worst possible action is just as likely to be selected as the second best, yet softmax favours actions with better values. This could also explain why softmax actions appear to be relatively stable as compared to those by ϵ-greedy. In Fig. 4-11, we show the effect of the proposed initialisation method (action selection based on softmax). We observe that an initialised policy requires about $350,000$ learning episodes less to converge than a random policy. This can be attributed to the

Table 4.5: Simulated Substrate and Virtual Network Properties

Parameter	**Substrate Network**	**Virtual Network**
Minimum Number of Nodes	25	5
Maximum Number of Nodes	25	10
Minimum Node Queue Size	(100×1518) Bytes	(10×1518) Bytes
Maximum Node Queue Size	(200×1518) Bytes	(20×1518) Bytes
Minimum Link Bandwidth	2.0Mbps	1.0Mbps
Maximum Link Bandwidth	10.0Mbps	2.0Mbps

agents not having to explore all possible actions in all states as initialisation makes some actions more valuable than others.

For these evaluations as well as those in the next subsection the reinforcement learning parameters used are: learning rate, $\alpha = 0.8$, discount factor, $\lambda = 0.1$ and temperature, $\tau = 1$. We remark that based on the results of the evaluations in this subsection, the rest of the simulations in this Chapter are based on an initialised policy and the action selection method is softmax.

4.5.3 Performance Metrics

We evaluate the performance of our proposal on two fronts; the quality of the embeddings, as well as the quality of service of the virtual networks. The idea is that the opportunistic use of virtual network resources should not be at the expense of the service quality expectations of the network users.

Embedding Quality

This is evaluated using the acceptance ratio and total instantaneous accepted virtual networks. The acceptance ratio is a measure of the long term number of virtual network requests that are accepted by the substrate network. The total instantaneous accepted virtual networks is a measure of the embedding cost incurred by a given substrate network, as a substrate network that incurs a lower embedding cost normally

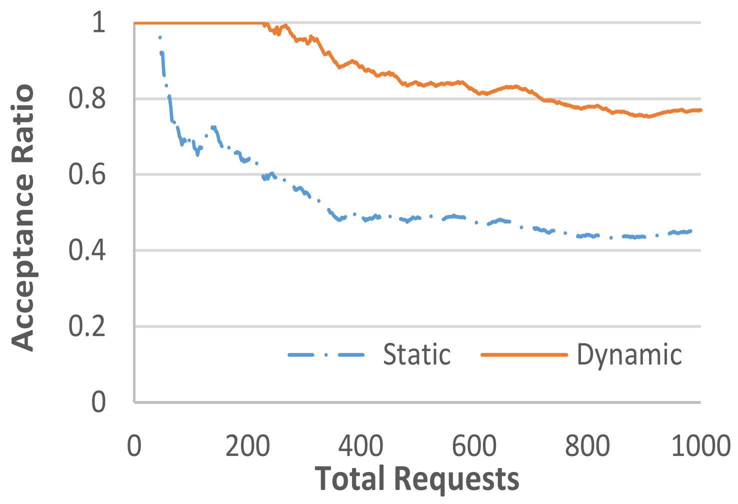


Figure 4-12: VN Acceptance Ratio

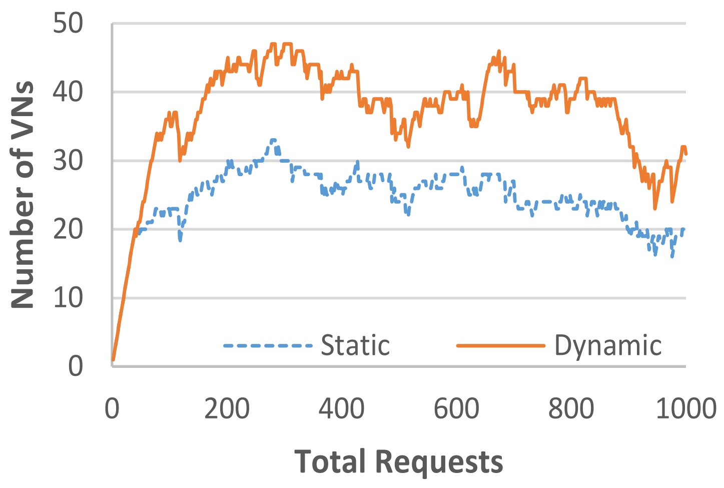


Figure 4-13: Number of Accepted Virtual Networks

has more extra resources at any point and hence is able to have many embedded virtual networks at any point.

Quality of Service

We use the packet delay and drop ratio as indications of the quality of service. We define the packet delay as the total time a packet takes to travel from its source to its final destination. The drop ratio is defined as the ratio of the number of packets dropped by the network to the total number of packets sent. As shown in Table 4.3, we model the networks to drop packets due to both node buffer overflow as well as

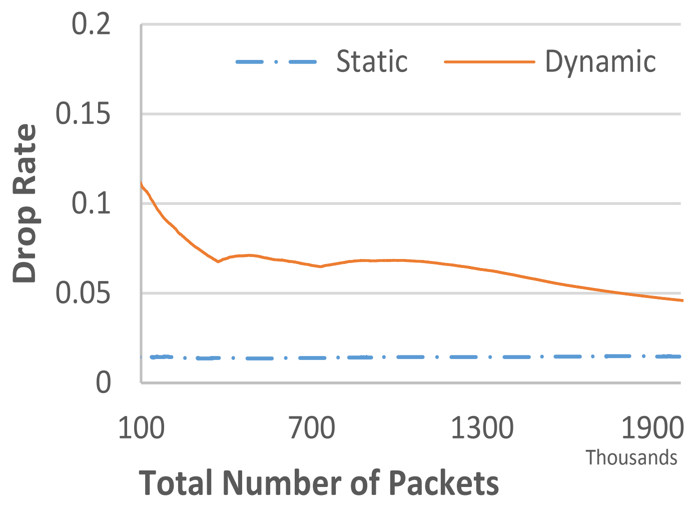


Figure 4-14: Node Packet Drop Ratio

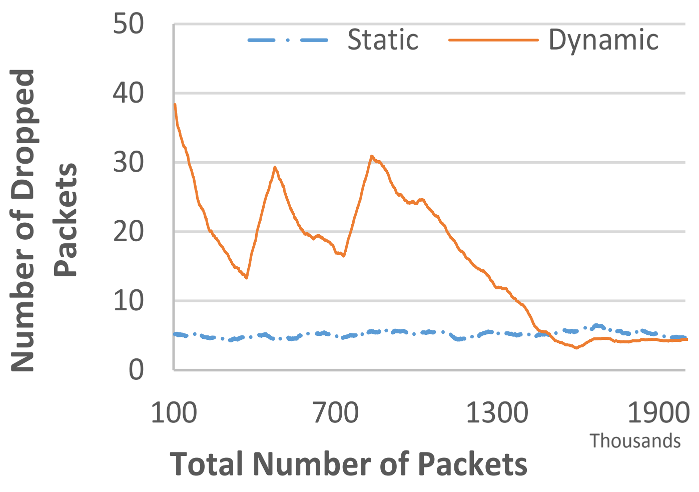


Figure 4-15: Node Packet Drop Ratio Variation

packet errors. In addition, as it is more important in some applications, we define the variations of these two parameters. The jitter (delay variation) is defined as the difference between delays during different time periods, while the drop ratio variation is defined as the variation between packet drops in different time periods. The time interval to update the measurements corresponds to the transmission of 50 packets.

4.5.4 Discussion of Results

The simulation results are shown in Fig. 4-12 – 4-17. As can be seen from Fig. 4-12, the dynamic approach performs better than the static one in terms of virtual network

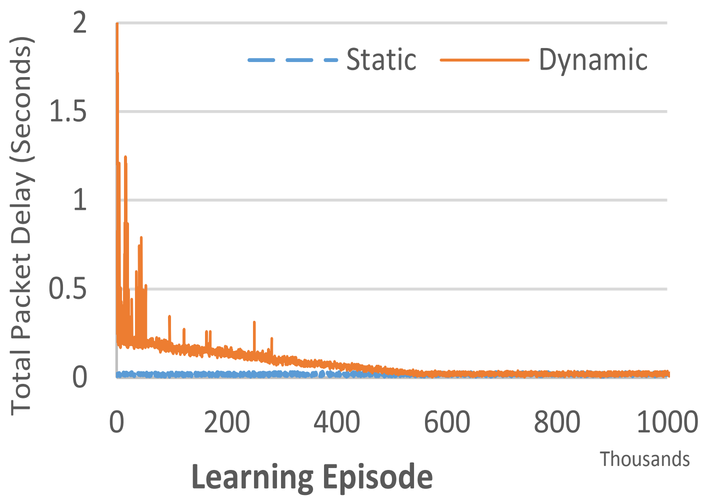


Figure 4-16: Link Packet Delay

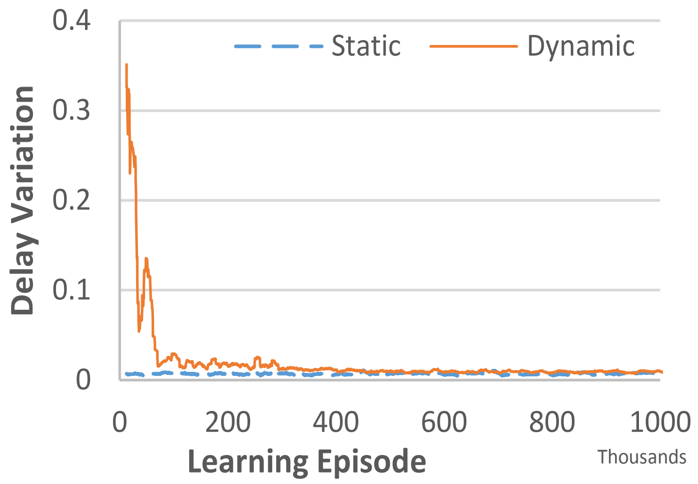


Figure 4-17: Link Packet Delay Variation

acceptance ratio. This can be attributed to the fact that in the dynamic approach the substrate network always has more available resources than in the static case, as only the resources needed for actual transfer of packets is allocated and/or reserved for virtual networks. This is further confirmed by Fig. 4-13 which shows that at any given point a substrate network that dynamically manages its resources is able to embed more VNs than a static one.

Fig. 4-14 shows that the packet drop ratio of the static approach is in general constant (due to packet errors as well as buffer overflows) while that of the dynamic approach is initially high, but gradually reduces. The poor performance of the dynamic approach at the start of the simulations can be attributed to the fact that at the beginning of the simulation when the agents are still learning, the virtual node queue sizes are allocated varying node buffers that lead to more packet drops. In fact, this initial number of packet drops affects the rate at which the overall drop ratio reduces towards the one for the static approach. This can be confirmed by observing the actual periodic drops in packets as shown in Fig. 4-15 which show that the total number of packets dropped by both approaches is comparable towards the end of the simulation.

Similarly, Fig. 4-16 shows that the packets in the dynamic approach initially have higher delays than those in the static approach. Once more, the reason for this is the initial learning period of the agents. This is again confirmed by observing that the delay variations in Fig. 4-17 easily converge to those of the static approach. It is however worth noting that unlike the packet drop ratio (Fig. 4-14), the actual delay (Fig. 4-16) of the dynamic approach finally converges to that of the static approach. Again, this could confirm that the slow convergence of the drop ratio is due to the initial packet drops, since initial packets delays would not affect the delays of other packets, yet initial packet drops remain factors in the final drop ratio.

4.6 Conclusion

This Chapter has proposed a dynamic approach to the management of resources in virtual networks, that opportunistically takes advantage of unused virtual network resources by allocating them to other virtual networks that need them, while ensuring that at any point, if needed, all virtual networks can have their originally assigned resources.

We started by modelling the substrate network as a distributed set of autonomous entities, in which each node and link was represented by an intelligent agent. These agents were then tasked to dynamically allocate substrate network resources ensuring that while virtual nodes and links do not keep idle resources, they always have enough resources to minimise node packet drops as well as link delays. We also proposed a method of initialising the agent learning policy that biases the agents to making certain decisions while at the beginning of the learning phase, and an agent action coordination approach that ensures that link agents do not take conflicting actions.

We have been able to show, through simulation, that our proposal improves the acceptance ratio of virtual networks by about 40% compared to a static resource allocation scheme (which would directly translate into better revenue for the substrate network providers), while ensuring that, after the agents have learnt an allocation policy, the quality of service (measured in terms of node packet drop ratio and link delays) to the virtual networks is not negatively affected. We have also been able to enhance the convergence rate of the learning algorithm by about 60% through the use a policy initialisation. Needless to mention, these results confirm hypothesises 3 and 4 as outlined in Section 1.2.

We are however mindful of the fact that it could require a much higher number of learning episodes for the overall packet drop ratio in Fig. 4-14 to finally converge to that of the static approach. This is because we used a learning policy with 4608 state-action values. With this high number of state-action values, the agents require a lot of time to learn an optimal policy. Moreover, it could improve the accuracy and precision of the agents' actions even more if the state-action values (policy size) were increased. It would therefore be better to use function approximation [135] or a more compact parameterised function representation to model the agents' policy other than a look-up table. In the next Chapter, a neural network-based approach is proposed to do away with the need for a look-up table based policy.

Chapter 5

Artificial Neural Network-based Dynamic Resource Allocation

5.1 Introduction

The decentralised reinforcement learning-based dynamic resource allocation scheme proposed in Chapter 4 uses a look-up table based policy. Since a look-up table representation suffers from the curse of dimensionality [136], the state and action spaces in Chapter 4 were discretised to limit the size, as well as the high memory required for reading, writing, and storage of the learning policy. This however comes at a cost of efficiency, as the learning algorithm is constrained in terms of perception and action granularity.

In this Chapter, we improve the efficiency of the approach in Chapter 4 by proposing an autonomous system based on artificial neural networks (ANN) [137] to achieve a self-adaptive allocation of resources to virtual networks, without restricting the input-output space dimensions. We start by representing each substrate node and link as an ANN whose input is the network resource usage status and the output an allocation action. We then use a reinforcement-like error function to evaluate the desirability of ANN outputs, and hence perform online training of the ANN. The use of neural a network has a great advantage over a lookup table in the context of our problem in that it has the ability to generalise [138]. This means that the neural network would still provide us with an appropriate state-action value even when presented with a previously un encountered virtual-substrate network resource state.

The rest of this Chapter is organised as follows: Section 5.2 gives a brief theoretical background on artificial neural networks. The proposed approach is presented in Section 5.3 and evaluated in Section 6.7. Finally, Section 6.8 concludes the Chapter.

5.2 Artificial Neural Networks (ANN)

ANNs are computational methodologies inspired by networks of biological neurons to perform multifactorial analyses [16]. A major strength of neural networks is their ability to handle high-dimensional input, as they do not suffer from the curse of dimensionality [139]. Therefore, although their complexity can make them difficult to harness, neural networks can be very efficient. In particular, they have excellent generalisation capabilities, which can help to solve difficult reinforcement-learning tasks, especially when the dimension of the state space is high [140].

5.2.1 Structure of a Neuron

Neural networks are made up of simple interconnected computing nodes known as *neurons*; which are connected by *links*. As shown in Fig. 5-1, a neuron operates as nonlinear summing device. It receives one or more inputs, which are first multiplied by *weights*[1] along each link, and then *summed* to produce a weighted sum. The weighted sum is then passed through an activation function (such as the logistic (or sigmoid) function [137]), which determines the input-output behaviour of the neuron.

5.2.2 Neural Network Structure

In ANNs, neurons are arranged in layers, with each layer consisting of one or more neurons. Fig. 5-2 shows the most commonly used structure of ANNs, which is

[1]The weights are a set of predefined numbers.

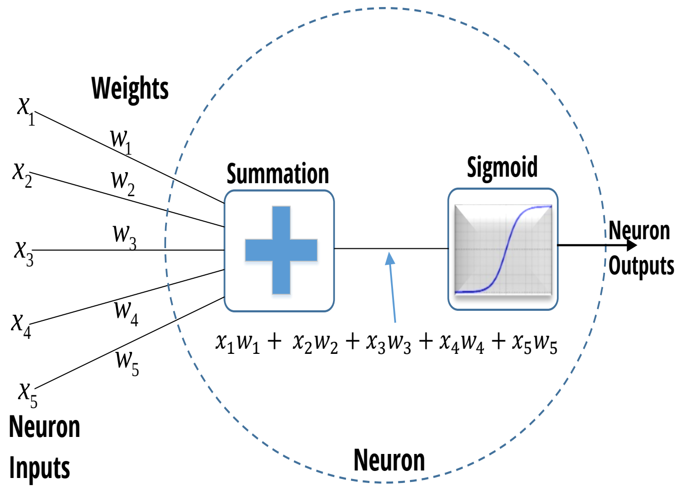


Figure 5-1: Structure of a Neuron

made up of 3 layers; an input layer, a hidden layer and an output layer [141]. Each layer consists of one or more neurons. While in general information can flow in both directions (with feedback paths), what we consider in this Chapter are ANNs where information flow is from input to output. These are called feed forward neural networks. It is also important to note that the neurons in the input layer are passive (they do not modify the data) as each of them receives a single value at its input, and duplicates the value to its multiple outputs and sends it to all hidden nodes (in case of fully connected networks [142]). On the other hand, the nodes of the hidden and output layer are active since they actually modify the data at their inputs.

5.2.3 Learning in Neural Networks

Before neural networks can be used for any task, they should be trained. The learning process of neural networks is represented in Fig. 5-3. Learning in ANNs consists of determining the proper set of connection weights to estimate a given training set. This requires that for every input, a desired/target output must be known so as to determine the error. The expected output is compared with an actual output to determine an error. It is this error that is used to adjust the weights along the links of the neural network, with the objective of gradually minimising the error. Training continues until a neural network produces outcome values that match the

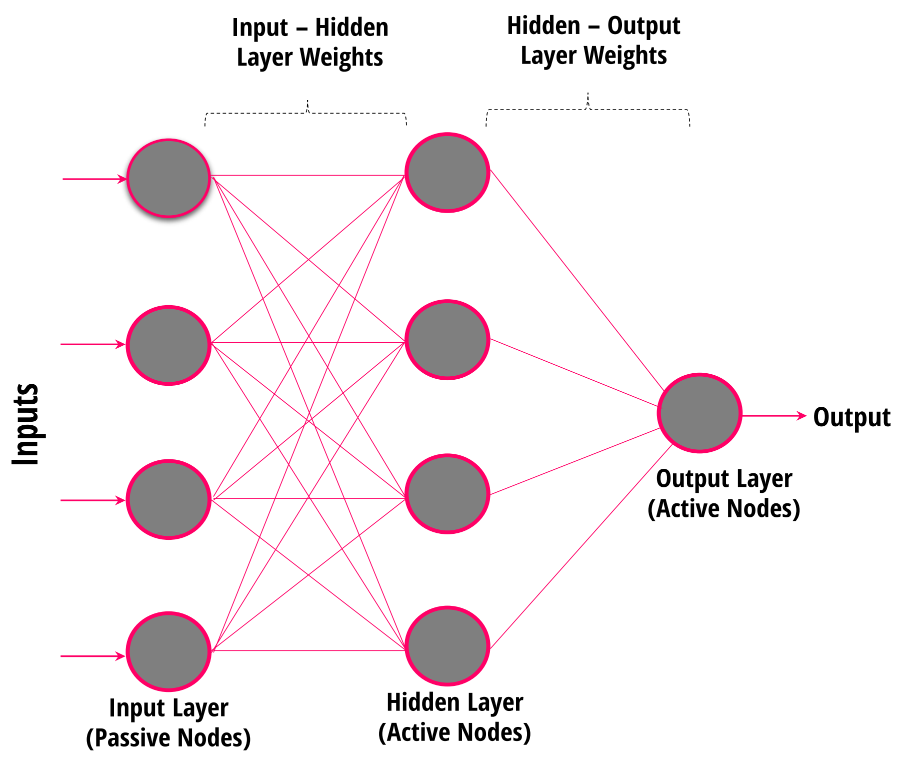

Input – Hidden Layer Weights
Hidden – Output Layer Weights
Inputs
Output
Output Layer (Active Nodes)
Input Layer (Passive Nodes)
Hidden Layer (Active Nodes)

Figure 5-2: Example of a Neural Network

known outcome values within a specified accuracy level, or until it satisfies some other stopping criteria. The most popular method for learning in ANNs is called back-propagation (BP) [120].

5.2.4 Back-propagation (BP) Algorithm

Just like the algorithm's name, in BP, after an output is obtained, an *error signal* is determined, and "propagated backwards" from the output layer to the input layer. This is achieved by calculating the gradient of the error of the network regarding the network's modifiable weights [143]. This gradient is almost always used in a simple stochastic gradient descent algorithm to find weights that minimize the sum squared error. Often the term "back propagation" is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent [144].

To illustrate the back propagation algorithm, Fig. 5-4 shows a block diagram with

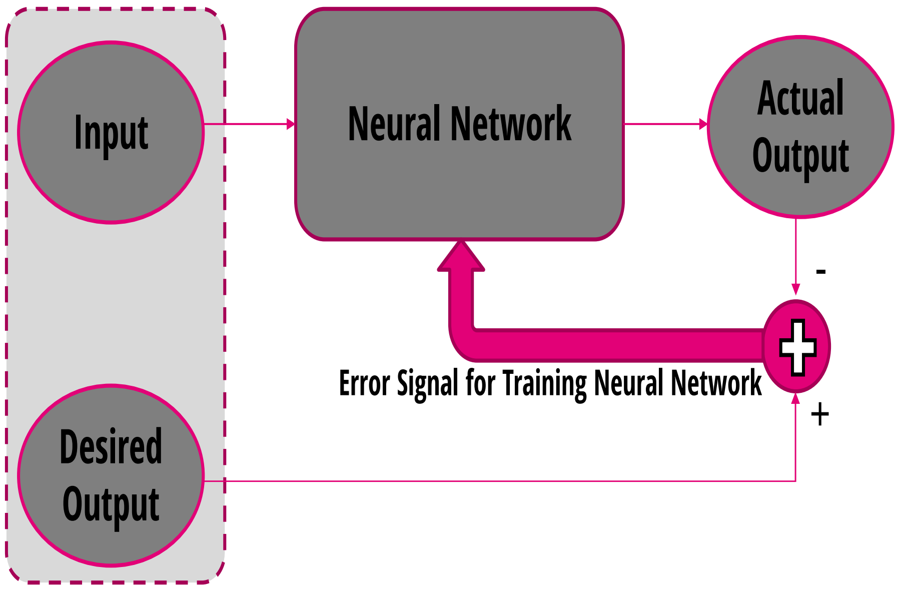


Figure 5-3: Neural Network Learning Model

the processes involved. The weights between the input layer (i) and the hidden layer (j) are represented as w_{ij} while those between the hidden layer and the output layer (k) are represented as w_{jk}. Therefore, in BP, for every input, the output is determined (feed forward step). Using the target output, an error is then determined. This error is then propagated backwards (error back propagation), first adjusting the weights between the output layer and hidden layer, w_{jk}, and then those between the hidden layer and the output layer, w_{ij}. In fact, for each set of weights, what the BP algorithm determines is the required change in weights Δw. This is determined using equation (5.1)[145] for weights w_{kj} as an example.

$$\Delta w_{kj} = \alpha \delta_j y_k \tag{5.1}$$

where Δw_{kj} is the amount by which weight w_{kj} should be changed so as to reduce the error, $0 \leq \alpha \leq 1$ is referred to as *learning rate*, and it determines how fast learning occurs. y_k is the output of the node in layer k. δ_j represents the product of the error with the derivative of the activation function [145]. Algorithm 5 shows the back propagation procedure in full.

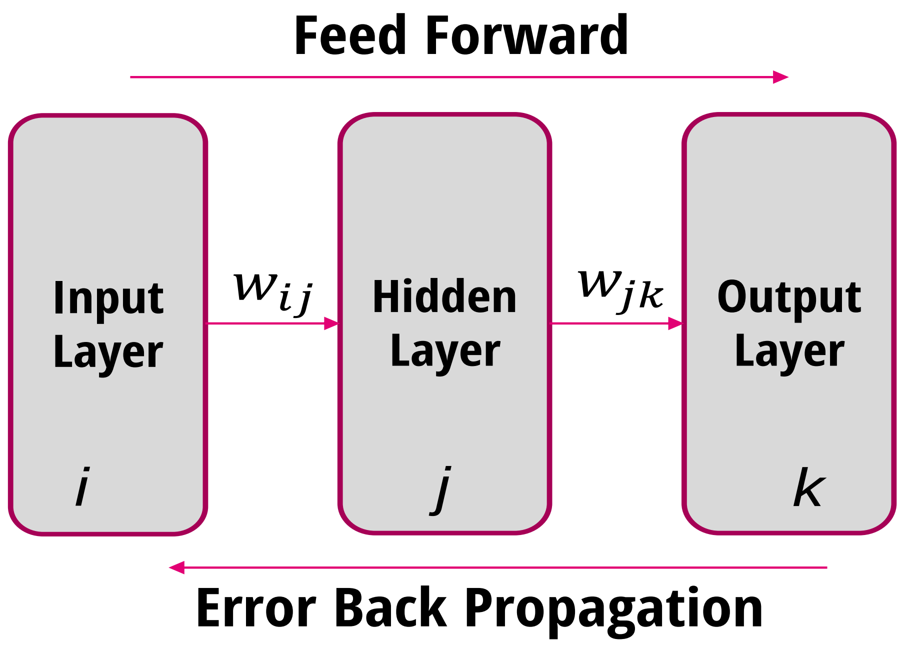


Figure 5-4: Neural Network Back Propagation

5.3 Proposed DRA Model

The system model used for our proposal is shown in Fig. 5-5. As can be observed from the figure, there are three main components: the multi-agent system representing the substrate network, the ANN that represents the internal components of each agent, and the evaluative feedback block that produces the error signal. In the following subsections, each of these elements of the model is detailed.

5.3.1 Multi-Agent System

The multi-agent system consists of all the agents that represent the SN. Specifically, each substrate node and link is represented by a node agent $n_a \in \mathcal{N}_a$ and a link agent $l_a \in \mathcal{L}_a$, where $\mathcal{N}_a$ and $\mathcal{L}_a$ are the sets of node agents and link agents respectively. The node agents manage node queue sizes while the link agents manage link bandwidths. The agents dynamically adjust the resources allocated to virtual nodes and links, ensuring that resources are not left under-utilised, and that enough resources are available to meet VN requirements. As shown in Fig. 5-5, a given agent receives as input the state, s of the substrate node it manages, and outputs an action, a.

Algorithm 5 *BackPropagation*

```
1: Initialize network weights (often with small random values)
2: repeat
3:    for Each training example do
4:       Feed Forward: Determine actual output
5:       Get desired output
6:       Determine error (i.e error = desired - actual) at the output layer
7:       Error back propagation: Compute $\Delta w_{jk}$ for all weights from hidden layer to
         output layer
8:       Error back propagation: Compute $\Delta w_{ij}$ for all weights from input layer to
         hidden layer
9:       update network weights
10:   end for
11: until The errors are acceptable
```

5.3.2 Artificial Neural Network

Our proposal uses a 3-layer ANN. An important design issue of any ANN is determining network topology, i.e. number of neurons in each of the network layers.

Input Layer

We model the state of any virtual resource (node queue size or link bandwidth) v hosted on a substrate resource z, by a 3-tuple, $s = \left(R_a^v,\ R_u^v,\ R_u^z\right)$, where R_a^v is the percentage of the virtual resource demand currently allocated to it, R_u^v is the percentage of allocated resources currently unused, and R_u^z is the percentage of total substrate resources currently unused. Therefore, the input layer consists of 3 neurons, one for each of the variables R_a^v, R_u^v and R_u^z.

Output Layer

During each learning episode, an agent should perceive the state s and give an output. This output, a, is a scalar that indicates which action should be taken to change the resource allocation for the virtual resource v under consideration. The action may be aimed at increasing (if it is positive) or reducing the resources allocated to any virtual node or link respectively. Therefore, the output layer consists of 1 neuron, representing the action, a. To illustrate the effect of an action, if a given

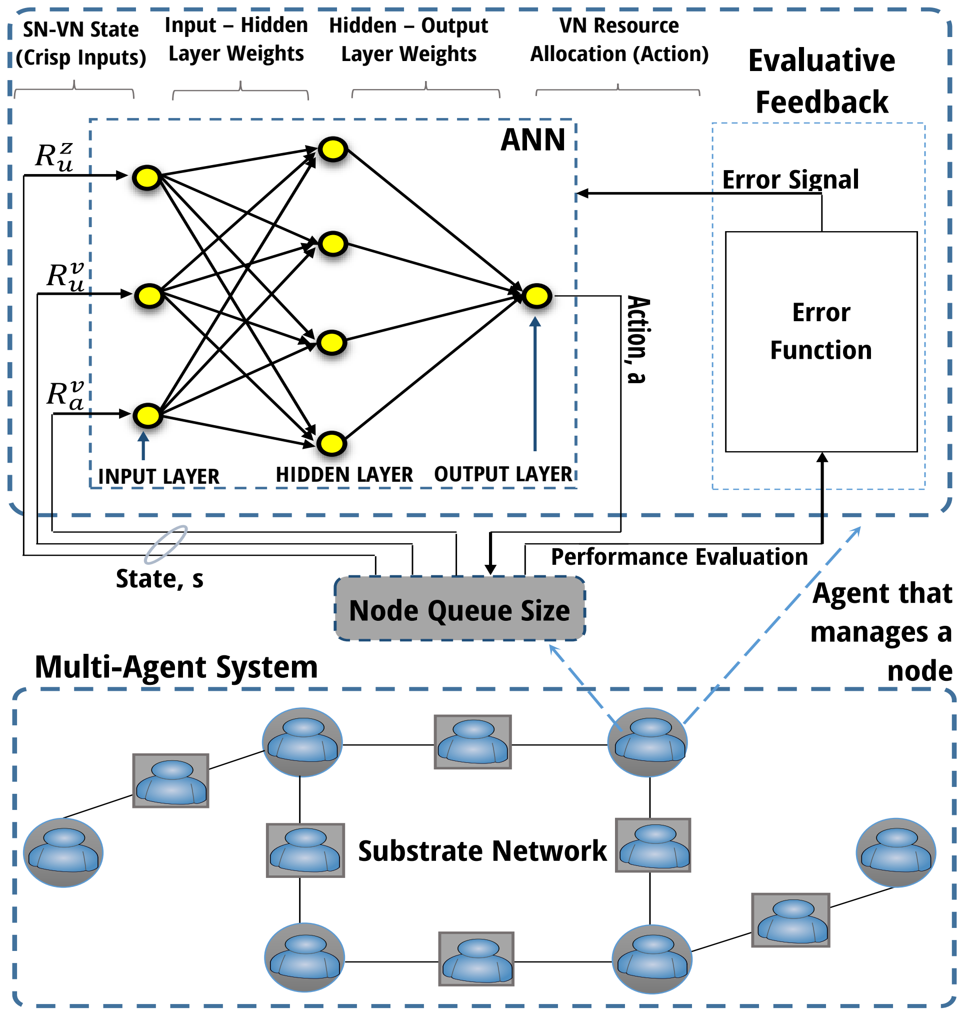


Figure 5-5: Artificial Neural Network-based Resource Allocation Model

virtual resource v has total allocated resources v_r and the agent action is a (where $-1 \leq a \leq +1$), then the resulting resource allocation is: $v_r = v_r + a \times v_t$, where v_t is the total initial demand of the virtual resource (as specified in the VN request before the VNE). It is worth mentioning that the agent only takes a given action if it does not violate resource allocation requirements, for example, at any point, $v_r \geq 0$ and $v_r \leq v_t$.

Hidden Layer

The optimal number of neurons in a hidden layer of any ANN is problem specific, and is still an open research question [137]. In this Chapter, we determine this number by

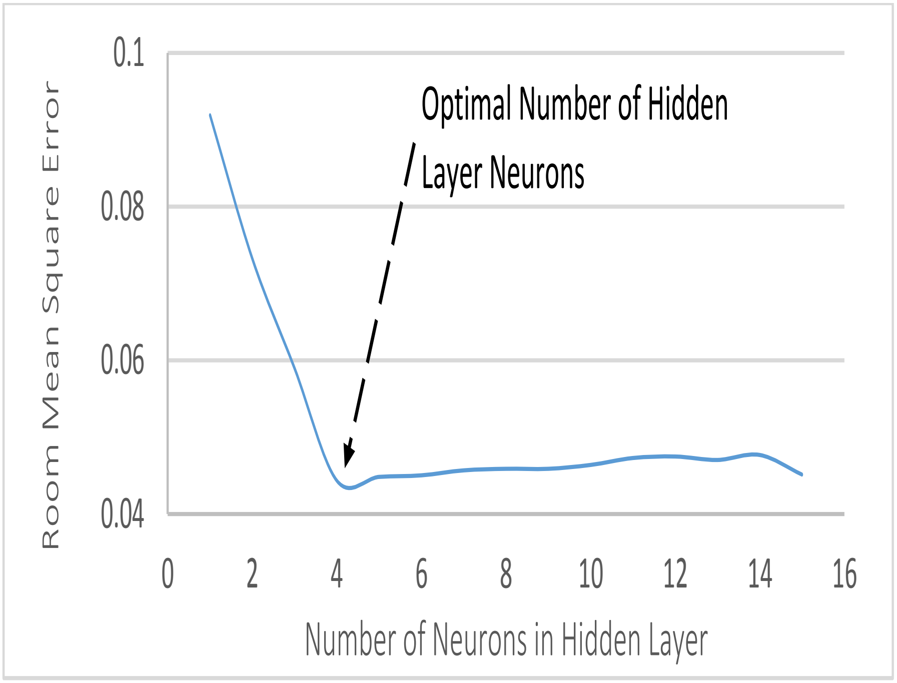


Figure 5-6: Variation of RMSE with Number of Neurons in Hidden Layer

experimentation. We perform a search from number of hidden layer neurons, $N_{HL} = 1$ to 15.

In order to achieve this, we need a test dataset. The dataset used for this purpose was saved from the q-table of a RL approach proposed in [146]. This q-table was a result of a learning system for a similar DRA task and it gives the state-action-values for the learning task. This dataset contains 512 entries, each showing the best action value in each of the possible 512 states.

With the above training set, a $10-$fold cross-validation was performed in Weka 3.6 [147], using the default parameters (learning rate, validation threshold, momentum, etc.) for the multilayer perceptron in Weka. Fig. 5-6 shows the average root mean square error (RMSE) values for 20 experiments. From the figure, the optimal number of neurons for the hidden layer is 4. The reason for choosing to use 4 neurons is not only to allow for a low RMSE, but also to avoid a possibility of over-fitting which could be caused by a network with too many neurons. This experimentation also provides initial weights that are used to initialise the neural network, and hence avoid the slow learning characteristics (and hence slow convergence) of BP.

ANN Learning

Since for the network virtualisation problem under consideration it is not possible to have a target output for each input, the training of the ANN is achieved by determining the error using a reinforcement learning-like approach. Therefore, our approach combines both ANNs and RL. One of the most remarkable achievements in the combination of ANNs and RL in machine learning research was its implementation as a backgammon player [148]. In addition, other combinations of ANNs and RL have been applied to many problems such as [149, 150, 140]. In these proposals, ANNs are used as function approximators for the RL policy. The proposal in this thesis differs from these works on two fronts: (1) we use RL to train the ANN rather than using ANNs to approximate the RL policy. This, remarkably, allows us the possibility to do away with the need for training examples and/or target outputs usually needed for learning in neural networks[2], and (2) we apply the combination ANN and RL to a network virtualisation environment.

5.3.3 Evaluative Feedback

After each learning episode, the affected substrate and virtual nodes/links are monitored, taking note of average utilisation of substrate resources, the delay on virtual links and packets dropped by virtual nodes due to buffer overflows. These values are fed back to the agent in form of a *performance evaluation*, used by the error function to produce an error signal, which is used by the BP algorithm to adjust the weights of the ANN, and hence improve future actions.

Error Function

The error $e(v)$, is an indication of the deviation of the agent's actual , from a target action. The objective of the error function is to encourage high virtual resource

[2]While we still use some training examples in our proposal (see Section 5.3.2), it is only aimed at guiding in the ANN structure design as well as ensuring a faster convergence of the algorithm (through problem specific weight initialisation) rather than as a requirement as would have been in a typical ANN learning scenario.

utilisation while punishing $n_a \in \mathcal{N}_a$ for dropping packets and $l_a \in \mathcal{L}_a$ for having high delays. Good actions by an agent are characterised by an $e(v)$ equal or close to 0, while any deviations indicate undesirable actions. Therefore, the value of $e(v)$ gives the degree of desirability or undesirability the agent's action, and is dependent on resources allocated to the virtual resources, unutilised resources, link delay in case of $l_a \in \mathcal{L}_a$ and the number of dropped packets in the case of $n_a \in \mathcal{N}_a$. The proposed error function is shown in (6.9).

$$e(v) = \begin{cases} \left(R_u^v + \alpha P_v\right) & \forall n_a \in \mathcal{N}_a \quad (5.2a) \\ \left(R_u^v + \beta D_v\right) & \forall l_a \in \mathcal{L}_a \quad (5.2b) \end{cases}$$

where α and β are constants aimed at ensuring that the magnitudes of the two terms in each of (5.2a) and (5.2b) are comparable. The values $\alpha = 0.05$ and $\beta = 40$ used in this Chapter, were determined by simulations. For example, looking at Fig. 5-9 shows that the maximum value of P_v is about 20. Therefore, to make these values comparable to $0 \leq R_u^v \leq 1$, we divide each value by 20 (multiply it by $\alpha = 0.05$). P_v is the number of packets dropped by node $n_a \in \mathcal{N}_a$ from the time the allocation action was taken, and D_v is the *extra* delay encountered by a packet using $l_a \in \mathcal{L}_a$. The extra delay is calculated as the difference between actual delay and the theoretical delay. We define theoretical delay as the delay the virtual link would have if it was allocated 100% of its bandwidth demand[3]. The actual delay is determined as the difference between when a packet is received at one end of the link, to when it is delivered to the other end. Once the error is determined, the ANN weights are adjusted using BP.

[3]The simulations in this Chapter determine this value from a parallel simulation using a virtual network with 100% resource allocation.

Table 5.1: SN and VN Properties

Parameter	**Substrate Network**	**Virtual Network**
Minimum Number of Nodes	25	5
Maximum Number of Nodes	35	15
Minimum Node Queue Size	(100×1518) Bytes	(10×1518) Bytes
Maximum Node Queue Size	(200×1518) Bytes	(20×1518) Bytes
Minimum Link Bandwidth	2.0Mbps	1.0Mbps
Maximum Link Bandwidth	10.0Mbps	2.0Mbps

5.4 Performance Evaluation

5.4.1 Simulation Environment

To evaluate our proposal, a simulation scenario similar to Fig. 4-9 was setup. SN and VN topologies were generated using Brite [110] with settings shown in Table 4.4. Thereafter, VN requests arrive, one at a time to the SN. Whenever a VN request is accepted by the SN, the VN topology is created in NS3 [133] using the network virtualisation module and real traffic traces explained in Section 4.5. Simulations were run on an Ubuntu 12.04 LTS Virtual Machine with 4.00GB RAM and 3.00GHz CPU specifications.

5.4.2 Simulation Parameters

Both substrate and virtual networks were generated on a 250×250 grid. The queue size and bandwidth capacities of substrate nodes and links as well as the demands of virtual networks are all uniformly distributed between minimum and maximum values shown in Table 5.1. Link delays are as determined by Brite. Each virtual node is allowed to be located within a uniformly distributed distance $7.5 \leq x \leq 15$ of its requested location, measured in grid units. We assumed that VN requests arrive following a Poisson distribution with an average rate of 1 per minute. The average service time of each VN is 60 minutes and is assumed to follow a negative exponential

Table 5.2: Compared Algorithms

Code	**Resource Allocation Approach**
D-ANN	Dynamic, based on Artificial Neural Networks [Our Contribution]
D-RL	Dynamic, based on Reinforcement Learning[146]
S-CNMMCF	Static, Coordinated Node Mapping and MCF for link mapping[8]
S-OS	Static, link based optimal one shot Virtual Network Embedding[146]

distribution. The ANN algorithm runs every minute[4].

5.4.3 Compared Algorithms

We compare the performance of our proposal with 3 representative state-of-art solutions. The first is a DRA approach that uses RL [146] (as proposed in Chapter 4); the second performs a coordinated node and link mapping [8]; and the third is also a static baseline formulation that performs a one shot mapping, and also used in performance evaluations in [146]. The solution in [8] was adapted to fit into our formulation of the problem. In particular, for [8] the link delay requirements were neglected at the embedding stage, and for this reason, it is not used in QoS evaluations. In addition, our consideration in these simulations is for unsplittable flows. We identify and name the compared approaches in table 6.4. The mathematical programs in all proposals are solved using CPLEX 12.5 [111].

5.4.4 Performance Metrics

Embedding Quality

We define embedding quality as a measure of how efficiently the algorithm uses the SN resources for accepting VN requests. This is evaluated using the acceptance ratio and the average level of utilisation of SN resources. The acceptance ratio is a measure

[4]It is worth remarking that while this thesis has not studied the effect of the frequency of running the algorithm, we expect that a lower running frequency would make the dynamic allocation become comparable to the static one, while a higher frequency might negatively impact system stability.

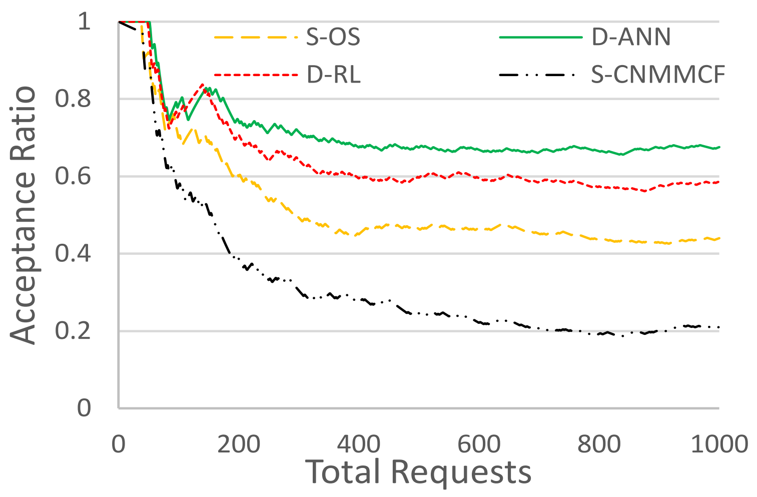


Figure 5-7: Acceptance Ratio

of the long term number of VN requests that are accepted by the substrate network. The average level of utilisation of substrate resources is a measure of how efficiently the SN resources are used.

Quality of Service

We use both packet drop a well as delay variation as indications of the quality of service. As shown in Table 4.3, we model the networks to drop packets due to both node buffer overflow as well as packet errors. We determine the packet delay variation as the difference in delays encountered by packets transmitted over the network over two successive time intervals, while packet drop is the number of packets dropped by a given VN during a given time interval. For these evaluations, the time interval used to update these measurements corresponds to the transmission of every 100 packets.

5.4.5 Discussion of Results

The simulation results are shown in Figs. 5-7 – 5-10. As can be seen from Fig. 5-7, while both dynamic approaches perform better than the static ones in terms of VN acceptance ratio, the ANN approach outperforms all three. The reason for the dynamic approaches performing better than the static ones is that in former cases, the

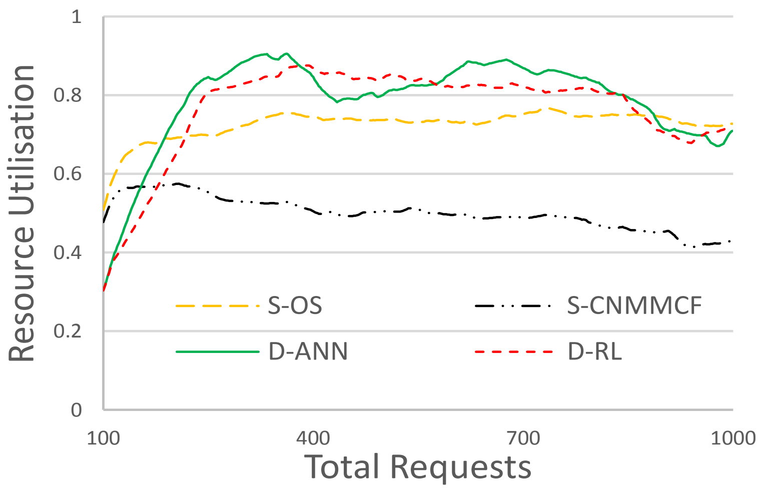


Figure 5-8: Resource Utilisation

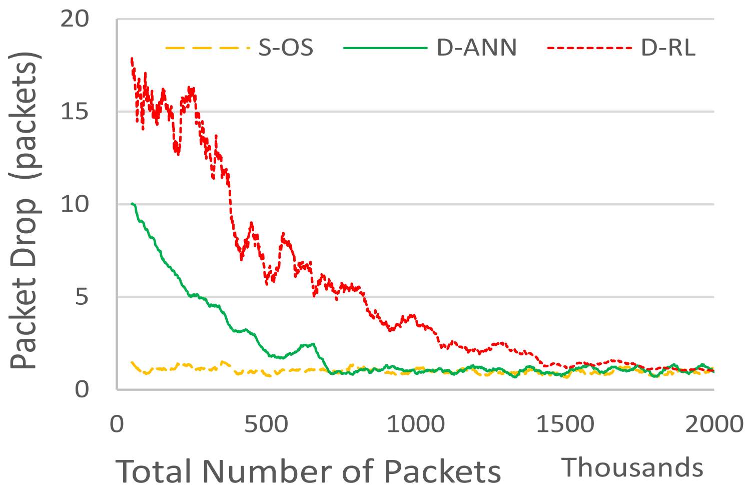


Figure 5-9: Packet Drop Ratio

substrate network always has more available resources than in the later case, which is a direct result of allocating and reserving only the required resources for the virtual networks. The fact that ANN outperforms the RL approach can be attributed to the fact that the ANN approach models the states and actions with better granularity i.e. without restricting the states and actions to few discrete levels. We also note that S-OS has a better acceptance ratio than S-CNMMCF. This is due to the fact that since S-CNMMCF performs node and link mapping in two separate steps, link mappings could fail due to locations of already mapped nodes.

Fig. 5-8 shows the average utilisation of SN resources. It can be observed that

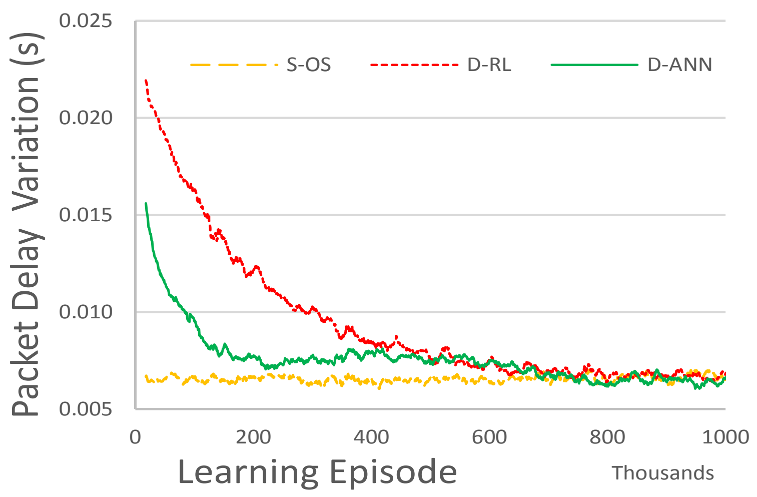


Figure 5-10: Delay Variation

except for S-CNMMCF, the other three approaches on average use the same amount of SN resources. The fact that S-CNMMCF has a lower resource utilisation is expected as a result of having slightly more resource requests rejected either due to a node mapping that makes link mapping impossible, or for previous link mappings using more resources. The fact that S-OS, D-RL and D-ANN all have on average the same resource utilisation profile is mainly due to all of them having the same initial mapping algorithm (which is S-OS). It can however be noted that while S-OS, D-RL and D-ANN all have similar resource utilisation levels, D-ANN uses these resources to achieve a higher acceptance of VNs, which further confirms the extra resource allocation efficiency of the ANN approach.

Fig. 5-9 shows that S-OS has an almost constant packet drop ratio variation while that for D-RL and D-ANN is initially high, but gradually converges to that of S-OS. At the beginning of the learning processes, the dynamic approaches vary the queue sizes quite considerably leading to more packet drops. The fact that D-ANN has a lower packet drop ratio than D-RL over the learning period can be explained since D-ANN has better granularity in perceiving the state of resources and allocation. We also note that the initial drop ratio of D-ANN is lower than that of D-RL which can be attributed to the weight initialisation obtained from Weka (See Section 5.3.2).

Finally, Fig. 5-10 shows that packet delay variations for the two dynamic approaches is initially higher but reduces over the learning period. Once more, these differences are attributed to the initial learning period, and the difference between D-RL and D-ANN is due to better options in perception and action for D-ANN, as well as the weight initialisation in D-ANN.

5.5 Conclusion

This Chapter has proposed an improved distributed and dynamic approach for allocation of resources in virtual networks, that opportunistically takes advantage of unused virtual network resources by allocating them to other virtual networks that need them, while ensuring that at any point, if needed, all virtual networks can have their originally assigned resources.

The substrate network was represented by a set of distributed 3-layer artificial neural networks, which were required to dynamically perceive continuous substrate and virtual network resource allocation and utilisation statuses, and perform continuous actions to adjust the resources allocated to virtual nodes and links. An evaluative feedback mechanism was proposed to train the neural networks to limit amount of idle resources in virtual networks, while being aware of their QoS requirements. We also performed experimentation in Weka, using the policy table of Chapter 4 to determine the optimal number of neurons in the hidden layer.

Through simulation, we have been able to show that our proposal improves the acceptance ratio of virtual networks by about 10% compared to a reinforcement learning based solution, and about 50% over a static resource allocation approach. We have also confirmed that this performance improvement can be achieved without incurring a penalty on the quality of service requirements of the virtual networks, especially after the neural networks have learned their weights. In addition, initial weight initialisation proposed in this Chapter improves the speed of convergence[5] by about 50% compared to the reinforcement learning approach.

[5]Measured from the rate at which the packet drops and delay in the dynamic approach converge to that of the static approach.

However, while the neural network system proposed in this Chapter improves performance, there are still more possibilities for improvement. In particular, in this Chapter, the neural network only learns the link weights, always maintaining the same network structure. It is also worth noting that a neural network approach requires specific modelling at the beginning, for example, to determine the number of neurons in the hidden layer, for which there is no defined way of optimisation of the network structure. The next Chapter builds on the advantages of the proposal in this Chapter, by adding a fuzzy system that allows a purely dynamic network structure.

Chapter 6

Neuro Fuzzy System-based Dynamic Resource Allocation

6.1 Introduction

In Chapter 5, a dynamic resource allocation system based on neural networks was proposed. However, while neural networks are important for their learning and generalisation capabilities, they do not have a clearly defined way on how the number of layers as well as the number of neurons in its layers are determined. In addition, due to the difficulties in modelling and interpreting a neural network structure, it is difficult to have a dynamically changing neural network structure, which would be important especially given the dynamic nature of resources in network virtualisation environments.

This Chapter extends the previous one by proposing an adaptive, hybrid neuro-fuzzy system (NFS)[17] which learns by use of a reinforcement learning-like reward to achieve dynamic resource allocation in virtualised networks. The NFS is composed of a set of rules that define the objectives of the system. Thereafter, the system does not only dynamically adjust the network weights (fuzzy weights), but also adjusts the network structure by adding or removing links (in form of rules).

We represent the substrate network by multiple neuro-fuzzy agents, each of which is tasked to dynamically adapt its knowledge base so as to efficiently utilise the resources of the substrate network. To this end, we start by creating an initial knowledge base for each agent using supervised learning. This is achieved by defining the maximum possible rule base, biasing it by use of expert knowledge, and then pruning it using examples from a training data set. We then use an unsupervised, RL-based evaluative feedback mechanism to continuously improve the knowledge base by ensuring that the agents learn from their actions. In addition, the agents cooperate so as to prevent conflicting actions and do share their knowledge so as to enhance their respective performances, and ensure faster convergence to optimal actions.

The rest of the Chapter is organised as follows: Sections 6.2 and 6.2.1 briefly introduce fuzzy systems and neuro-fuzzy systems respectively, while the proposed NFS is described in Section 6.3. The proposed evaluative feedback and rule base initialisation approaches are presented in Sections 6.4 and 6.5 respectively, while we define an agent cooperation scheme in Section 6.6. The evaluation of performance is given in Section 6.7, and the Chapter is concluded in Section 6.8.

6.2 Fuzzy Systems (FS)

FSs are rule-based expert systems, which use fuzzy rules and fuzzy inference [151]. As shown in Fig. 6-1, fuzzy systems are usually made up of three functional blocks: fuzzification, knowledge base and inference, and defuzzification. At the fuzzification step, the values of numerical inputs are compared with membership functions (MFs) so as to determine their degree of membership in the respective fuzzy sets[1]. The knowledge base and inference block contains the fuzzy rules (rule base), a database of fuzzy sets (represented by membership functions) used to represent the fuzzy rules, and a decision making unit which performs inference on the fuzzy rules. A fuzzy rule represents knowledge and skills, and is of the form:

$$\text{IF} < \textit{Antecedent} > \text{THEN} < \textit{Consequent} > \tag{6.1}$$

[1]While we have done our best to make this thesis self-contained, a reader who is not familiar with such terms as fuzzy variable, fuzzy set, fuzzy logic, fuzzy rule, t-norm, t-conorm, membership functions is referred to [151] for an introduction.

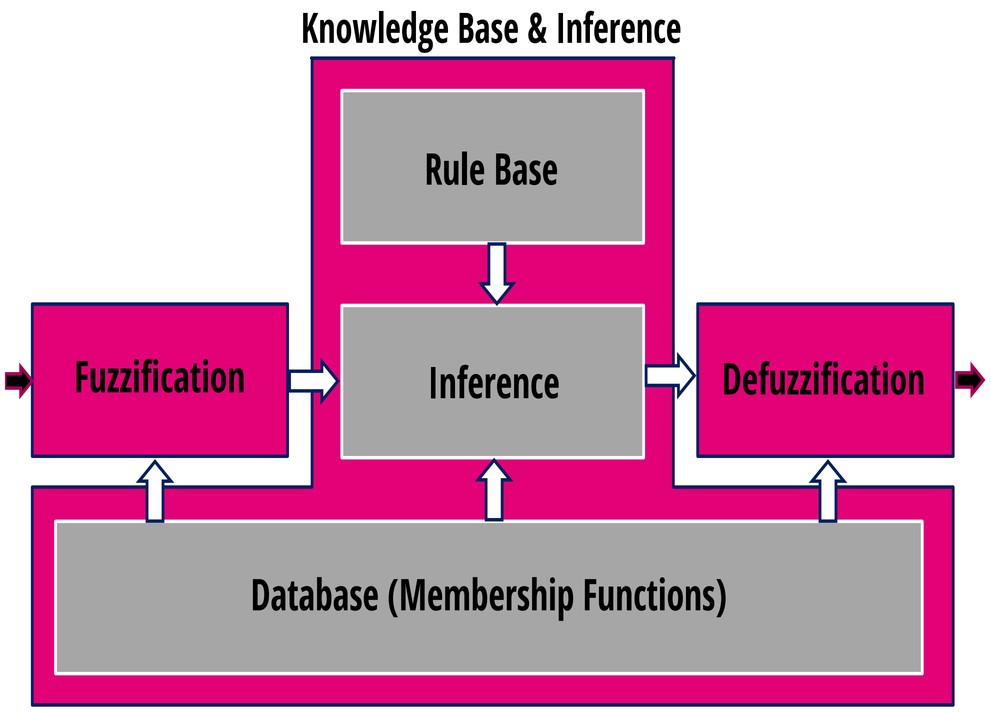


Figure 6-1: Functional components of a Fuzzy System

An example of an antecedent could be *altitude* **is** *high* while that of a consequent could be *temperature* **is** *low*. In this case, *altitude* and *temperature* are fuzzy variables, while *high* and *low* are fuzzy sets and are usually represented by membership functions such as triangular, trapezoidal or monotonic. Membership functions therefore give a mapping of the fuzzy sets to real numbers $\mathbb{R} \to [0,1]$. An example of a fuzzy rule R_i in the form of (6.1) is shown in (6.2).

$$\textbf{IF} \;\; x_1 \;\text{ is }\; \mu_i^1 \;\; \textbf{AND} \ldots \textbf{AND} \;\; x_m \;\text{ is }\; \mu_i^m \;\; \textbf{THEN} \;\; a \;\; is \;\; \lambda_i \tag{6.2}$$

where $x_1, ..., x_m \in \mathbb{R}$ are input variables and $\mu_i^m \in \mathbb{R} \to [0,1]$ are their possible fuzzy sets, while $a \in \mathbb{R}$ is an output variable and $\lambda_i \in \mathbb{R} \to [0,1]$ are its possible fuzzy sets. In general, it is usually required to evaluate the antecedent of each rule and thereafter to combine more than one rule to obtain a final result. This is known as *inference*. Inference takes two steps: evaluating the antecedent of each rule (the set of **AND**s) so as to determine its consequent, and a way of aggregating the consequents of different rules to form a final result. The **AND** in the antecedents of each rule is usually evaluated by a $t - norm$, usually the minimum operation [17]. Then, the different rules are combined using the maximum operation. Therefore, the complete inference process involves the $max - min$ operation shown in (6.3), and leads to an output fuzzy set $\lambda(a)$.

$$\lambda(a) = \max_{R_i}\left(min\Big(\mu_i^1(x_1), ..., \mu_i^m(x_m), \lambda_i(a)\Big)\right) \tag{6.3}$$

Finally, in the deffuzification function, the fuzzy set $\lambda(a)$ resulting from the inference of rules is transformed back into a crisp value. In general, there are many methods for performing deffuzification. Most fuzzy systems use the centre of gravity method for this purpose [151].

6.2.1 Neuro-Fuzzy System (NFS)

Since fuzzy systems are created from explicit knowledge in form of rules and membership functions (MFs), applying them in dynamic systems requires a way of automatically tuning these parameters (rules and MFs), or even changing the structure of the system. One of the most popular ways of dynamically tuning these parameters and/or adapting the structure of fuzzy systems is by use of neural networks. A combination of neural networks and fuzzy systems leads to NFSs.

The key advantage of neuro-fuzzy approach over traditional ones (i.e individual neural or fuzzy approaches) lies in the fact that the former does not require a mathematical description of the system while modeling [152]. Moreover, in contrast to pure neural or fuzzy methods, the neural fuzzy method possesses both of their advantages; it brings the low-level learning and computational power of neural networks into fuzzy systems and provides the high-level human-like thinking and reasoning of fuzzy systems into neural networks [153, 154, 155]. Therefore, neural networks and fuzzy systems can be combined to benefit from both their advantages while solving their individual weaknesses. In particular, neural networks introduce its computational characteristics of learning in the fuzzy systems and receive from them the interpretation and clarity of systems representation. Thus, the disadvantages of the fuzzy systems are compensated by the capacities of the neural networks. These techniques

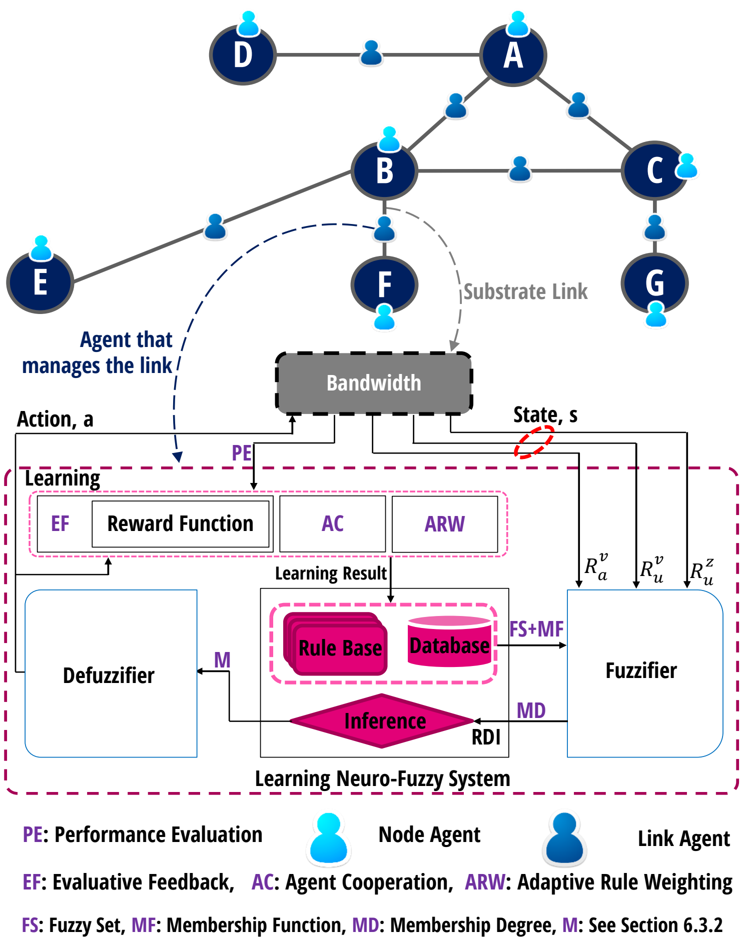

D
A
B
C
E
F
G
Substrate Link
Agent that manages the link
Bandwidth
Action, a
State, s
PE
Learning
EF
Reward Function
AC
ARW
Learning Result
R_a^v
R_u^v
R_u^z
Rule Base
Database
FS+MF
Defuzzifier
M
Fuzzifier
Inference
MD
RDI
Learning Neuro-Fuzzy System
PE: Performance Evaluation
Node Agent
Link Agent
EF: Evaluative Feedback, AC: Agent Cooperation, ARW: Adaptive Rule Weighting
FS: Fuzzy Set, MF: Membership Function, MD: Membership Degree, M: See Section 6.3.2

Figure 6-2: Learning Neuro-Fuzzy System VN Resource Allocation Model

are complementary, which justifies its use together [156].

A NFS can be viewed as a special 3-layer feedforward neural network in which the neurons use t-norms and t-conorms instead of the usual neural network activation functions [17]. The first layer represents the input variables, the hidden layer represents the fuzzy rules and the third layer represents output variables. The fuzzy sets are encoded as (fuzzy) connection weights. The knowledge and hence accuracy of the network is determined by its structure (the different connections), as well as the fuzzy weights on these connections. We model the interactions between the different components of the designed NFS, as well as with the multi-agent system for the application to dynamic virtual network resource management in 6.3.

6.3 Proposed NFS-based DRA Model

As mentioned in Section 5.2, VNE approaches allocate resources to each virtual node and link as specified in VN requests. In static allocation schemes, the amount of allocated resources is kept fixed irrespective of actual utilisation, which leads to inefficient utilisation of substrate network resources [146]. Our approach dynamically adjusts the resources allocated to each virtual node and link using a neuro-fuzzy system. The overall system model used for this purpose is shown in Fig. 6-2. As can be observed from the model, we highlight the multi-agent system representing the substrate network and the learning neuro-fuzzy system that represents the internal components of each agent. We further split the learning neuro-fuzzy system into four functional modules: fuzzifier, RDI (rule base, database, inference), defuzzifier, and learning (EF, AC, ARW). In the following subsections, each of these elements of the model and their respective interactions is detailed.

6.3.1 Multi-agent Environment

The multi-agent environment[2] consists of all the agents that represent the substrate network. Specifically, each substrate node and link is represented by a node agent $n_a \in \mathcal{N}_a$ and a link agent $l_a \in \mathcal{L}_a$, where $\mathcal{N}_a$ and $\mathcal{L}_a$ are the sets of node agents and link agents respectively. The node agents manage node queue sizes while the link agents manage link bandwidths. The agents dynamically adjust the resources allocated to virtual nodes and links, ensuring that resources are not left under-utilised, and that enough resources are available to serve user requests. As shown in Fig. 6-2, a given link agent receives as input the state of the substrate link it manages.

The state of any resource is a vector $\mathbf{s}$ with each term $s \in \mathcal{S}$ representing the state of one of the virtual links/nodes mapped onto it. More specifically, this state, for any given virtual resource v hosted on a substrate resource z, is represented by a 3-tuple, $s = (R_a^v,\ R_u^v,\ R_u^z)$, where R_a^v is the percentage of the virtual resource demand currently

[2]The multi-agent environment, the modelling of the states and actions in this subsection are all similar to the ones in Chapters 4 and 5. We re-state them here only for completeness.

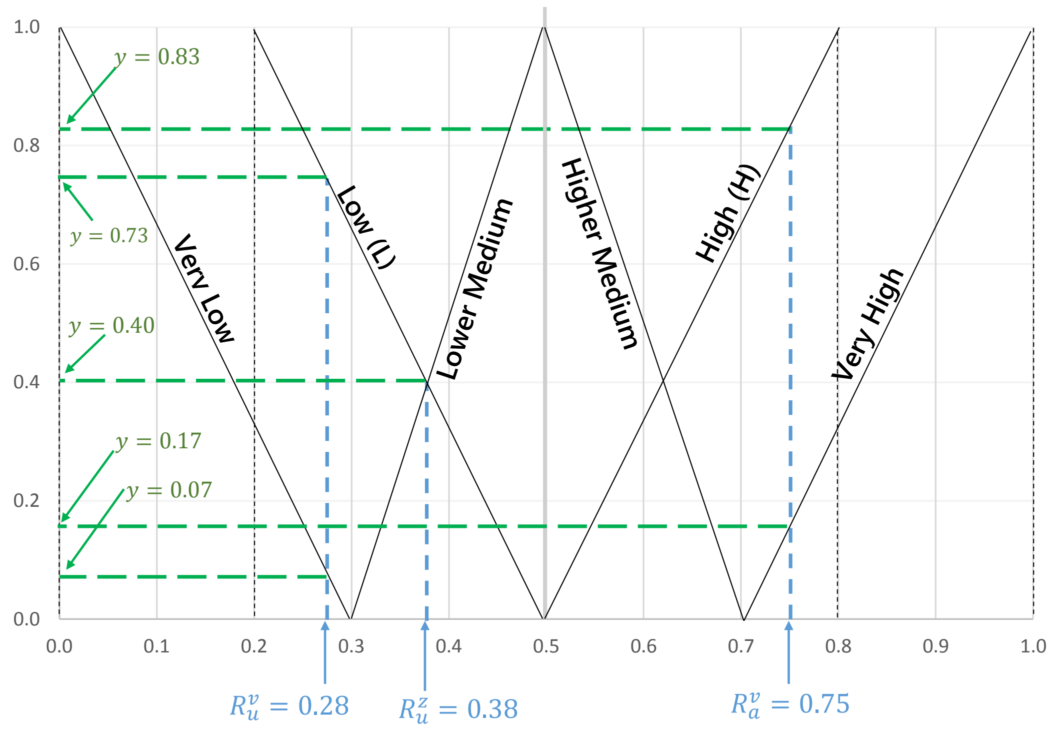


Figure 6-3: Monotonic MFs for input (state) fuzzy sets

allocated to it, R_u^v is the percentage of allocated resources currently unused, and R_u^z is the percentage of total substrate resources currently unused. While in a given state, s, the agent gives as output, an action. The action of each agent is a vector $\mathbf{A}$, where each term $a \in \mathcal{A}$ indicates which action should be taken to change the resources for each of the virtual node/link mapped onto it. The action may be aimed at increasing or reducing the resources (queue size or bandwidth) allocated to any virtual node or link respectively. Each element in $\mathbf{A}$ corresponds to a unique element in $\mathbf{S}$. Based on how well a given action drives the objectives of the system, the agent gets a performance evaluation (PE), which is used to learn so as to improve future actions. We consider that each $n_a \in \mathcal{N}_a$ has information about the substrate node resource availability as well as the resource allocation and utilisation of all virtual nodes mapped onto the substrate node. In the same way, we expect that each $l_a \in \mathcal{L}_a$ has information about substrate link bandwidth as well as the allocation and utilisation of these resources by all virtual links mapped to it.

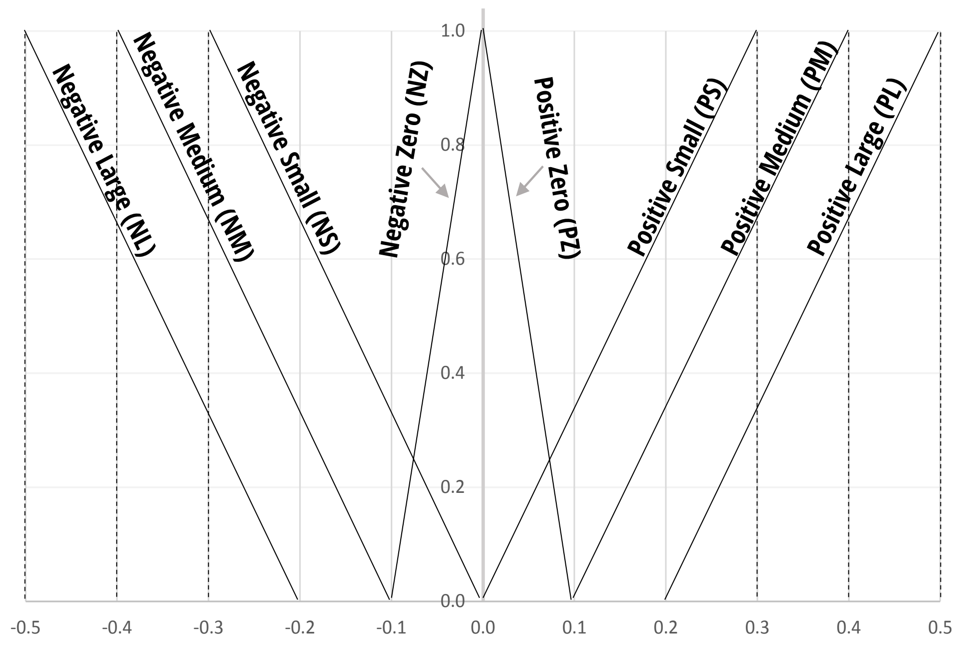


Figure 6-4: Monotonic MFs for output (action) fuzzy sets

6.3.2 Learning Neuro-Fuzzy System (NFS)

Database

The database is a definition of the fuzzy sets (FS) and the membership functions (MF) that represent them, and is used in the creation of fuzzy rules. We have defined six fuzzy sets into which the input variables can fall. These are: very low (VL), low (L), lower medium (LM), higher medium (HM), high (H) and very high (VH). These fuzzy sets are represented by the monotonic membership functions in Fig. 6-3. Each membership function $y = \mu(x)$ in Fig. 6-3 is characterised by two parameters p and q such that $\mu(p) = 0$ and $\mu(q) = 1$ as defined in equation (6.4). The MFs are used to determine a value or degree of membership, which quantifies the grade of membership of a given variable to the fuzzy sets.

$$y = \mu(x) = \begin{cases} \dfrac{p-x}{p-q} & \text{if } \Big((p \leq q) \wedge (x \in [p,q])\Big) \qquad (6.4a) \\ \dfrac{p-x}{p-q} & \text{if } \Big((p > q) \wedge (x \in [q,p])\Big) \qquad (6.4b) \\ 0 & \textit{otherwise} \qquad (6.4c) \end{cases}$$

In the same way, we have defined eight fuzzy sets for the output variable. These are: negative large (NL), negative medium (NM), negative small (NS), negative zero (NZ), positive zero (PZ), positive small (PS), positive medium (PM) and positive large (PL). These fuzzy sets are represented by the monotonic membership functions in Fig. 6-4, and have similar definitions as in (6.4).

Rule Base

The rule base consists of the rules that are used by the agents to take actions while in given states. These rules are based on the input and output fuzzy sets defined in Figs. 6-3 and 6-4. As an example, (6.5) presents four possible rules that could be formulated for the system under consideration.

$$R_1: \quad \textbf{if } R_a^v \text{ is VH } \textbf{and } R_u^v \text{ is L } \textbf{and } R_u^z \text{ is LM } \textbf{then } O \text{ is PZ} \tag{6.5a}$$

$$R_2: \quad \textbf{if } R_a^v \text{ is L } \textbf{and } R_u^v \text{ is H } \textbf{and } R_u^z \text{ is HM } \textbf{then } O \text{ is NZ} \tag{6.5b}$$

$$R_3: \quad \textbf{if } R_a^v \text{ is H } \textbf{and } R_u^v \text{ is L } \textbf{and } R_u^z \text{ is L } \textbf{then } O \text{ is PS} \tag{6.5c}$$

$$R_4: \quad \textbf{if } R_u^v \text{ is VL } \textbf{then } O \text{ is PS} \tag{6.5d}$$

In words, rule (6.5a) states that: if the resource allocation to the virtual node/link v is very high AND a low percentage of these resources are unused AND the substrate node/link z onto which it is embedded has a low−medium percentage of free resources, then the action should be to increase the amount of resources allocated to v by an amount determined by *positive zero*. In general, each agent will have more than one rule at any given time. Therefore, since we have 3 input variables each with 6 possible fuzzy sets and 1 output variable with 8 possible fuzzy sets, the maximum number of rules we can have in the system is $6 \times 6 \times 6 \times 8 = 1728$. Initialising a system with this number of rules would take a long time before the NFS has reduced the rules to the *necessary* ones[3]. In this thesis, we propose supervised learning-based rule initialisation scheme that is aimed at enhancing the learning speed of the agents. This scheme is presented in Section 6.5.

Fuzzifier

Given a state s represented by the real-valued variables R_a^v, R_u^v and R_u^z, a fuzzifier determines the set of rules $R' \subseteq R$ that are satisfied (fired), where R is the rule base. Then, for each satisfied rule, fuzzification involves determining the membership degree (MD) of each input variable in the respective fuzzy set. This is the result of the function $\mu(x)$ for a particular input variable, rule and membership function, and is is calculated using equations (6.4). To illustrate this, consider that rule (6.5a) has been satisfied by a given input state. Then, the fuzzifier would have to determine the membership degrees $\mu_{5a}^{\mathrm{VH}}(R_a^v)$, $\mu_{5a}^{\mathrm{L}}(R_u^v)$ and $\mu_{5a}^{\mathrm{LM}}(R_u^z)$ to which the inputs R_a^v, R_u^v and R_u^z belong to the fuzzy sets VH, L and LM respectively. As can be noted from Fig. 6-3, a given input variable can belong to one or more MFs. For example, the variable $R_a^v = 0.75$ belongs to both H and VH with membership degrees $y = \mu_i^{\mathrm{H}}(0.75) = 0.83$ and $y = \mu_i^{\mathrm{VH}}(0.75) = 0.17$ respectively. In a similar way, given values of $R_u^v = 0.28$ and $R_u^z = 0.38$, their degrees of membership to the respective fuzzy sets can be determined as shown in Fig. 6-3.

Inference

The inference block receives, for each rule $R_i \in R'$, a membership degree $y_i^{R_a^v}$, $y_i^{R_u^v}$ and $y_i^{R_u^z}$ for each of the variables R_a^v, R_u^v and R_u^z respectively. Ideally, standard inference would involve a $max - min$ operation on these membership degrees [151]. However, due to the nature of the deffuzification function used in this Chapter (see 6.3.2), inference only involves the min operation. This operation results into the minimum membership degree $y_{min} = min\left(y_i^{R_a^v}, y_i^{R_u^v}, y_i^{R_u^z}\right)$ for each rule. Then, the inverse $\lambda^{-1}(y_{min})$

[3]We refer to rules as being necessary if they are often required by an agent, and their actions usually lead the agent to efficient resource allocations.

of each y_{min} is determined from (6.6).

$$x = \lambda^{-1}(y) = p - y(p - q) \tag{6.6}$$

(6.6) is similar to (6.4), re-arranged to make x the subject. A matrix $\mathbf{M}$ is then created with the each row containing the value y_{min} in the first column and $\lambda^{-1}(y_{min})$ in the second column. Therefore, the matrix $\mathbf{M}$ has as many rows as the number of fired rules. It is this matrix that is used at the deffuzification step.

Defuzzifier

The output of the inference block is a matrix $\mathbf{M}$ of membership degrees and their respective inverse values. The role of the defuzzifier is to convert this matrix into an output action a for the agent. In this thesis, we adopt a non-standard defuzzification approach proposed in [157], in which the crisp output a is given by (6.7).

$$a = \frac{\sum\limits_{i=1}^{n} \left(m_{i1} \times m_{i2} \right)}{\sum\limits_{i=1}^{n} m_{i1}} \tag{6.7}$$

where $n = |R'|$ is the number of fired rules, and m_{i1} and m_{i2} are the elements in the first and second columns respectively of the i^{th} row of M. The rationale behind (6.7) is to determine a weighted average of the potentially applicable actions by their corresponding membership values. It is worth noting that this simplified deffuzification process is a direct result of the monotonic membership functions used to define the output fuzzy sets. These membership functions make it unnecessary to make any translations of the inverse $\mu_{y_{min}}$ of each rule as these values are already crisp [157]. The use of monotonic membership functions for the input fuzzy sets is for simplicity.

Example: Fuzzification, Inference, Deffuzification To illustrate the processes described above with regard to the VN resource allocation problem, we revisit the four rules defined in (6.5). Consider that the state of a given substrate resource z and a virtual resource v is such that $R_a^v = 0.75$, $R_u^v = 0.28$ and $R_u^z = 0.38$ as shown in

Table 6.1: Running Example - Fuzzification

Input Variable	**Fuzzy Sets**	**Satisfied Rules**	**Membership Degrees**
$R_a^v = 0.75$	VH, H	R_1, R_3, R_4	$\mu_1^{VH}(0.75) = 0.17, \quad \mu_3^H(0.75) = 0.83$
$R_u^v = 0.28$	VL, L	R_1, R_3, R_4	$\mu_1^L(0.28) = 0.73, \quad \mu_3^L(0.28) = 0.73, \quad \mu_4^{VL}(0.28) = 0.07$
$R_u^z = 0.38$	LM, L	R_1, R_3, R_4	$\mu_1^{LM}(0.38) = 0.4, \quad \mu_3^L(0.38) = 0.4$

Fig. 6-3. In the *fuzzification* step, we note that the input variable $R_a^v = 0.75$ lies in two fuzzy sets VH and H with membership degrees $\mu^{VH}(R_a^v) = 0.17$ and $\mu^H(R_a^v) = 0.83$ respectively. In a similar way, the membership degrees of $R_u^v = 0.28$ and $R_u^z = 0.38$ to their respective fuzzy sets can be determined. We see that the rule R_2 is not satisfied (e.g. since R_a^v does not belong to the fuzzy set L) while rules R_1, R_3 and R_4 are satisfied. In Table 6.1, we show these details for each input variable. For R_1, the *inference* step is $y_{min}^1 = min(0.17, 0.73, 0.4) = 0.17$, that for R_3 is $y_{min}^3 = 0.4$, and that for R_4 is $y_{min}^4 = 0.07$. We therefore have the membership degrees for the three output fuzzy sets: PZ, PS and PS for rules R_1, R_3 and R_4 respectively. We now find the inverse for each y_{min} using (6.6). As can be seen from Fig. 6-4, for R_1, $p = 0.1$ while $q = 0.0$. Therefore, $\lambda^{-1}(y_{min}^1) = 0.1 - 0.17 \times (0.1 - 0.0) = 0.083$. In a similar way, $\lambda^{-1}(y_{min}^3) = 0.12$ and $\lambda^{-1}(y_{min}^4) = 0.021$. Finally, the input into the output layer node is a $n \times 2$ *matrix* where each row i contains the output y_{min}^i of the i^{th} rule (R_i) in the first column and its inverse $\lambda_i^{-1}(y_{min}^i)$ in the second column. For our example, the matrix, M below will be the input to the output node (deffuzifier).

$$M = \begin{bmatrix} 0.17 & 0.0830 \\ 0.40 & 0.1200 \\ 0.07 & 0.0210 \end{bmatrix}$$

The last step is *deffuzification* and uses equation (6.7) on the contents of $\mathbf{M}$ to produce the agent action. The action a in this case would be

$$\frac{(0.17 \times 0.083) + (0.4 \times 0.12) + (0.07 \times 0.021)}{0.17 + 0.4 + 0.07} = 0.11$$

This would mean that the resources allocated to the virtual resource in question has to be increased by 11% of its total demand.

Learning

After an agent takes an action, we evaluate the action so as to determine if it led to a better utilisation of substrate resources without negatively impacting the QoS requirements of the virtual node or link. Therefore, as shown in Fig. 6-2, the learning module receives as input the agent's action, and an evaluation of the performance (PE) that resulted from this action. Then, the module outputs a "learning result" that is aimed at adjusting the knowledge base, and hence lead to better actions in future. The agents designed in this work perform learning to achieve one or more of three objectives, which are aimed at (1) deleting rules deemed unnecessary, (2) adding rules expected to be useful in future, and (3) adjusting the membership functions. In order to achieve these objectives, the learning module is made up of three sub-modules, each of which is associated to one of the objectives as detailed in what follows.

Adaptive Rule Weighting (ARW) Adaptive rule weighting involves adjusting a weight w_i. The weight w_i is defined, initialised and attached to each rule R_i during the rule base initialisation step (see Section 6.5). After each learning episode, the weight w_i is adjusted in two ways; First, w_i is decremented by a constant $\varphi_1 = 1$ if rule R_i was fired, and incremented by a constant $\varphi_2 = 0.5$ if the rule was not fired. The reason for adjusting w_i based on whether a rule was fired or not is that if a given rule is consistently not used by an agent, then it is more likely that the rule is unnecessary, and should therefore be dropped from the rule base. In this thesis, rules with lower values of w_i are considered more important than those with higher values; in fact, in our proposal, when the value of w_i is greater or equal to a constant Ψ_1, the agent can consider the rule unnecessary or counterproductive and hence the rule R_i is deleted from the rule base.

In addition, for all fired rules, w_i is changed according to equation (6.8).

$$w_i^{new} = w_i - r(v) \tag{6.8}$$

where $r(v)$ is a dynamic evaluation reward defined in (6.9). The objective of (6.8) is to reduce the weight w_i whenever the rule contributes to a correct action ($r(v)$ is positive), and increase it otherwise.

Agent Cooperation (AC) While adaptive rule weighting helps an agent get rid of unnecessary rules, it does not help the agent acquire more rules that are expected to be important to future actions. Agent cooperation can be used for this purpose. Our proposal in this regard involves a *cooperation* between different agents in two ways. First, the agents coordinate to avoid conflicting actions, and then, at predefined times, agents share information aimed at improving their individual performances. By sharing knowledge, the agents can either add or delete rules from their rule bases. We describe cooperation between agents in Section 6.6.

Evaluative Feedback (EF) Both ARW and AC can only add or delete a rule from the rule base. However, it is also necessary to be able to adjust a given rule, by changing the parameters of the membership functions so as to improve the output of those rules that remain in the rule base. This is achieved by using a reinforcement learning-based evaluative feedback mechanism that uses a reward function to adjust the membership functions. We describe the reward function used in this Chapter in 6.4.1, and then derive the rule updating mechanism used to learn the parameters of rule membership functions in 6.4.2.

6.4 Evaluative Feedback

6.4.1 Reward Function

After each learning episode, the affected substrate and virtual nodes/links are monitored, taking note of average utilisation of substrate resources, the delay on virtual links and packets dropped by virtual nodes due to buffer overflows. These values are fed back to the agent in form of a performance evaluation (PE). The reward resulting from a learning episode of any agent is therefore a vector $\mathbf{R}$ in which each term $r(v)$ corresponds to the reward of an allocation to the virtual resource v. This reward is an *indication* of the deviation of the agent's *actual action* from a *desired action*, and is therefore aimed at minimising this deviation. The objective of the reward function is to encourage high virtual resource utilisation while punishing $n_a \in \mathcal{N}_a$ for dropping packets and $l_a \in \mathcal{L}_a$ for having high delays. The reward function defined in this Chapter is designed so as to carry two pieces of information; a *magnitude* and a *direction*. If the agent's action was desirable, $r(v)$ is positive, otherwise it is negative. The magnitude of $r(v)$ gives the degree of desirability or undesirability of the agent's action, and is dependent on resources allocated to the virtual resources, unutilised resources, link delay in case of $l_a \in \mathcal{L}_a$ and the number of dropped packets in the case of $n_a \in \mathcal{N}_a$. The resulting reward function is presented in (6.9).

$$r(v) = \begin{cases} \kappa\tau & \text{if } \big(\tau < 0.0\big) \quad (6.9a) \\ \big(R_v - D_v\big) & \forall l_a \in \mathcal{L}_a \quad (6.9b) \\ \big(R_v - P_v\big) & \forall n_a \in \mathcal{N}_a \quad (6.9c) \end{cases}$$

where κ is a constant, and τ is an index of the correctness of the action adopted by the agent. Specifically, τ gives an indication of whether the action a taken by the agent should have been *increasing* resource allocation or *reducing* it, and can take on positive or negative values based on the perceived expected direction of action. It is worth noting that (6.9a) takes precedence over (6.9b) and (6.9c), implying that whenever it is satisfied, the reward is calculated from it. Equations (6.9b) and (6.9c) respectively apply for link and node agents. The value of τ is defined in equations (6.10).

$$\tau = \begin{cases} a & \text{if } \Big((R_a^v \leq \xi_1) \wedge (a < 0.0) \Big) \quad (6.10a) \\ a & \text{if } \Big((R_u^v \leq \xi_2) \wedge (a < 0.0) \Big) \quad (6.10b) \\ -a & \text{if } \Big((R_u^v \geq \xi_3) \wedge (a > 0.0) \Big) \quad (6.10c) \\ 0 & otherwise \quad (6.10d) \end{cases}$$

where ξ_1, ξ_2, and ξ_3 are constants, and a is the crisp output value of the agent. The definition of τ is aimed at ensuring that actions that could possibly lead an agent away from its objective receive a negative feedback. In fact, equation (6.10a) is aimed at ensuring that resource allocations below ξ_1 are avoided as this could easily have a negative impact on the QoS requirements of the virtual resource. In particular, (6.10a) states that: if the resource allocated to v is already below a given minimum ξ_1 and the agent's action is to further reduce the allocation ($a < 0.0$), then the reward for this agent should be negative, proportional to the amount by which resource allocation was reduced. In the same way, equation (6.10c) states that if at least ξ_3 of the resources allocated to the virtual resource v are unutilised and the agent decides to increase the resource allocation ($a > 0.0$), then this action takes the agent in the wrong direction, and as a result, τ should be negative. The constant κ in equation (6.9a) can be adjusted to result into higher/lower negative rewards so as to guide the agent away from situations where $\tau < 0.0$. The values $\kappa = 1$, $\xi_1 = \xi_2 = 0.25$ and $\xi_3 = 0.75$ were used in this Chapter.

R_v is the utilisation of resources allocated to v and is derived from R_u^v, while D_v and P_v are measures of the performances of link agents and node agents respectively, and their values are derived from the link delay D_{ij} and number of lost packets P_i.

$$R_v = 1 - R_u^v, \;\; D_v = \frac{D_{ij}}{0.1} \;\; and \;\; P_v = \frac{P_i}{100}$$

The reason for scaling D_v and P_v is to ensure that $0 \leq D_v \leq 1$ and $0 \leq P_v \leq 1$ which is also part of the effort to ensure that $-1 \leq r(v) \leq 1$. The choice of the values 100 and 0.1 is based simulations in which we observed that $0 \leq D_{ij} \leq 0.1$ and $0 \leq P_i \leq 100$. In any case, the values of D_v and P_v are capped at a value 1 and if the scaling results into a value greater than 1, then the maximum value 1 is used.

6.4.2 Membership Function Learning

With the overall reward determined, we now need to determine the contribution of each fired rule $R_i \in R'$ to the reward, and use it in the learning process to update the membership functions. We can derive the contribution of each rule as a gradient descent (6.11) on squared error (6.14), and thereafter adjust a parameter, ε of the membership functions belonging to the rule R_i so as to reduce the general error.

$$\Delta\varepsilon = -\alpha\left(\frac{\partial E}{\partial \varepsilon}\right) \tag{6.11}$$

where $\Delta\varepsilon$ is the amount by which parameter ε should be changed so as to reduce the error E in its action, $0 \leq \alpha \leq 1$ is referred to as *learning rate*, and it determines how fast learning occurs. Therefore, an updated value ε_{new} of the parameter can be determined from its current value ε using equation (6.12).

$$\Delta\varepsilon = \varepsilon_{new} - \varepsilon \tag{6.12}$$

Combining (6.11) and (6.12) results into the general learning rule for ε shown in (6.13)

$$\varepsilon_{new} = \varepsilon - \alpha\left(\frac{\partial E}{\partial \varepsilon}\right) \tag{6.13}$$

The error E is a measure of the difference between the expected optimal action a^* and the actual action a, and is usually given by a square of the difference between a^* and a as shown in (6.14).

$$E = \frac{1}{2}(a^* - a)^2 \tag{6.14}$$

In order to determine a learning procedure over ε, we start by determining the error rate $\partial E/\partial\varepsilon$, which can be derived using the chain rule shown in (6.15).

$$\frac{\partial E}{\partial \varepsilon} = \frac{\partial E}{\partial a} \times \frac{\partial a}{\partial \mu} \times \frac{\partial \mu}{\partial \varepsilon} \tag{6.15}$$

Using (6.14), $\partial E/\partial a = -(a^* - a)$, and as defined in 6.4.1, the difference between expected and actual actions $(a^* - a)$ is given by the reward function $r(v)$. $\partial a/\partial \mu$ is the contribution of membership function μ to the overall action a (and hence its contribution towards the error). For this work, the value $\lambda^{-1}(y_{min})$ as defined in 6.3.2 as the inference result of the antecedent of the rule under consideration is used. Finally, if the parameter ε is along the horizontal axis of the membership function, then $\partial \mu/\partial \varepsilon$ is the gradient of the membership function (i.e. $\partial \mu/\partial \varepsilon = \partial y/\partial x$) and is given by $1/(q - p)$ (using equation (6.6)). Therefore, (6.15) becomes

$$\frac{\partial E}{\partial \varepsilon} = -r(v) \times \lambda^{-1}(y_{min}) \times \frac{1}{(q-p)} \tag{6.16}$$

Substituting (6.16) into (6.13) gives (6.17), which is the membership parameter learning equation used in this thesis.

$$\varepsilon_{new} = \varepsilon + \alpha\left(\frac{r(v)}{(q-p)} \times \lambda^{-1}(y_{min})\right) \tag{6.17}$$

In this thesis, the membership parameter ε to be learnt is chosen as q. This is based on the observation, from equation (6.17), that for negative values of $r(v)$ (i.e. the action taken by the agent was undesirable), the new value ε_{new} increases if $p > q$ and reduces otherwise. Choosing to learn the value of q ensures that whenever an undesirable action is taken, the value $\Delta x = |q - p|$ is reduced (the gradient of the membership function is increased), hence making the rule under consideration less likely to be fired by inputs in the same range. On the other hand, for positive values of $r(v)$ (a good action was taken), we do increase Δx, making the rule more likely to be used by inputs in a close range. To avoid possibilities of division by zero (when $p = q$), we add a small constant δ_0 to the value $(q - p)$ in the denominator of (6.17). Therefore,

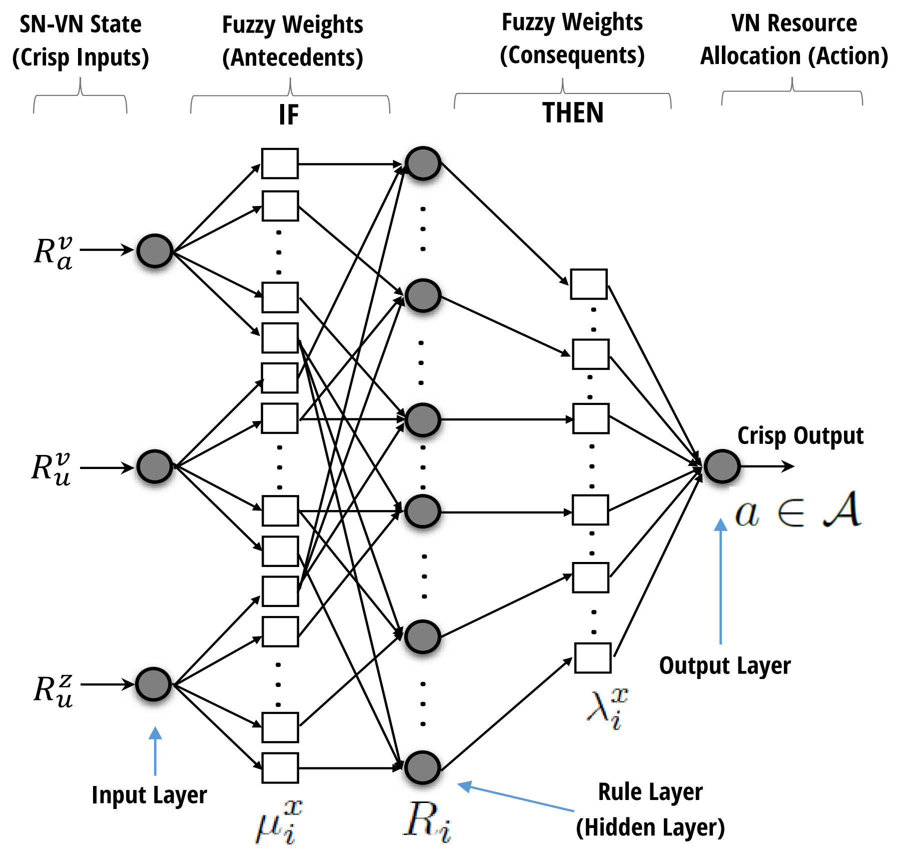


Figure 6-5: Neuro-Fuzzy Network for VN Resource Allocation

the final membership function parameter learning rule is given in (6.18).

$$q_{new} = q + \alpha \left(\frac{r(v)}{(q-p)+\delta_0} \times \lambda^{-1}(y_{min}) \right) \tag{6.18}$$

Since the crisp value of the output is based on a combination of the different input membership functions, adjustments in input MFs directly affect future outputs of the same rule. For this reason, in this proposal, the updating of the membership functions is restricted to antecedents of the rules.

6.4.3 Neuro-Fuzzy System Network Structure

With the overall model defined, the next step is to design the actual neuro-fuzzy network. We propose a 3-layer feedforward network with an input layer, a rule (hidden) layer and an output layer. In Fig. 6-5 we show such type of network, designed for the resource allocation model represented in Fig. 6-2. As can be seen, the input layer contains 3 neurons and has as inputs the three variables R_a^v, R_u^v and R_u^z which we use to define the state of system resources. The output layer contains a single neuron, and its output is an action $a \in \mathcal{A}$ aimed at changing resource allocation. The rule (hidden) layer contains fuzzy if-then rules that are used by the system to make resource allocation decisions. On the left side of the rule layer are fuzzy weights μ_i^x that represent the weight of each connection from an input to a rule node R_i, and using the fuzzy set x. Similarly, the right hand side of the rule layer has another set of fuzzy weights λ_i^x connecting a rule R_i to the output, represented a fuzzy set x. While it is possible for different rule nodes to share some weights on either side, the network designed in this thesis creates a unique fuzzy weight for each rule. This way, the rule addition/deletion described in 6.3.2 involves creating/cutting connections between the appropriate input-rule-output nodes of the network. In the same way, membership function learning involves adjusting these weights for the appropriate connections/rules.

6.5 Rulebase Initialisation

At the beginning of the learning process, the system has no rules. Therefore, we need to define a way of establishing an initial rule base. One rule initialisation possibility is a decremental rule learning proposed in [17] in which all the possible rules are initialised into the system and then subsequently reduced as the agent learns. As already mentioned our system can work with up to 1728 i.e. $(6 \times 6 \times 6 \times 8)$ rules, and starting the learning process with this high number of rules would slow down the learning process. Our proposal starts by creating the maximum possible rule base with 1728 rules. A weight $w_i = 0$ is then attached to each rule R_i, and is used to perform a weighting and pruning process that is based on expert knowledge. The initialisation proposed in this thesis includes three sequential steps as described below.

1. The first step takes into account the likelihood that a rule may not be required when in optimal operation. This is based on the design objectives of the system.

Table 6.2: Action and State to Membership Function Mapping

(a)

Previous Action	**MF**
Maintain R_a^v unchanged	PZ
Decrease R_a^v by 50.0%	NL
Decrease R_a^v by 37.5%	NM
Decrease R_a^v by 25.0%	NS
Decrease R_a^v by 17.5%	NZ
Increase R_a^v by 17.5%	PZ
Increase R_a^v by 25.0%	PS
Increase R_a^v by 37.5%	PM
Increase R_a^v by 50.0%	PL

(b)

Previous State (% Value)	**MF**
$0 <$ Variable ≤ 12.5	VL
$12.5 <$ Variable ≤ 25	L
$25 <$ Variable ≤ 37.5	LM
$37.5 <$ Variable ≤ 50	LM
$50 <$ Variable ≤ 67.5	HM
$67.5 <$ Variable ≤ 75	HM
$75 <$ Variable ≤ 87.5	H
$87.5 <$ Variable ≤ 100	VH

Table 6.3: Output Membership function to Integer Mapping

NL	NM	NS	NZ	PZ	PS	PM	PL
-4	-3	-2	-1	1	2	3	4

Specifically, since we would like the system to be mindful of the QoS requirements of VNs, the system is unlikely to be in states where $R_a^v =$ VL. Similarly, efficient substrate resource utilisation would ensure that states with $R_u^v =$ VH and/or $R_u^z =$ VH are less likely to occur. For each of such rules, the weight w_i is incremented by δ_1.

2. The second takes into account the likelihood that a given rule could cause the agent to take a wrong action. Examples of such rules could be in situations where a given resource allocation is very high, but the selected action is to increase the allocation even more by a *very high* percentage. In particular, rules in which $R_a^v =$ VH with actions PM and PL, $R_a^v =$ VL with actions NM and

NL and those R_u^v = VL with actions NM and NL are likely to lead to wrong actions. For each of such rules, the weight w_i is incremented by δ_2.

3. In the final step, each of the rules is evaluated based on an input-output dataset, and the weight w_i adjusted again. The dataset used for this purpose was saved from the q-table of a reinforcement learning approach proposed in [146]. This q-table was a result of a resource allocation learning system for a similar resource allocation task and it gives the state-action-values for the learning task. The table is made up of 3 columns, one for the *state* (which is also defined by three variables R_a^v, R_u^v and R_u^z), another for a possible *action*, and the other for a *value* that shows the desirability of taking the action while in the given state. However, before the dataset can be used for the pruning proposed in this thesis, we need to process it so as to put in a form similar to fuzzy rules. In 6.5.1, we describe the preprocessing steps taken. After this process, all rules for which the weight w_i is greater than a pre-established constant Ψ_2 are pruned from the rule base.

6.5.1 Dataset Preprocessing

The training dataset is made up of 4608 entries (resulting from 512 possible states and 9 possible actions), each showing the value of every possible action while in each state. The first step is to choose only those entries corresponding to the best possible action (actions with the best values) for each state. This leaves us with 512 entries. We then convert these state-action-values into fuzzy rules. However, this requires a mapping from the state and action codes used in [146] to the fuzzy sets used in this thesis. In Tables 6.2(a) and 6.2(b) we show the mapping that has been performed on the states and actions. However, since each of the state MFs can only take on 6 values for the 3 input variables, the maximum number of possible *unique* antecedent combinations is $6 \times 6 \times 6 = 216$. Therefore, we again prune the training dataset eliminating entries with the same antecedent (remaining with one rule with the highest original state-action-value for all duplicate rules). After this final step, we have a training rule set R_{RL} with 216 rules that we can use as a training set for an initial pruning of the rule base.

6.5.2 Initial Rule Base Pruning

For each rule $r_j \in R_{RL}$, each rule $R_i \in R$ in the rule base is evaluated so as to determine the correctness of its consequent. To this end, a *rule matching* procedure is performed. This determines whether the antecedent of r_j is the same as that of R_i. The rule matching step involves comparing the fuzzy sets representing the input and/or output variables. For this initialisation step, two rules match if all the three corresponding antecedent fuzzy sets are the same. The result of a rule matching process is either a success if the rules match, or a failure otherwise. If R_i matches r_j then an *error*, which is a measure of how the consequent of r_j differs from that of R_i, is evaluated. To achieve this, we model each of the 8 possible consequents with an integer value. These values are shown in Table 6.3. The *absolute* value, of the difference between the value a_j for the consequent of rule r_j and a_i for rule R_i is then used to increment the weight w_i for rule R_i.

We show the pseudocode for the rule initialisation in Algorithm 6. In the algorithm, μ_i^1, μ_i^2, μ_i^3 and λ_i^1 are the *initial* fuzzy sets for the variables R_a^v, R_a^v, R_a^v and O respectively. The fuzzy sets shown Figs. 6-3 and 6-4 are used for the initialisation stage, but as learning progresses, these sets change for the respective rules. After the weighting and pruning stage, the proposed initialisation algorithm reduces the initial 1728 rules to 1215 rules[4].

[4]The initial rule base, training dataset and final initialised rule base can be downloaded from: `http://www.maps.upc.edu/rashid/files/nfsrules.rar`.

Algorithm 6 *Rule Base Initialisation*

```
1: **Initial Rule Base $R$ Creation**
2: Initialise rule weight: $i = 1$
3: **for** $\mu_i^1 \in \mu$ **do**
4:   **for** $\mu_i^2 \in \mu$ **do**
5:     **for** $\mu_i^3 \in \mu$ **do**
6:       **for** $\lambda_i^1 \in \lambda$ **do**
7:         Create Rule: $R_i$ **if** $\Big(R_a^v$ is $\mu_i^1$ **and** $R_u^v$ is $\mu_i^2$
8:                     **and** $R_u^z$ is $\mu_i^3\Big)$ **then** $O$ is $\lambda_i^1$
9:         Initialise weight: $w_i = 0$
10:        Increment rule weight: $i++$
11:      **end for**
12:    **end for**
13:  **end for**
14: **end for**
15: **Rule Weighting and Pruning**
16: **for** $R_i \in R$ **do**
17:     *Step 1: Efficiency and QoS Awareness*
18:   **if** $(\mu_i^1 = VL)$ **or** $(\mu_i^2 = VH)$ **or** $(\mu_i^3 = VH)$ **then**
19:     $w_i = w_i + \delta_1$
20:   **end if**
21:     *Step 2: Protection against wrong actions*
22:   **if**
      ($\mu_i^1$ = VH **and** $\lambda_i^1$ = PM) **or**...**or** ($\mu_i^2$ = VL **and** $\lambda_i^1$ = NL)
        **then**
23:     $w_i = w_i + \delta_2$
24:   **end if**
25:     *Step 3: Learning from Dataset*
26:   **for** $r_j \in R_{RL}$ **do**
27:     Perform **Rule Matching** $(r_j, R_i)$
28:     **if** $(Matching = Success)$ **then**
29:       $w_i = w_i + (|o_j - o_i|)$
30:       **if** $w_i >= \Psi_2$ **then**
31:         DELETE $R_i$
32:       **end if**
33:     **end if**
34:   **end for**
35: **end for**
```

6.5.3 Time Complexity of Rule Base Initialisation

The initial rule base creation stage in Lines $2-13$ involves three basic operations (Lines 6, 8 and 9) for each of the possible rules R. Therefore, this step can be performed in time $O(|3R|)$. The rule matching operation involves at most four operations (3 for the antecedent and 1 for the consequent). Therefore, Lines $24-30$ includes at most six operations, implying that the for loop from Lines $23-31$ runs in time $O(|6R_{RL}|)$. Including the two operations on Lines 16 and 21 leads to the time $O\Big(|(6R_{RL}+2)R|\Big)$ for the Lines $14-32$. Therefore, the dominating factor in Algorithm 6 is $O(|R_{RL} \times R|)$. It is worth noting that both $|R|$ and $|R_{RL}|$ are predefined as 1728 and 216 respectively.

6.6 Agent Cooperation

In this approach, the substrate node or link agents can cooperate on two fronts. The first is an action coordination aimed at conflict prevention, while the other is a knowledge sharing aimed at learning enhancement. We briefly describe both of them below.

6.6.1 Coordination amoung Agents

From the VNE problem formulation, a given virtual link l_{ij} may be mapped onto more than one substrate link. This creates a possibility of more than one substrate link agent dynamically managing the resources allocated to such a virtual link. In this case, the set of agents $L_a^{l_{ij}} \subset \mathcal{L}_a$ that are able to change the resource allocation to l_{ij} must coordinate their actions to avoid conflicting resource allocations. The first step in the conflict prevention proposed in this Chapter is the creation of the agent set $L_a^{l_{ij}}$. After every VNE step, each substrate link agent that participated in the embedding determines - for each new embedded virtual link - the set of other substrate link agents that manage the virtual link resources. Since we consider that all the agents in our model belong to the same organisation (the infrastructure provider), we consider that this kind of information is readily available to all agents. The next step is to allow each agent $l_a \in L_a^{l_{ij}}$ to communicate with every other agent in the same set, sharing resource allocation information every time an allocation is performed. Therefore, after each learning episode, if an agent $l_a \in L_a^{l_{ij}}$ decides to change the amount of resources allocated to l_{ij}, it sends an update to other agents in $L_a^{l_{ij}}$ providing information about the action a to be taken as well as the final percentage resource allocation $R_a^{l_{ij}}$ resulting from the action. All the agents in $L_a^{l_{ij}}$ therefore perform the resource change at the same time, ensuring that the final percentage resource allocation to the virtual link is $R_a^{l_{ij}}$. Once again, the fact that a given agent is able to trust and take actions based on decisions of another agent is reasonable since all these agents belong to the same organisation and as such, they cannot have conflicting objectives. Finally, in order to ensure that the actions and information sharing of the agents $L_a^{l_{ij}}$ is synchronised, only one of them learns at any given time. This is achieved by starting the learning processes of each agent at different times on their creation and thereafter performing learning at regular intervals.

Scalability consideration of the Agent Cooperation mechanism

It is worth noting that in general, if the link mapping algorithm is efficient, the agent set $L_a^{l_{ij}}$ will contain an average of $2-3$ agents[5]. This means that at any point, a given agent $l_a \in L_a^{l_{ij}}$ only needs to send update messages to about $1-2$ other agents. We consider that this number of update messages is manageable, and would not congest the network. In addition, we specifically avoid the exchange of "acknowledge" messages to diminish as much as possible the traffic among agents, instead preferring to use an update message that also includes the final resource allocation to l_{ij}. This ensures that if for any reason a given agent does not get an update message, this can be corrected at the next learning episode.

[5]Based on simulations carried out using the S-OS algorithm (see Table 6.4) for average link bandwidth utilisation levels between 50% and 70%.

Algorithm 7 *Neuro − Fuzzy Learning Algorithm*

```
1: Initialisationnfs
2: Initialise Rule Base, $R$
3: Determine current state, $s_c$
4: Define: previous state, $s_p = s_c$, previous action, $a_p = 0.0$, next state, $s_n = s_c$, set
   of fired rules $F = \emptyset$.
5: Thread1: Learn from others (Knowledge Sharing)
6: repeat
7:    Wait(Cooperation Interval)
8:    Receive rule base $R_x$ from other agents
9:    for $R_j \in R_x$ do
10:       if $R_j \in R$ then
11:          for $R_i \in R$ do
12:             Update $p_i$ as described in 6.6.2
13:          end for
14:       else
15:          Add $R_j$ to $R$
16:       end if
17:    end for
18: until Learning is stopped
19: Thread2: Learn from actions (Evaluative Feedback)
20: repeat
21:    Wait(Learning Interval)
22:    Read $s_p$, $a_p$, $s_n$, $F$
23:    Determine $r(v)$ using equation (6.9)
24:    for $r_i \in F$ do
25:       for $\lambda_i^x \in r_i$ do
26:          Determine $q_{new}^{\lambda_i^x}$ using equation (6.18)
27:       end for
28:       Update weight $w_i$ as using ARW in 6.3.2
29:    end for
30:    Set $F = \emptyset$
31:    for $R_i \in R$ do
32:       if $w_i \geq \Psi_1$ then
33:          Delete $R_i$
34:       end if
35:    end for
36:    Determine current state $s_c$, add all fired rules to $F$
37:    Determine action, $a \in A$ as explained in 6.3.2 - 6.3.2
38:    Take action $a$, and determine next state, $s'_n$
39:    Set $s_p = s_c$, $a_p = a$, $s_n = s'_n$
40: until Learning is stopped
```

6.6.2 Knowledge Sharing amoung Agents

The second form of cooperation between agents involves sharing of their knowledge bases. In this proposal, each learning agent $a_s \in (\mathcal{L}_a \cup \mathcal{N}_a)$ periodically shares its rule base R_{a_s} as well as database of membership functions f_{a_s} with other agents $a_t \in A_t$, where $A_t \subset (\mathcal{L}_a \cup \mathcal{N}_a)$. For a given link agent, A_t consists of the node node agents at its ends, while for a node agent, A_t includes all the link agents for the substrate links connected to the node[6].

There are two advantages that are derived from this cooperation. First, it leads to performance enhancement if it leads to addition of new knowledge to the base or to improvement of the membership functions of existing rules and then it allows a faster convergence to optimal network structure in case it leads to deletion of some rules. For each newly received rule $R^i_{a_s} \in R_{a_s}$, the integration into the rule base is a four-step matching and elimination process. The receiving agent a_t compares $R^i_{a_s}$ to each of the rules in its rule base, performing a rule matching (described in 6.5.2) over both the input and output variable fuzzy sets. If the result of matching for the whole rule base is a failure (the rule $R^i_{a_s}$ does not match any of the rules in the rule base), then the agent adds the rule $R^i_{a_s}$ together with its membership functions to its knowledge base. On the other hand, if the result of matching is a success (there is a rule that is similar to $R^i_{a_s}$), then the agent learns from the membership functions of $R^i_{a_s}$.

This is achieved by replacing the $p - values$ of the membership functions of the matching rule R_i with a weighted sum defined in (6.19).

$$p_{new} = (\gamma \times p_1) + (\beta \times p_2) \tag{6.19}$$

where p_1 is the old *p-value* and p_2 is the *p-value* of the received rule. γ and β are constants intended to bias the sensitivity of the agent to knew information. $\gamma + \beta = 1$. In this Chapter, the values $\gamma = 0.7$ and $\beta = 0.3$ are used. Needless to mention, the

[6]The restriction of the respective sets of agents is for scalability reasons as explained later in this subsection

overall learning scheme proposed in this thesis learns both the p and q parameters of the membership functions. The q parameter is learnt through the evaluative feedback described in 6.4.2 while this subsection has defined a learning procedure for the p values.

Scalability of the Knowledge Sharing Mechanism

Allowing every agent $a_s \in (\mathcal{L}_a \cup \mathcal{N}_a)$ to communicate and share knowledge with every other agent in the system would be non-scalable. In this thesis, we restrict each agent to only share knowledge with its direct neighbours. This means that any link agent $l_a \in \mathcal{L}_a$ can only share experience with at most two node agents at either end of the link. In the same way, any given node agent $n_a \in \mathcal{N}_a$ can only share knowledge with only those link agents that directly connect it to its adjacent nodes.

The complete learning algorithm proposed in this Chapter is shown in Algorithm 7. As can be seen, the learning process is made up of different steps as already described. However, we note that some of the processes take place in parallel, for example, the agents continuously learn from each other (knowledge sharing) independently of the learning achieved from the evaluative feedback.

6.6.3 Time Complexity of Neuro-Fuzzy Learning Algorithm

The initialisation in Line 1 can be performed in time $O(|R_{RL} \times R|)$ as established in 6.5.3. Lines $7-15$ require at most $O(|R|^2)$ time, while the for loop in lines $21-26$ can run in time $O(|5R|)$. Finally, Lines $28-32$ can be performed in time $O(|R|)$. Since $R > R_{RL}$, the overall time complexity of algorithm 7 is $O(|R|^2)$. Therefore, both algorithms proposed in this Chapter run in polynomial time.

Table 6.4: Compared Algorithms

Code	**Resource Allocation Approach**
D-NFS	Dynamic, based on Neuro-Fuzzy System [Our Contribution]
D-RL	Dynamic, based on Reinforcement Learning[146]
D-ANN	Dynamic, based on Artificial Neural Networks [158]
S-CNMMCF	Static, Coordinated Node Mapping and MCF for link mapping[8]
S-OS	Static, link based optimal one shot Virtual Network Embedding[146]

Table 6.5: Compared Approaches - Initilisation and Agent Cooperation

Code	**Resource Allocation Approach**
D-RL	Reinforcement Learning[146]
D-ANN	Artificial Neural Networks[159]
I-C	Initialised Rule Base, Cooperating Agents
I-NC	Initialised Rule Base, Non Cooperating Agents
NI-C	Non Initialised Rule Base, Cooperating Agents
NI-NC	Non Initialised Rule Base, Non Cooperating Agents

6.7 Performance Evaluation

6.7.1 Simulation Model

To evaluate our proposal, a simulation scenario similar to Fig. 4-9 was setup. SN and VN topologies were generated using Brite [110] with settings shown in Table 4.4. Thereafter, VN requests arrive, one at a time to the SN. Whenever a VN request is accepted by the SN, the VN topology is created in NS3 [133] using the network virtualisation module and real traffic traces explained in Section 4.5. Simulations were run on an Ubuntu 12.04 LTS Virtual Machine with 4.00GB RAM and 3.00GHz CPU specifications. The substrate and virtual network topologies are created according to parameters described in Section 5.4.2.

6.7.2 Comparison against Alternatives

We compare the performance of our proposed solution with closely related solutions. In particular, four representative solutions from the literature are chosen. The first two perform dynamic resource allocation using one shot VNE for the first step and reinforcement learning [146] (our approach in Chapter 4) and artificial neural networks [159] (our approach in Chapter 5) respectively for resource management, the third is a static allocation approach that performs a coordinated node and link mapping [8]; and the last one is also a static baseline formulation that performs a one shot mapping, and also used in performance evaluations in [146]. The solution in [8] was adapted to fit into our formulation of the problem. In particular, for [8] the link delay requirements were neglected at the embedding stage, and for this reason, it is not used in QoS evaluations. In addition, our consideration in this Chapter is for unsplittable flows. We identify and name the compared solutions in table 6.4. We also compare different variations of our proposal to determine the effect of initialising rule bases as well as sharing knowledge between the agents. Details of these variations are shown in Table 6.5.

6.7.3 Performance Metrics

We evaluate the performance of our proposal on two fronts; the embedding quality, as well as the quality of service of the virtual networks. Our goal is that the opportunistic use of virtual network resources should not be at the expense of the service quality expectations of the network users.

Embedding Quality

We define embedding quality as a measure of how efficiently the algorithm uses the substrate network resources for accepting virtual network requests. This is evaluated using the acceptance ratio and the total instantaneous accepted virtual networks. The acceptance ratio is a measure of the long term number of virtual network requests that are accepted by the substrate network. The total instantaneous accepted virtual networks is a measure of the embedding cost incurred by a given substrate network, as a substrate network that incurs a lower embedding cost normally has more extra resources at any point and hence is able to have many embedded virtual networks at any point.

Quality of Service

We use the packet delay and drop ratio as indications of the quality of service. We define the packet delay as the total time a packet takes to travel from its source to its final destination. The drop ratio is defined as the ratio of the number of packets dropped by the network to the total number of packets sent. As shown in Table 4.3, we model the networks to drop packets due to both node buffer overflow as well as packet errors. In addition, as it is more important in some applications, we define the variations of these two parameters. The jitter (delay variation) is defined as the difference between delays during different time periods, while the drop ratio variation is defined as the variation between packet drops in different time periods. The time interval to update the measurements corresponds to the transmission of 500 packets.

6.7.4 Discussion of Results

The simulation results are shown in Fig. 6-6 – 6-13. As can be seen from Fig. 6-6, while all three dynamic approaches perform better than the static ones in terms of virtual network acceptance ratio, the neuro-fuzzy approach outperforms all four. The reason for the dynamic approaches performing better than the static ones is that in former cases, the substrate network always has more available resources than in the later case, which is a direct result of allocating and reserving only the required resources for the virtual networks. The fact that NFS outperforms the RL approach can be attributed to two factors: (1) the NFS system models the states and actions with better granularity i.e. without restricting the states and actions to few discrete levels, and (2) the NFS system is more dynamic in the sense that it continuously changes its knowledge base by adding rules, modifying them, and deleting others.

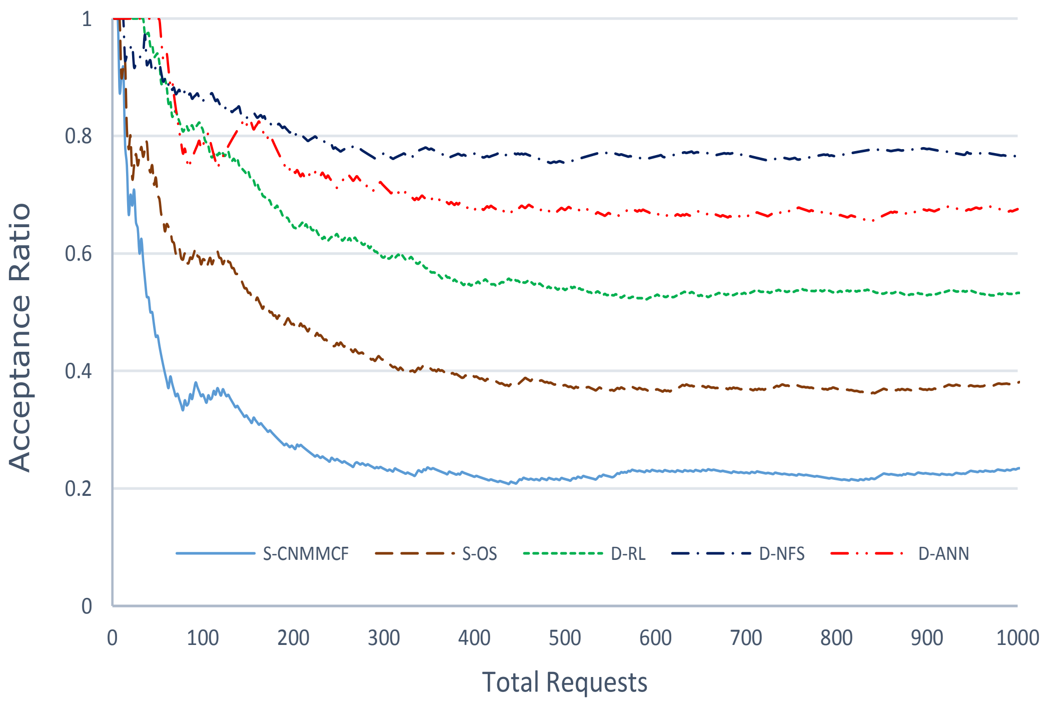


Figure 6-6: VN Acceptance Ratio

On the other hand, the superiority in performance of D-NFS over D-ANN can be attributed to the fact that the network structure in the D-NFS approach is more dynamic (the addition/removal of rules implies addition/removal of network links) compared to a static network structure in D-ANN.

We also note that S-OS has a better acceptance ratio than S-CNMMCF. This is due to the fact that since S-CNMMCF performs node and link mapping in two separate steps, link mappings could fail due to locations of already mapped nodes. In addition, the link mapping phases could potentially use more resources than if both steps are performed once. A similar performance pattern can be noted from Fig. 6-7 which further confirms that at any given point, the substrate networks that dynamically manage resources are able to embed more VNs than the static ones, and that D-NFS performs better than both D-RL and D-ANN, and S-OS better than S-CNMMCF.

Fig. 6-8 and Fig. 6-9 show the average utilisation of substrate node queue size and link bandwidth respectively. It can be observed that except for S-CNMMCF, the other

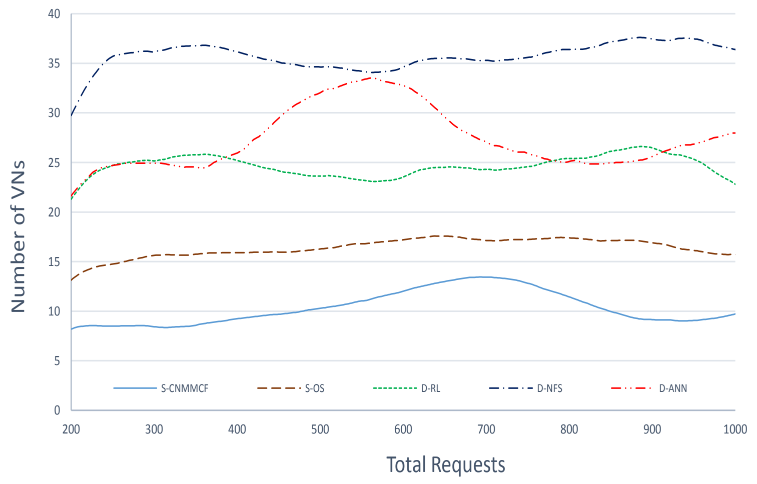


Figure 6-7: Number of Accepted Virtual Networks

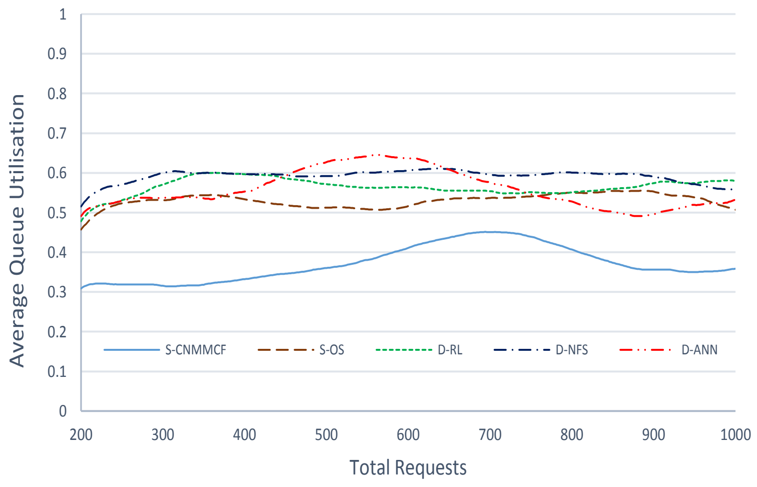


Figure 6-8: Average SN Queue Size Utilisation

four approaches on average use the same amount of substrate network resources. The fact the S-CNMMCF has a lower resource utilisation is expected as a result of having slightly more resource requests rejected either due to a node mapping that makes link mapping impossible, or for previous link mappings using more resources. The fact that S-OS, D-RL, D-ANN and D-NFS all have on average the same utilisation is mainly due to all of them having the same initial mapping algorithm (which is S-OS). However the interesting point from looking at Figures 6-6 – 6-7 is that while S-OS, D-RL, D-ANN and D-NFS all have a similar resource utilisation levels, D-NFS uses these resources to serve a higher number of VNs at any given time, which confirms

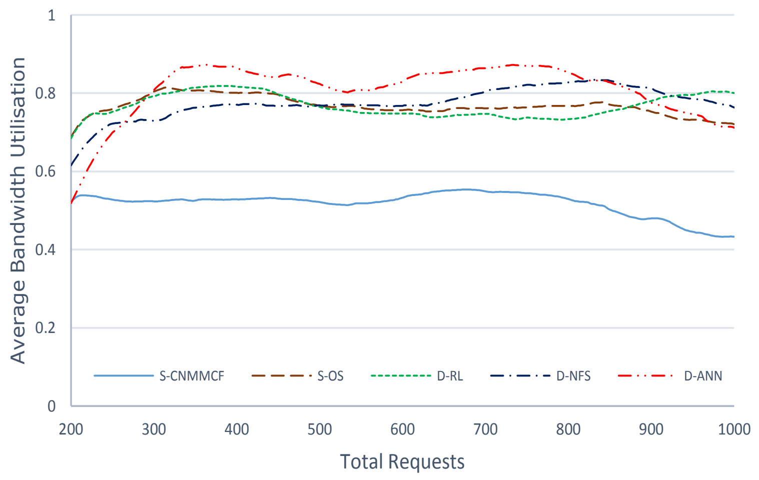


Figure 6-9: Average SN Bandwidth Utilisation

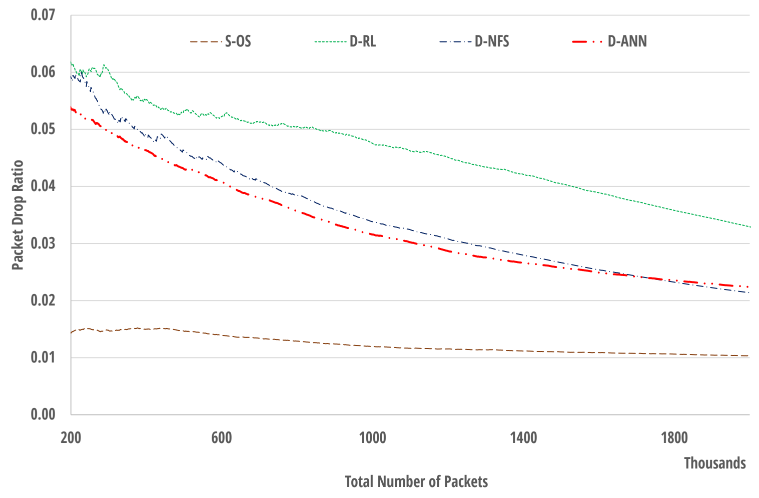


Figure 6-10: Node Packet Drop Ratio

the extra efficiency introduced by the proposed approach.

Fig. 6-10 shows that S-OS has an almost constant packet drop ratio while that for D-RL, D-ANN and D-NFS is initially high, but gradually reduces. The fact that the dynamic approaches initially perform badly is expected, since, at the beginning of the learning processes the agents may vary the queue sizes quite considerably leading to more packet drops. This high initial packet drop also affects the overall speed at which the drop ratio converges to the one in the static approach. This can be confirmed by noting from Fig. 6-11, that in fact, the periodic drops in packets by all approaches finally converge. Once again, the fact that D-NFS has a lower packet drop

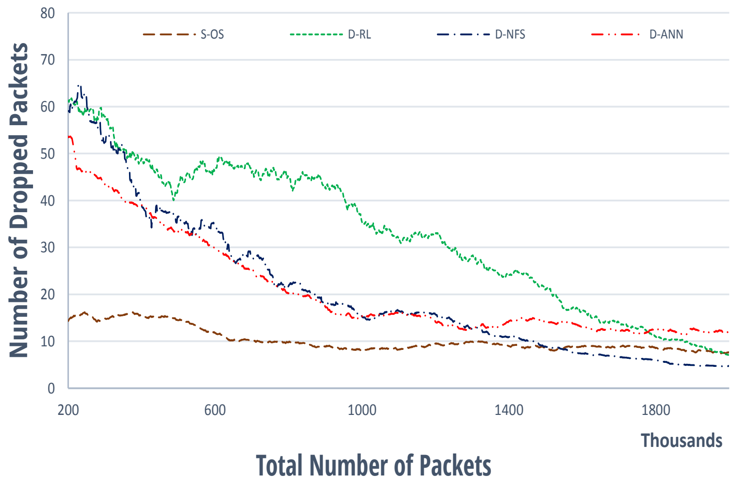


Figure 6-11: Node Packet Drop Ratio Variation

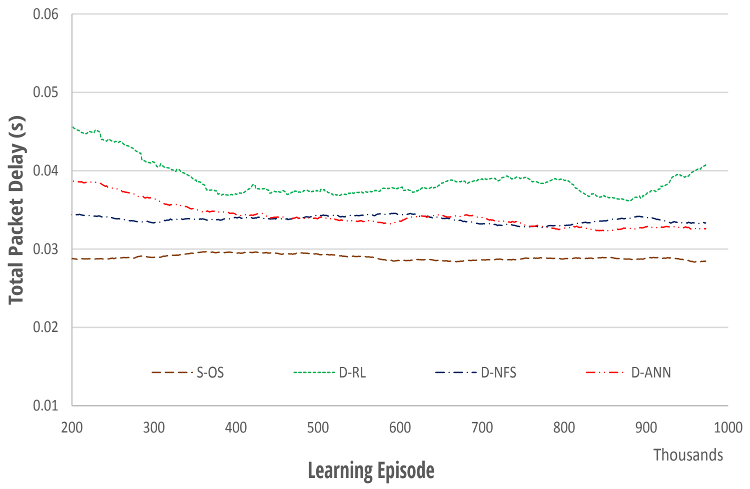


Figure 6-12: Link Packet Delay

ratio than D-RL over the learning period can be explained since D-NFS has better granularity in perceiving the state of resources and allocation. It can however be noted that both D-ANN and D-NFS have a roughly similar packet drop profile. This can be explained from the fact that both of them are initially trained (initialised) using a similar data set (See [158], [158]).

In a similar way, Fig. 6-12 shows that the packet delays for the two dynamic approaches is initially higher but reduces over the learning period, while 6-13 shows that in fact the variations in this delay converge to the static approach. Again, these differences are attributed to the initial learning phase, and the difference in D-RL and

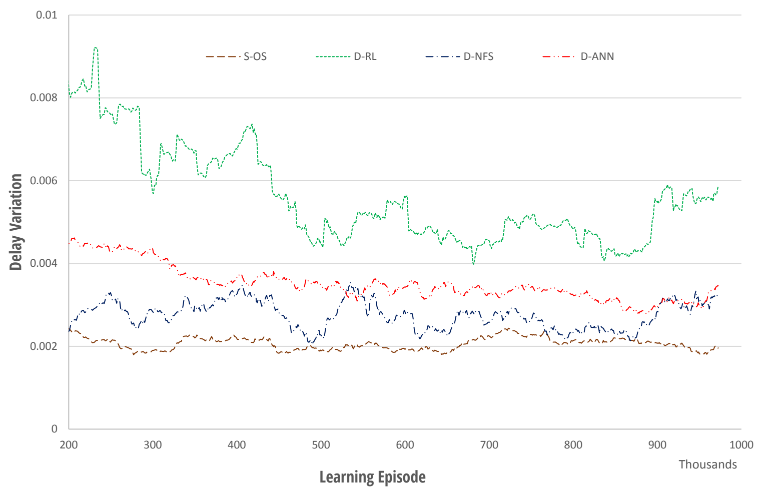


Figure 6-13: Link Packet Delay Variation

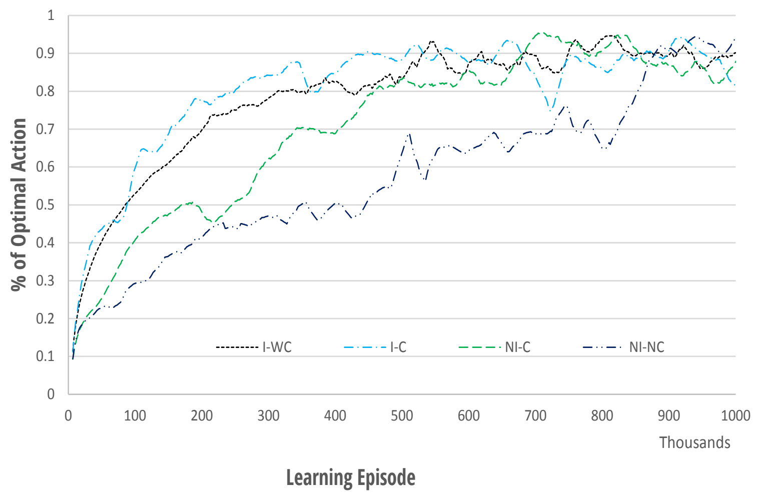


Figure 6-14: Initialisation and Agent Cooperation

D-NFS is due to better options in perception and action for D-NFS. We also note, once more, that due to being initially trained from the same data set, both D-ANN and D-NFS have closely matching packet delay profiles.

Finally, Figs 6-14 and 6-15 show how fast the agents learn optimal rules (for D-NFS), optimal weights (for D-ANN), and an optimal policy (for D-RL). The actions of the agents are compared with *optimal actions*. An optimal action for an agent is that action that would lead to a resource allocation equal to what the network is actually using [146]. The deviations in these evaluations are therefore with reference to actual resource usage in a similar network that is not performing dynamic allocations. The

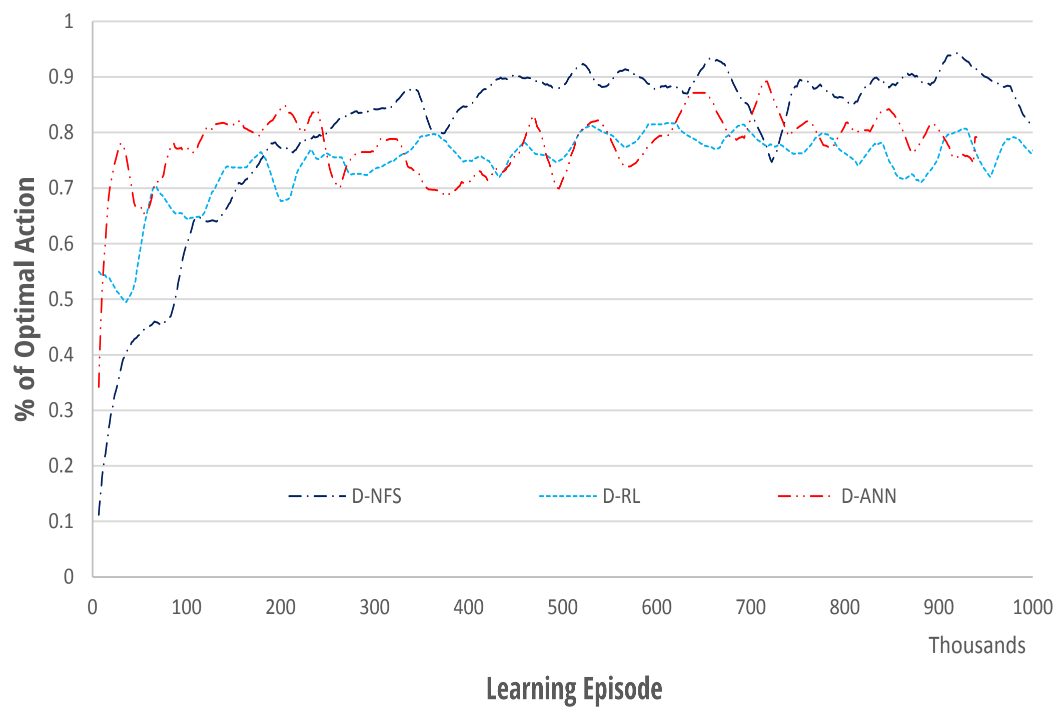


Figure 6-15: Convergence Rate of Machine Learning Approaches

approaches compared in this regard are shown in Table 6.5. Fig. 6-14 shows different variants of D-NFS, while 6-15 compares the best convergence of D-NFS with that of D-ANN and D-RL.

It can be observed from Fig. 6-15, that in general, after convergence, the actions from D-NFS have a slightly higher optimality than those from D-RL and D-ANN. As earlier explained, this is expected due to the granularity of actions taken by D-NFS (with respect to D-RL), and due to a more dynamic network structure (with respect to D-ANN). However, we also note that D-RL and D-ANN both converge to optimal actions slightly faster than D-NFS. This can be explained by the fact that D-NFS does not only need to learn its fuzzy weights, but also its structure, which understandably requires a slightly higher time.

Finally, we also note from the graphs in Fig. 6-14, that the approaches proposed for rule base initialisation and agent cooperation do enhance the speed at which the agents' actions converge optimal ones. However, as can be seen from the graphs, even without rule base initialisation and no cooperation between the agents (NI-NC), the actions would finally converge to the optimal ones, albeit at a much later stage than others.

6.8 Conclusion

This Chapter has proposed an autonomous system that uses an adaptive and hybrid rule-based system which is a combination of neural networks, fuzzy systems and reinforcement learning to achieve dynamic self-management of resources in network virtualisation.

We modelled the substrate network as a distributed system of autonomous, adaptive and cooperative intelligent agents. We started by defining an initial set of fuzzy rules - the knowledge base - for each agent, which were then pruned by use of a hybrid initialisation algorithm which uses supervised learning to initialise the rule base and then uses unsupervised learning to adapt the rule base and fuzzy sets of each rule. A procedure for the agents to cooperate and hence coordinate their resource allocation actions to avoid conflicts and to share their knowledge so as to enhance their learning speed and improve action selection efficiency was then proposed, and finally, a reinforcement learning based algorithm for continuously changing the structure and weights of the designed neuro-fuzzy network is proposed.

We have extensively evaluated the proposed algorithms through comparisons with state-of-the-art approaches and been able to show that our proposal lead to better utilisation of substrate network resources by accepting more virtual network requests, compared to both a tradition reinforcement learning and artificial neural network based solution. Once again, we have also shown that when the agents have learnt a network structure and fuzzy weights, the agents are able to ensure that the QoS requirements of the virtual networks are not negatively impacted.

It is however worth noting that in addition to the improvements in the performance that are obtained by combining neural networks, reinforcement learning, and fuzzy systems, the most important improvement achieved by our approach is in fact un quantifiable. It is in the fact that when the network modelling is represented by physical rules, it is easy to duplicate the designed system to multiple applications, by only adjusting the rules to suit related system tasks. It also avoids the difficulties encountered in designing neural networks, such as the determination of the number of neurons in the hidden layer.

Chapter 7

Virtual Network Survivability

7.1 Introduction

The resource management proposals in Chapters 3 - 6 perform substrate to virtual network resource mappings and allocations in single infrastructure provider environments, and they assume that the substrate network remains fully functional at all times. However, in practice, physical networks do not remain operational at all times [160], hence making the provisioning of resources for service restoration an inevitable part of any survivable network resource management approach. In addition, while multi-provider environments are common in practice, current approaches to survivable VNE have focused on the single provider environment [61], yet an extension of survivability from the single to multi domain environments is not trivial since it involves both intra and inter domain link failures [26]. In a multi-domain setting, provisioning a back-up for protection and/or restoration of a failed substrate resource could involve using resources of more than one InP, which are normally not only competitive, but also secretive. This calls for mechanisms that allow InPs to dynamically and autonomously negotiate these aspects of resource allocation and to be able to make coalitions so as to achieve not only their independent and self-interested goals, but also those of the VNPs whose assembled VNs are distributed across multiple InPs[1].

[1]The roles of InPs and VNPs will be clarified in a subsequent section.

In this Chapter, we propose a distributed, multi-attribute negotiation approach based on a multi-entity negotiating system to support survivability in a multi-domain virtual network environment. The objective is to make each of the VNPs adaptive and dynamic by giving them the capacity to perform price-based back-ups[2] and restorations in case of physical link failures.

To this end, we model each of the InPs and VNPs as an intelligent autonomic negotiating entity [120]. The objectives of the VNP and InP negotiating entities are conflicting; with the VNP aiming at achieving survivable embeddings (by ensurig that the VN it assembles has minimal QoS violations resulting from physical resource failures) at lower costs, while the InPs try to maximise their profits, keeping their resource deployment and pricing strategies private. Our proposed distributed system of negotiating entities is clearly a competitive one, not only between VNPs and InPs, but also amongst InPs as each of them tries to maximise their profits. Therefore, we propose an auction-based multi-attribute negotiation approach that allows InPs to form resource coalitions iteratively, autonomously and selfishly amongst each other, and then pass on final resource proposals to the VNP with which final negotiations occur. We refer to the negotiation as multi-attribute, as it is based not only on the quoted price of the resource, but also on QoS aspects such as expected link delay and data rate.

Since network link failures occur about 10 times more than node failures [61], and given that about 70% of unplanned link failures are single link failures [161], this thesis focusses on protecting and restoring single substrate link failures. We however note that any node failure can be considered as a failure of links adjacent to the node [26], and as such, our proposal can be extended to cover multiple link failures, and hence node failures.

The major contributions of this Chapter are as follows: a negotiation protocol that ensures virtual network survivability with minimum communication message

[2]The back-ups are referred to as "price-based" because the InPs decide whether (or not) to reserve resources for the back-up of a given virtual link depending on the prices that have been quoted (by InPs) for the back-up resources, for example, if the cost of the back-up resources are higher than the "reserve price" of the VNP for that virtual link, then the back-up will not be done.

overhead; VNP and InP negotiation strategies that minimise QoS violation penalties, hence ensuring both VNP and InP profitability; and a dynamic substrate resource pricing model that ensures efficient utilisation of resources. To the best of our knowledge, this is the first endeavor to propose survivability in independent multi-domain virtual network environments.

The rest of this Chapter is organised as follows: We formulates the virtual network survivability problem in Section 7.2, and propose negotiation algorithms to achieve survivable virtual networks in Section 7.3. Our proposals are evaluated and discussed in Section 7.4, and we conclude the Chapter in Section 7.5.

7.2 Problem Formulation

7.2.1 Business Model

The network virtualisation model considered in this Chapter is shown in Fig. 7-1, where a virtual network operator (VNO) assembles and owns virtual networks, using physical resources from one or more infrastructure providers (InPs). In order for the VNO to be able to deal with multiple InPs, the model includes a virtual network provider (VNP) which plays a VNO/InP mediation role (i.e. acts as a broker) to locate and aggregate virtual resources that compose a VN[3].

In Fig. 7-1, the interfaces between the three players (VNO, VNP, and InP) are represented by numbers 1 – 3, and they are defined in Table 7.1. These interfaces represent two types of relationships; vertical (interface 1) and horizontal (interface 2). The vertical interface represents a relationship between a VNP and InPs and mainly represents initial service requests from the VNP to InPs, and the subsequent negotiations resulting from it, while the horizontal relationship is amoung InPs, and involves either forwarding of a service request that could not be completed (from one InP to another) or requests to setup inter-domain substrate paths.

However, while this thesis considers that all the InPs have business relationships

[3]See Section 2.2 for a description of Business models and roles in NVEs

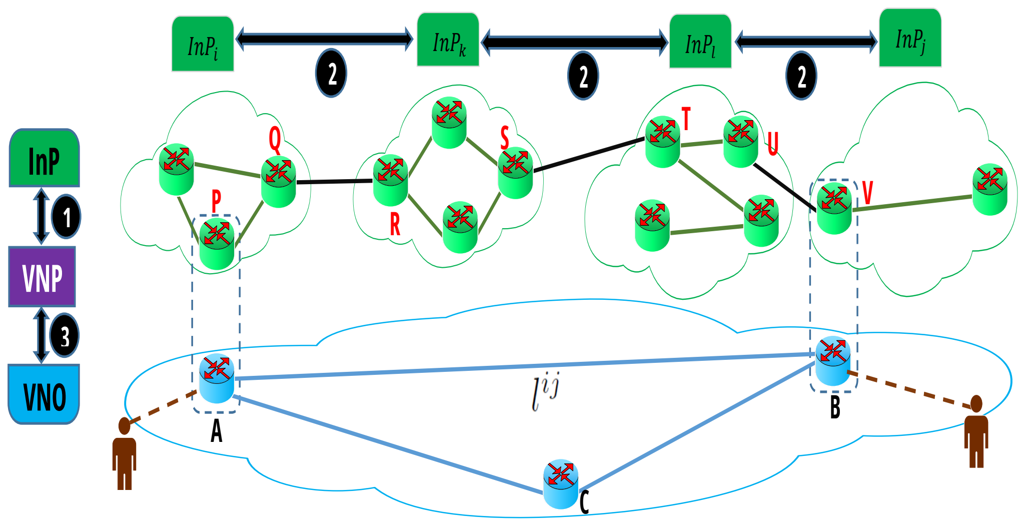


Figure 7-1: Multi-Domain Virtual Network Embedding Problem Formulation

Table 7.1: Network Virtualisation Interfaces

Number	Players	Interface Description
1	VNP/InP	Request and negotiation of virtual resources
2	InP/InP	Setup of inter-domain virtual links
3	VNO/VNP	Virtual network description and request

with a single VNP, it is worth noting that our proposal can be extended to consider a third level of horizontal relationships between two or more VNPs who could, by virtue of different InP policies or resource limit restrictions, have access to different sets of InPs, and thereby have the capacity to re-lease out resources to other VNPs.

7.2.2 Problem Description

Link failures can be managed by either provisioning backup resources, or by attempting to perform re-routing upon failures [26]. While it would be more resource efficient to wait for links to fail and thereafter perform re-routing of the affected paths, re-routing schemes can be time consuming since the availability (or not) of resources to support backup links has to be established at fault time [162].

From the business model described in Section 7.2.1, VNOs provide all their requirements for creating VNs to VNPs, and they (VNOs and VNPs) have SLAs with regard to VN provisioning, for example, in terms of virtual network downtime. We consider that the agreements between VNOs and VNPs involve varying penalties for violating QoS, and that the biggest contributors to QoS violation are substrate link failures, which ultimately lead to virtual link failures. Therefore, to guard against high penalties resulting from QoS violation, a VNP takes decisions with regard to backing up of virtual links. The objective of the VNPs is to maximise its profits by minimising both QoS violation penalties as well as the high expenses from resource backup reservations.

Except for the penalties due to failed links, the overall relationship between VNOs and VNPs is well defined in the state of the art (as presented in Section 1.5), and mainly involves virtual network modelling (see Section 2.4.1) and virtual network embedding. For this reason, this thesis starts after a successful VNE[4]. We concentrate on the interactions between the VNP and InPs after the initial embedding stage, which consist of creating survivable virtual links, by provisioning back-up links for each of the already mapped virtual links, and the negotiation algorithms which provide support to these interactions. The idea is to reserve resources that can be used by virtual links in case of failures in the substrate network. This, however, must be done carefully to avoid that VNPs incur very high costs for resource reservations. This is why, for any virtual link, the VNPs determine a maximum cost that they are ready to incur for provisioning reserve resources.

7.2.3 Work Flow

In order to give a general description of the work flow in the proposed negotiation algorithms, we use Fig. 7-1, and consider that a VNP wants to provision backup resources for the virtual link l^{ij}. We assume that the virtual link l^{ij} has already been mapped, with its two ends A and B being mapped by InPs InP_i and InP_j respectively[5].

The VNP starts by determining an initial set of InPs to which the request can be sent. This initial set of InPs is such that it includes InPs that performed the initial

[4]The initial multi-domain VNE is performed by use of PolyViNE [75].

[5]It is worth noting that while the description in this subsection only considers a single link, the general process will involve identical processes for each of the virtual links that are part of an embedded VN, and subsequently multiple VNs arriving one at a time.

mapping (and/or their direct neighbours) of the virtual link under consideration. For virtual link l^{ij}, the initial set would include the InPs InP_i and InP_j. The procedure, and justification for selecting the InP set is described in 7.3.2. With the InP set determined, the VNP sends the same service request (request to provision backup resources for a given virtual link) to each of the InPs in the set. The request includes the identity of the InPs that are mapping each end of the virtual link.

On reception of a request from the VNP, a given InP begins by determining if it is able to complete the mapping on its own, i.e. if both ends of the virtual link are mapped with in its domain, and it has enough substrate link resources to provision the link. If the InP can perform the mapping on its own, then, it uses the model proposed in 7.3.2 to determine the price, and then sends a proposal to the VNP. However, in the example of Fig. 7-1, InP_i is not able to complete the mapping on its own since one end of the virtual link is mapped by a different InP. In this case, InP_i would forward the request to (its direct neighbour) InP_k.

Whenever an InP receives a forwarded request from one of his neighbours, it starts by ensuring that the inter-domain link connecting them has enough capacity to support the service being requested. If the inter-domain link does not have this capacity, then, the mapping cannot be completed, and the VNP will be informed about the failure. In our case, this means that InP_i must be able to provision link resources from node P (which maps one end of the virtual link), to node R (in the InP where the request has been forwarded). For instance, these resources could be along the substrate path PQR. At this point, since InP_k already has a connection to the node A of the virtual link (through the path PQR), the request issues from InP_k will include InP_k as the "most recent connection to the virtual node" (see subsection 7.2.4). Therefore, the requests forwarded by InP_k will be a provisioning request for a link starting from InP_k to InP_j. Following a similar procedure, InP_l and InP_k will collaborate to create the connection RST, and finally, InP_l and InP_j will create the final path TUV. At this point, InP_j will send back its cost to InP_l, who would, after adding his own cost forward his proposal to InP_k, and so on, until a final mapping proposal is delivered to the VNP. On reception of a proposal, the VNP may accept or reject it based on its own evaluation (see Section 7.3.2). In the rest of this Chapter, we detail the negotiation procedures and the justification of the decision making processes explained in this sub section.

7.2.4 Virtual and Substrate Network Modelling

To achieve inter-domain mapping, InPs should make inter-domain connections to InPs that map the two ends of a virtual link. Considering Fig. 7-1, the two ends A and B of virtual link l^{ij} have been mapped by InPs InP_i and InP_j respectively. For the link l^{ij} to be mapped, all the four InPs must participate in the mapping. Before any InP that is not mapping any of the two end nodes (e.g. InP_k and InP_l) can participate in the mapping, at least an immediate neighbour must have participated in the mapping, or one of the end nodes of the virtual link must have been mapped by its direct neighbour. As an example, for InP_k to connect to virtual node B of virtual link l^{ij}, then InP_l must have a connection to the same via InP_j i.e InP_l should have participated in the mapping. For this reason, in addition to the parameters used to model virtual and substrate networks as described in Section 2.4.1, in this Chapter each virtual link l^{ij} whose ends belong to InP_i and InP_j has to be characterised by an InP, $\text{InP}(l_u^{ij})$, which performed the most recent connection to the virtual node u of the link l^{ij}. This means that since the mapping of a given link may involve more than one InP, the mapping always starts from one end and ends and the other (or starts from both ends and joins in the middle). As an example, the virtual link l^{ij} in Fig. 7-1 is mapped onto the multi-domain substrate path PQRSTUV. Assuming that InP_i makes an initial mapping for the substrate path PQ and forwards the mapping request to InP_k, then according to InP_k, the most recent connection to virtual node A was performed by InP_i. This is necessary because before InP_k attempts to add its own intra-domain mapping RS, it should first connect to InP_i to add the inter-domain path QR. Therefore, the information is used during coalition formation to allow InPs that receive link mapping requests to know which InP to contact so as to perform inter-domain mapping, and hence ensure connectivity. This information does not reveal any private information about $\text{InP}(l_u^{ij})$, except that it participated in the resource coalition, and is only known to an adjacent InP. For the scenario in Fig. 7-1, for InP_k to make a connection to the virtual node B, then $\mathrm{InP}(l_B^{ij})$ must be InP_l. In addition, each virtual link l^{ij} has a length l_x^{ij}, a bandwidth requirement l_b^{ij}, and a QoS value, l_q^{ij} units.

7.2.5 Design Considerations

In this subsection, we describe the design considerations which are the basis for most of the decisions we have made with respect to the proposed negotiation system.

Scalability

The negotiation algorithms used by VNPs and InPs should scale. For this reason, our proposal is fully distributed across the multi-InP domain, with each InP making its own decisions about participation in resource coalition processes. In addition, the communication overhead due to message exchange should be minimised. To achieve this, each InP only forwards the mapping request to its direct neighbours. In addition, the VNP also chooses the initial set of InPs (to which the original resource request is sent) based on the previous mapping of a given virtual link i.e. based on the initial VNE step. In addition, each original resource request has a lifetime beyond which it is not forwarded to other InPs. This avoids flooding the network with a single request which cannot be fulfilled.

Trust

We ensure that InPs both amoung each other, and between them and VNPs can always trust that the price obtained for specific substrate resources is a trustworthy valuation of the resources. This helps to achieve two objectives. On one hand, it helps to avoid endless message exchange between InPs and VNP (in form of contra-proposals), hence contributing to scalability; and on the other hand, it ensures that the InPs do not exploit VNPs by overpricing resources.

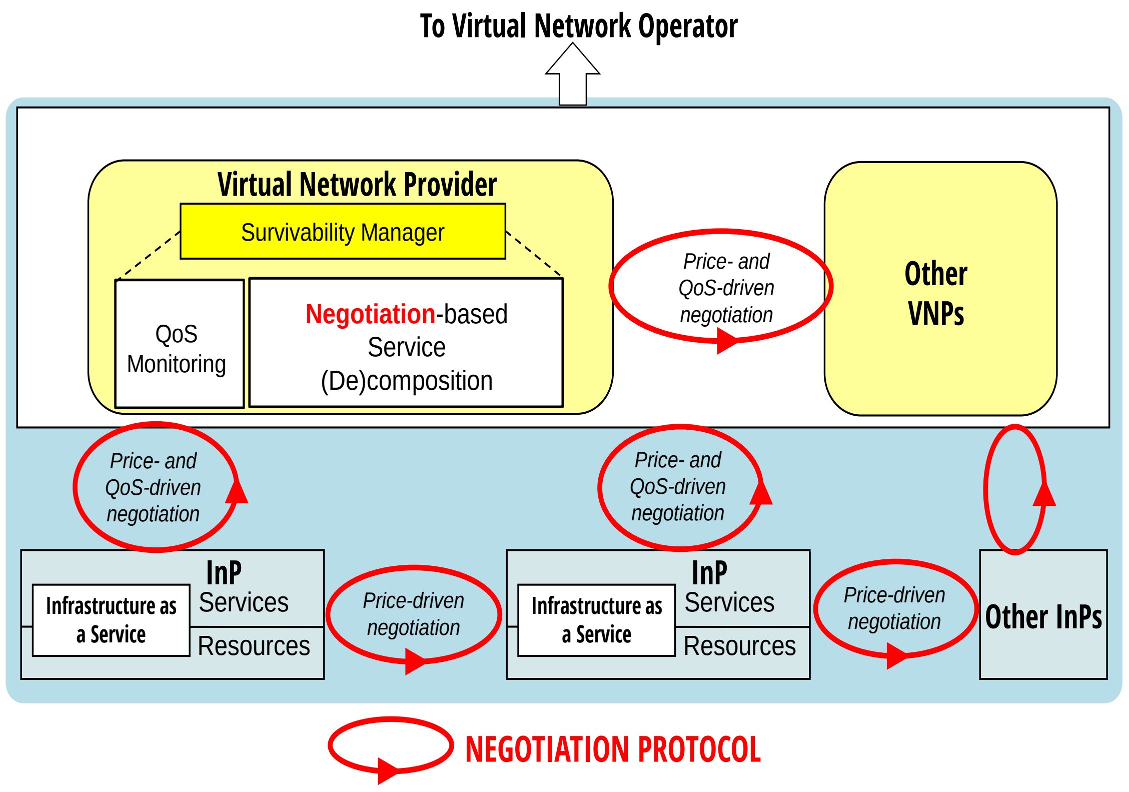

To Virtual Network Operator
Virtual Network Provider
Survivability Manager
QoS Monitoring
Negotiation-based Service (De)composition
Price- and QoS-driven negotiation
Other VNPs
Price- and QoS-driven negotiation
Price- and QoS-driven negotiation
InP
Infrastructure as a Service
Services
Resources
Price-driven negotiation
InP
Infrastructure as a Service
Services
Resources
Price-driven negotiation
Other InPs
NEGOTIATION PROTOCOL

Figure 7-2: Proposed inter-domain negotiation model

Privacy

Our proposal is mindful of the fact that InPs are secretive. As such, it does not require exchange of information that we believe may impact on the privacy of InPs. As an example, during coalition formation, it is not required for InPs to know the level of participation (e.g. in form of which part of a given link is mapped by a given InP) of preceding InPs. All that is needed for any InP is information on whether a direct neighbour performed the most recent mapping of a given virtual link, to be able to collaborate in creating the inter-domain mapping.

7.3 Proposed Negotiation System

Negotiation is a process by which a group of entities communicate with each other so as to try and come to a mutual agreement over some matter [163]. It is a key form of interaction for resolving conflicts in distributed systems in which multiple stake holders are interconnected. Negotiation systems are usually intended to achieve different objectives e.g to avoid negative interactions such as resource and time incompatibilities, or to get mutual benefits; for example in self-interested domains. Any negotiation system should involve three major aspects; negotiation objects, negotiation protocol, and negotiation strategy [164]. In the following subsections, we describe these aspects with respect to our proposal, whose main components are graphically depicted in Fig. 7-2.

7.3.1 Negotiation Objects

Negotiation objects are the range of issues over which agreement must be reached. The main objective of both InPs and VNPs is profit maximisation. However, since a given virtual link may be mapped across multiple InPs, instead of InPs taking the responsibility to initiate virtual link backups, this responsibility is taken on by VNPs. The VNP determines the level of necessity of the backup for any given virtual link (and hence the price that can be paid of backup resources) according to the required link QoS (see QoS Monitoring in Fig. 7-2), while InPs accept or reject proposals from each other only based on prices. For this reason, negotiations between InPs only involve a single attribute - price, while those between InPs and VNPs are multi-attribute, including both price and QoS. In particular, in this proposal, the VNP considers 3 attributes; price, expected delay, and expected data rate. As will be detailed later, the VNP evaluates proposals from any InP by use of a rank resulting from a combination of these 3 attributes.

7.3.2 Negotiation Strategies

This is a way in which a negotiating entity within the protocol (InPs, VNPs in Fig. 7-2) acts in an effort to get the best outcome of the negotiation. They are decision making engines the participants employ to act in line with the negotiation protocol in order to achieve their negotiation objectives and involve e.g. criteria for generating, accepting or rejecting proposals.

VNP Income, QoS Violation Penalty, and Profit

For each unit length, and unit bandwidth demand of a given virtual link, l^{ij}, the VNP earns an income, $Y_{l^{ij}}$ from VNOs for establishing the link, and pays, a fee $C^{p}_{l^{ij}} > 0$ as a cost for primary bandwidth, a fee $C^{s}_{l^{ij}} \geq 0$ as a cost for backup bandwidth, and a fee $Q_{l^{ij}} \geq 0$ as penalty for QoS violations[6]. The initial VNE step is out of the scope of this thesis. We use the proposal in [75] to achieve this, and hence to determine $C^{p}_{l^{ij}}$. In this subsection, we formulate $Y_{l^{ij}}$ and $Q_{l^{ij}}$, and then $C^{s}_{l^{ij}}$ is formulated in the next subsection. We propose $Y_{l^{ij}}$ as a variable chosen uniformly between the ranges in (7.1). The decision to make $Y_{l^{ij}}$ uniformly distributed is to account for other factors (such as link location) which could affect it.

$$Y_{l^{ij}} = U\left[Y_{min} \times \left(\frac{Y_{max}}{Y_{min}}\right)^{l^{ij}_{q}-1}, Y_{max} \times \left(\frac{Y_{max}}{Y_{min}}\right)^{l^{ij}_{q}-1}\right] \tag{7.1}$$

where Y_{min} and Y_{max} are the minimum and maximum income from links with the lowest quality of service. The motivation behind (7.1) is ensuring that virtual links that have high QoS requirements always lead to a higher income for the VNP. Therefore, the total income, Y earned by the VNP from all links it assembles is given by (7.2).

$$Y = \sum_{l^{ij}} \left(Y_{l^{ij}} \times l^{ij}_{x} \times l^{ij}_{b}\right) \tag{7.2}$$

where l^{ij}_{x} and l^{ij}_{b} are respectively, the length and bandwidth of virtual link l^{ij}. In a similar way, the penalty for QoS violation $Q^{s}_{l^{ij}}$ per unit of virtual link length and bandwidth is given by a random variable between:

$$Q^{s}_{l^{ij}} = U\left[Q_{min} \times \left(\frac{Q_{max}}{Q_{min}}\right)^{l^{ij}_{q}-1}, Q_{max} \times \left(\frac{Q_{max}}{Q_{min}}\right)^{l^{ij}_{q}-1}\right] \tag{7.3}$$

where Q_{min} and Q_{max} are the minimum and maximum penalties per unit length and unit bandwidth for links with the lowest quality of service. Needless to say, (7.3) is aimed at ensuring that links that require high QoS also lead to higher penalties when

[6]Once more, we remark that our consideration in this thesis is that QoS violations and hence the penalties resulting from them are only due to link failures.

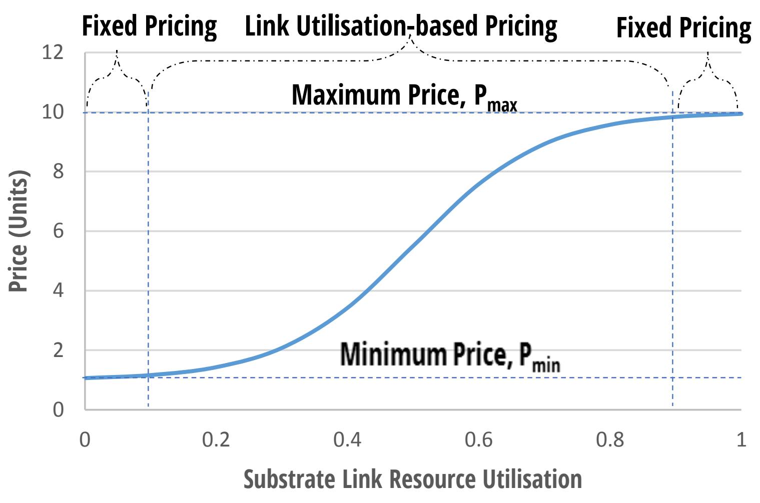


Figure 7-3: Substrate Resources Pricing Utility Function

their QoS is violated. The total cost for all QoS violations is therefore given by (7.4).

$$Q = \sum_{l^{ij}} \left(Q^s_{l^{ij}} \times l^{ij}_x \times l^{ij}_b \right) \tag{7.4}$$

At any point, the profit, $\prod$ of the VNP is given by:

$$\prod = Y - Q - C^s - C^p \tag{7.5}$$

where C^s is the total fee paid for backup resources (see (7.7)), and C^p is that for primary mapping resources. It is worth noting that the income, penalty and profit values defined in (7.1)–(7.5) are per unit time, and hence total values are obtained as integrals of the respective values over the time axis for the considered time periods.

Pricing Model

In order for InPs to generate proposals in response to a mapping request, they should be able to determine prices for their resources. We have chosen to use a hybrid pricing function that is based on the logistic function. This pricing model represents a dynamic pricing scheme that is based on the level of resource utilisation for the substrate network, which is restricted at either end by maximum and minimum allowed prices for the substrate resource in question. This pricing model has advantages over the constant pricing model that has been used in most network virtualisation proposals such as [8], [75] and [26], as it does not only allow prices to reflect network loading (hence encouraging better resource utilisation, and minimising network failures from over loading), but also ensures that resources have reserve prices (to cater to minimum fixed costs), and maximum prices to ensure competitiveness. Therefore, we model the price per unit of flow $P(s)$ on a substrate link s as shown in (7.6).

$$P(s) = l_x^s\left(P_{min}^s + \frac{P_{max}^s - P_{min}^s}{1 + exp\Big(c_1 - c_2u(s)\Big)}\right) \tag{7.6}$$

where P_{min}^s is the minimum acceptable price for s whose resource utilisation level is $u(s)$ and length l_x^s, and P_{max}^s is the maximum allowed price. c_1 is a constant aimed at shifting the pricing function horizontally (and hence affecting the levels of resource utilisation where the *minimum* and *maximum* prices come into effect), and c_2 is a constant that determines the slope of the pricing function (and hence the rate at which pricing changes from minimum pricing to maximum price). Therefore, the total price C^s that should be paid for all the secondary flows i.e. flows over backup resources f_v^s is given by (7.7)

$$C^s = \sum_{all f_v^s} f_v^s P(s) \tag{7.7}$$

The resulting pricing curve is shown in Fig. 7-3. While in general a purely dynamic environment would benefit if all the parameters (P_{max}^s, P_{min}^s, c_1 and c_2) in (7.6) are dynamically adapting to resource availability and InP policies, it is out of the scope of the current work to evaluate these possible advantages in network virtualisation environments. As such, the evaluations in this Chapter consider that for any given substrate link, these parameters are fixed. For the simulations in 7.4.1, for each substrate link, the constants c_1 and c_2 have values normally distributed between 1 & 10 and 10 & 20 respectively. The choice of these ranges is aimed at maintaining the shape of the pricing curve in Fig. 7-3. It should also be remarked that general survivability approach in this thesis is independent of pricing model, and as such, other pricing models could be used.

InP Proposal Selection

To avoid network overload that would result from endless negotiations and counter-proposals, *truthful* pricing of substrate network link resources is an important requirement of our proposal. For this reason, InPs use the sealed-bid second price auction (SBSP) model [165] for proposal selection amoungst each other. In SBSP auctions, a buyer selects the lowest bidder, but pays him the price for the second lowest bid. This forces bidders to be truthful about their pricing, as the winning bidder can never influence the price he or she pays. In this kind of negotiations, the dominant strategy is to bid the true valuation of the good [166]. If only one proposal is received within a given time limit, that proposal is accepted at its price.

VNP Multi-Attribute Evaluation

Since VNPs attempt to backup virtual links that have already been mapped, the price $Price$ for the original mapping is known. This price is used as a benchmark to accept or reject decisions. Depending on the QoS requirement l_q^{ij} of a given virtual link l^{ij}, a reserve price P_{res} is determined using (7.8).

$$P_{res} = \eta \times l_q^{ij} \times Price \tag{7.8}$$

where η is a VNP specific constant aimed at scaling the reserve prices.

As the VNP receives mapping proposals, it must evaluate them with regard to price, link delay, and expected data rate, based on the proposal contents. These three attributes, should be combined to obtain a single evaluation index, E of the proposals. For simplicity, our choice is a linear combination of the attributes. Therefore:

$$E = \alpha(Price) + \beta(NumLinks) + \gamma(Loading) \tag{7.9}$$

where α, β and γ are constants aimed at not only biasing the evaluation to price sensitivity or QoS sensitivity, but also on normalising the three attributes as they all have varying orders of magnitude. Using (7.9), the reserve evaluation E_{res} can also be determined as (7.10).

$$E_{res} = \alpha(Price_{res}) + \beta(NumLinks) + \gamma(Loading) \tag{7.10}$$

For each Service Request, the VNP initialises a $|InP_{set}| \times 5$ rank matrix, $\mathbf{M}$. Each of the 5 columns of $\mathbf{M}$ represents InP, $Price$, $NumLinks$, $Loading$ and E respectively. This matrix is updated whenever the VNP receives a mapping proposal, by populating each row with the: InP, $Price$, $NumLinks$, $Loading$, E; and thereafter sorted to ensure that rows with the lowest value of E are always on top. When the VNP receives replies from all contacted InPs, or when the $Expiry$ time of a given SR is reached, the VNP chooses from M the top row, and its evaluation value, E_{top} is compared with the reserve evaluation E_{res} and a response is sent to the InP concerned according to:

$$\text{Negotiation Result} = \begin{cases} AP(ID, Price) & \text{if } E_{top} \leq E_{res} \\ \\ RP(ID) & \text{if } E_{top} > E_{res} \end{cases}$$

To ensure VNP profitability as well as QoS of mapped links, if VNP cannot accept proposals, instead of considering raising the value η, we propose that the VNP waits a random time dictated by an exponential backoff [167] to resend the SR, with a hope that at one point the substrate network resources would have lower prices.

Determining the InP_{set}

For scalability reasons, the VNP cannot send service requests to all InPs. The set of InPs to which a SR is sent is determined dynamically for each substrate link which is part of the path mapping a given virtual link. If the substrate link is an intra-domain link, this set includes the InP that owns it, as well as all its direct neighbours. The reasoning behind this is that we expect that either the original InP will be able to perform the mapping for the backup resources on its own, or at the very minimum, it will have to form coalitions with its neighbours. Using a similar argument, if the substrate link under consideration is an inter-domain link (like the link l^{ij} in Fig. 7-1), both InPs at the end of the link, as well as both their respective direct neighbours form part of the set of InPs. With reference to Fig. 7-1, this means that the InP set would include all the four InPs, InP_i, InP_k, InP_l and InP_j. In each of these cases, a given InP is added to the set, only if it is not in the $Blacklist$.

7.3.3 Negotiation Protocol

This is a set of rules that governs the interactions between entities, i.e. between InPs and VNPs (see Fig. 7-2). Since Provisioning of the virtual links over multiple InP domains requires collaboration and hence communication between VNPs and InPs and amoung InPs, the protocol defines the flow of messages both amoung InPs, and between InPs and VNPs. After a successful VNE, the Survivability Manager of the VNP (see Fig. 7-2) sends out out multiple and independent service requests (SRs) to InPs. Each SR represents a back up for a given substrate link making up the substrate path for each of the mapped virtual links. Therefore, the Survivability Manager of the VNP starts by replicating each virtual link into as many requests as the substrate links onto which it is mapped. This allows the VNP to provision independent backup paths for each part of the substrate path.

In what follows, we define the 7 messages that form the proposed negotiation protocol, which in turn are graphically represented in Fig. 7-4 in a typical negotiation process.

- **Service Request**$(InP_i,\ InP_j,\ l_b^{ij},\ ID,\ BlackList,\ Expiry)$: A service request (SR) message is sent by either a VNP (to initiate negotiation) or by InP (to forward mapping request) to a given set of InPs to request for mapping of a given virtual link with a unique identification ID. In case it is sent by a VNP, it is the first message in the negotiation process, and initiates the provisioning of the backup for a virtual link. In case it is sent by an InP, this message represents forwarding of a given virtual link backup provisioning. InP_i, InP_j and l_b^{ij} have been defined in Section 7.2.4. $BlackList$ is a set of InPs which cannot participate in the resource coalition. This may be due to policy considerations, or the fact that a given InP has already participated in

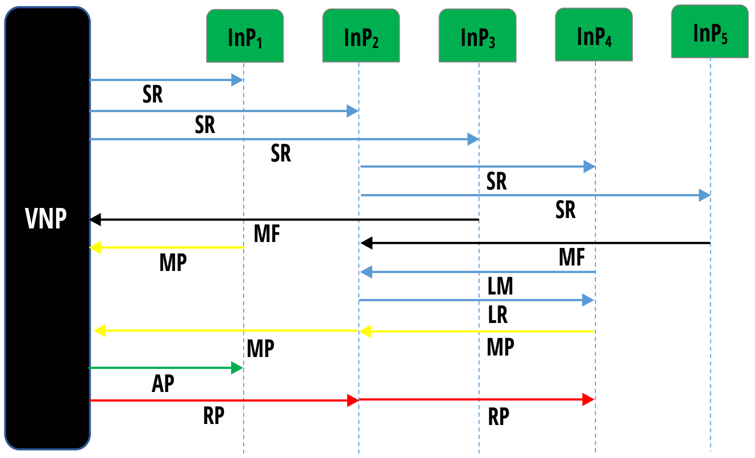


Figure 7-4: Message Exchange between InPs and VNP

the mapping, and hence re-sending a SR to this InP would only increase message exchanges. It is worth noting that for privacy reasons, the actual substrate node mapping the end nodes of the link are not revealed during the message exchanges, instead giving the link ID so that the responsible InP can use it to find the actual start and/or end node. Finally, each SR has a fixed life time represented by $Expiry$. This ensures that the mapping attempts for a given link can only go on for a given time, saving possibilities of flooding the network with mapping requests that cannot be fulfilled.

- **Mapping Proposal**(ID, $Price$, $NumLinks$, $Loading$): After receiving a SR, an InP attempts to perform a link mapping. If the mapping is successful, the InP replies with a mapping proposal (MP) to the sender, giving details of the mapping such as $NumLinks$ which is the total number of substrate links onto which the virtual link is mapped, (which is a measure of the total expected delay), $Loading$ which is the maximum of the percentage loadings of all of the substrate links participating in the virtual link mapping (which is a measure of possible service invocation failures due to diminished substrate link bandwidth, or substrate link failures due to over loading), and the $Price$, which is the cost of the embedding, per unit bandwidth and time.

- **Reject Proposal**(ID): After receiving a MP, a given InP or VNP may reject it either due to policy violations, or being above the reserve cost expectation, or not the winning bid. In this case, a reject proposal (RP) message is sent.

- **Accept Proposal**$(ID, Price)$: If on the other hand a VNP or InP determines that a given MP is the best, or acceptable considering both pricing and policy aspects, an accept proposal (AP) message is sent to the corresponding InP. The $Price$ is specifically useful if the negotiation is between two InPs as this will usually be different (higher) from the one which was proposed in the MP (see Section 7.3.2).

- **Link Map**$(u, SNode, ID)$: In case of links that need to be mapped inter-domain, a link map (LM) message is sent by an InP to $InP(l^{ij}_u)$ that performed the last mapping with respect to the virtual node u, which is at the end of the link ID, specifying the inter domain substrate egress node $SNode$ that it connects to. On reception of this message, the receiving InP determines its node $SNode_x$ onto which u was last mapped, and attempts to create a substrate path between $SNode_x$ and $SNode$. Before the LM message is sent, the sending InP ensures that the inter-domain link connecting them has enough resources to support the virtual link. Using the example of the virtual link l^{ij} in Fig. 7-1 again, and assuming that InP_i has made an initial mapping for the substrate path PQ and forwarded the mapping request to InP_k, then according to InP_k, the most recent connection to virtual node A was performed by InP_i. Therefore, InP_k sends a $\mathbf{LM}(A, R, l^{ij})$. With this, InP_i would already know that this virtual link was mapped onto node Q in its domain, and will therefore attempt to create the inter-domain link QR.

- **Link Result**$(ID, Price, Result)$: The link result (LR) message is sent in response to a LM, after attempting to make a connection to the egress node of the sending InP. $Result$ is a binary value that is 1 if the mapping is successful, and 0 otherwise. If $Result == 1$, then $Price$ is the cost of the link mapping, which the sending InP should add to its mapping cost.

- **Mapping Failed**(ID): The mapping failed (MF) message is sent by any InP to either an InP or VNP when a mapping cannot be provided either due to policy violations or resource constraint restrictions.

The algorithms described earlier for VNP negotiation as well as InP mapping and negotiation are shown in algorithms 8 and 9.

Algorithm 8 VNP Negotiation Procedure

```
1:  Thread1: Receive Proposals
2:  while true do
3:      Receive Proposal for virtual link $l^{ij}$
4:      Determine $E$
5:      Update and Sort $\mathbf{M}(l^{ij})$, determine $E_{top}$
6:  end while
7:  if $E_{top} \leq E_{res}$ then
8:      Accept corresponding proposal, reject others
9:  else
10:     Reject all proposals
11:     Backoff
12: end if
13: Thread2: Survivability Management
14: while Expecting VN Requests do
15:     Get VN and Perform VNE
16:     if VNE Result $\equiv$ False then
17:         Mapping Failed
18:     else
19:         for Each virtual link $l^{ij}$ accepted $VN$ do
20:             Determine $E_{res}$
21:             for Each substrate link $l^s$ mapping $l^{ij}$ do
22:                 Determine $InP_{set}$ for $l^s$
23:                 Initialise $\mathbf{M}(l^{ij}) = [|InP_{set}|][5]$
24:                 for $InP \in InP_{set}$ do
25:                     Send service request to $InP$
26:                 end for
27:             end for
28:         end for
29:     end if
30: end while
```

Algorithm 9 InP Mapping and Negotiation Procedure

```
1: for each virtual node $u$ and $v$ of the virtual link do
2:    Determine latest mapping $InP(l_u^{ij})$ and $InP(l_v^{ij})$
3: end for
4: Initialise $\mathbf{M} = [|InP_{set}|][4]$
5: if $InP(l_u^{ij}) \equiv InP(l_v^{ij}) \equiv$ CurrentInP then
6:    Perform intra-domain link mapping
7:    if Result is True then
8:       Return Mapping Proposal
9:    else
10:      Determine $InP_{set}$
11:      if $InP_{set} \neq \emptyset$ then
12:         Forward Service Request
13:         while Expecting more proposals do
14:            Receive Proposals, Update $\mathbf{M}$
15:            Determine Best Proposal $BP$
16:            Get Price of $2^{nd}$ Best Proposal, $P_{SBP}$
17:         end while
18:         if $\mathbf{M} \equiv$ Empty then
19:            Mapping Failed
20:         else
21:            Accept Proposal $BP$ at price $P_{SBP}$
22:         end if
23:      else
24:         Mapping Failed
25:      end if
26:   end if
27: else
28:   Send LM message to $InP(l_u^{ij})$ and/or $InP(l_v^{ij})$
29:   if LM Result $\equiv$ Success then
30:      Update $InP(l_u^{ij})$ and/or $InP(l_v^{ij})$
31:      Go To Line 4
32:   else
33:      Mapping Failed
34:   end if
35: end if
```

Table 7.2: Simulation Parameters

Parameter	**Substrate Network**	**Virtual Network**
Number of Nodes	$[100, 150]$	$[20, 40]$
Number of Links	From Brite	From Brite
Link Bandwidth	$[200, 500]$	$[100, 200]$
Node CPU	$[100, 200]$	$[50, 100]$
Node Locations	$[0, 250]$	$[0, 250]$

7.4 Performance Evaluation

7.4.1 Simulation Setup

We start by creating a SN topology, and thereafter VN requests arrive one at a time to the SN. The arrival rate is $10s$ and service time $300s$, and both follow Poisson distribution. All evaluations are performed for 1000 VN arrivals. The substrate and virtual network topologies are generated using Brite [110]. The InPs are connected to each other with an inter-domain link with probability of 0.5 and a bandwidth uniformly distributed between 500units and 1000units to form the multi-entity negotiation system. Each of the InP and VNP negotiating entities is implemented in the Java Agent Development Framework (JADE) [168], and the proposed negotiation protocol is implemented as an extension to the ACLMessage [168], which is compliant to the FIPA 2000 specifications [169]. Expiry time of each message is distributed uniformly between 30 and 60s from time the message was created by VNP.

Substrate link minimum price P_{min} is determined as a function of the link length. Specifically, the simulations in this Chapter use the actual link length as the minimum price per unit for the substrate link. Link lengths are determined from the output of Brite. Maximum price P_{max} is determined as $(P_{min} + P_{dev})$, where deviation P_{dev} is chosen from a uniform distribution between 0 and P_{min}.

For each substrate link, the mean time between failures (MTBF) and mean time

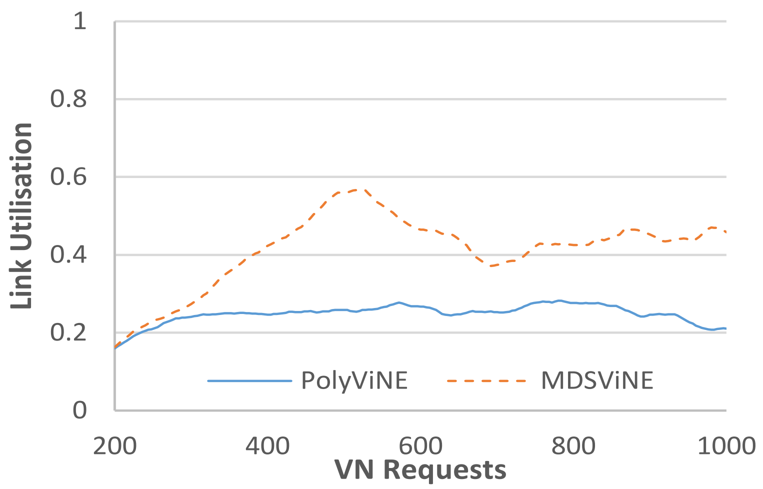


Figure 7-5: Average Substrate Resource Utilisation

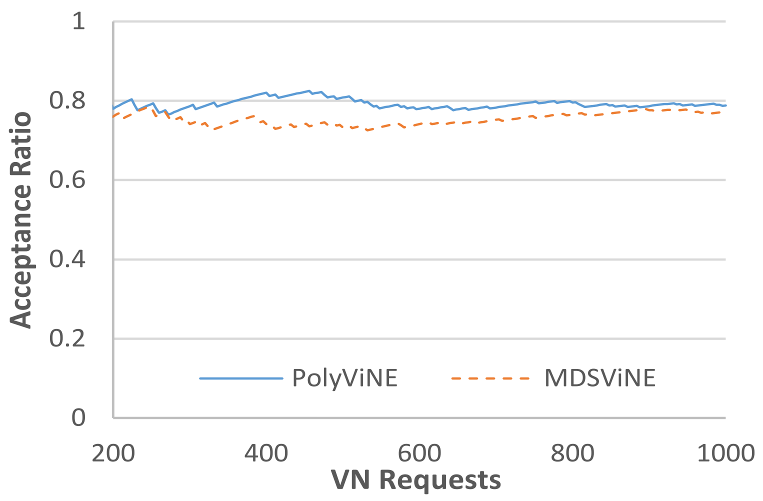


Figure 7-6: Virtual Network Acceptance Ratio

to repair (MTTR) used in our simulation are based on a characterisation of link failures in a real ISP backbone performed in [161] and both follow a Weibull distribution [170] with shape parameter, $\beta = 0.5$, and scale parameter, $\alpha = 0.4$ and 0.2 for MTBF and MTTR respectively. The quality of service l_q^{ij} values for each virtual link are uniformly distributed between 1 and 5 where 1 is the lowest QoS and 5 the highest. The rest of the substrate network and virtual network parameters are uniformly distributed between values shown in Table 7.2.

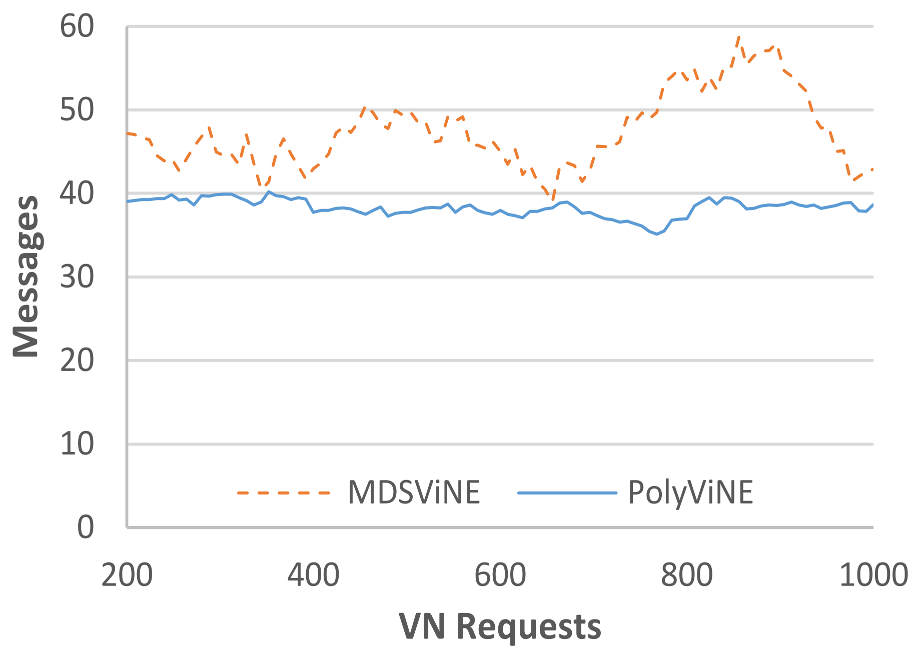


Figure 7-7: Variation in Message Overhead

7.4.2 Comparison with other approaches

We compare the multi-domain survivable VNE (MDSViNE) proposal in this Chapter with PolyViNE [75] which performs multi-domain embedding without consideration for survivability. We also note that it is this approach which is used for performing the initial VNE before our survivability proposals start. Node mapping is performed using the greedy approach in [112] while link mapping is performed by formulating the problem as a multicommodity flow (MCF) [27] and solving the resulting linear program using CPLEX12.6 [111].

7.4.3 Performance Metrics

While the simulations in this Chapter have considered a range of aspects from message overhead, VN acceptance ratio, to efficiency of resource utilisation, the main performance evaluation metric is the profitability of the VNP. As already defined in Section 7.3.2, this is a measure of the difference between the total income of the VNP and the total expenses for initial mappings, back up resource reservations, and penalties due to QoS violations.

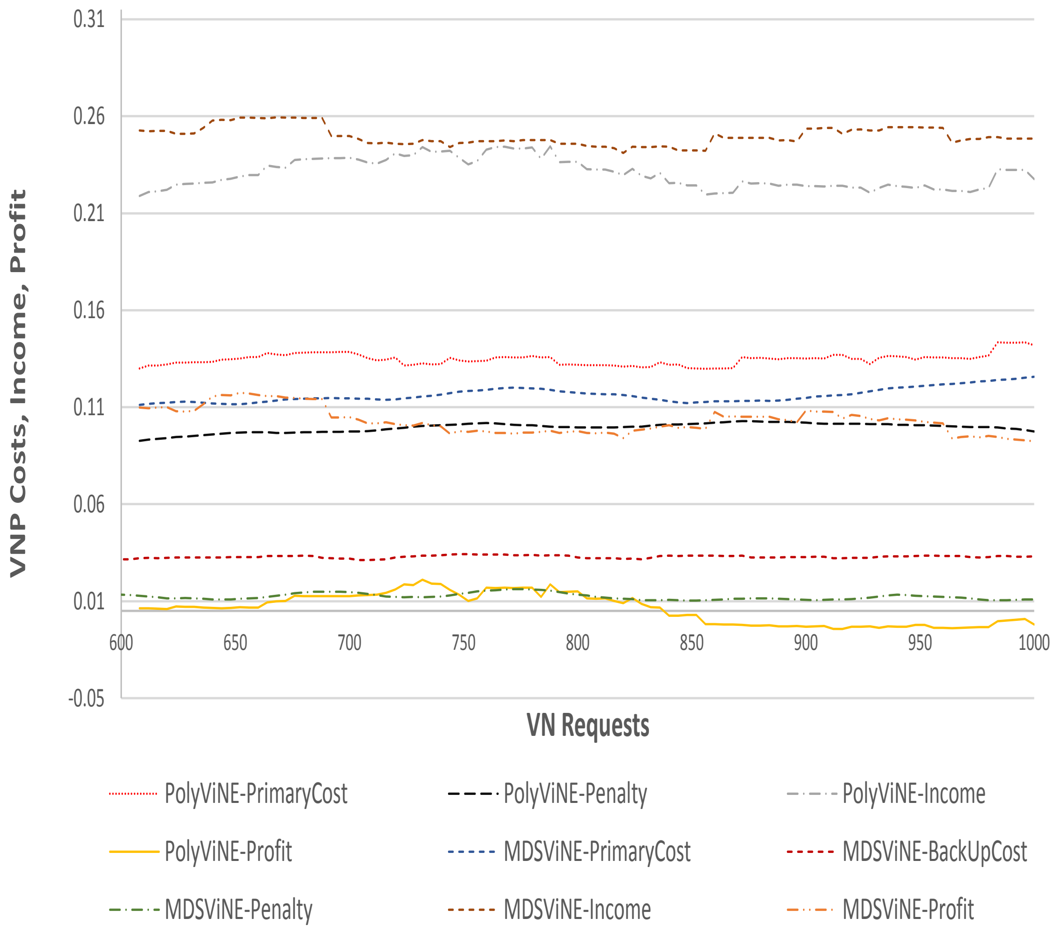


Figure 7-8: VNP Average Costs, Income, Profits

7.4.4 Discussion of Results

Fig. 7-5 shows that MDSViNE has a better utilisation of substrate network resources compared to PolyViNE. This is is expected since MDSViNE commits some of the link resources for failures. For the same reason, since MDSiNE has less free resources to accept resource requests, we note in Fig. 7-6 that PolyViNE has a marginally better acceptance ratio for VN requests. In Fig. 7-7, we observe that PolyViNE involves an exchange of fewer messages compared to MDSViNE. This is also expected, since MDSViNE includes PolyViNE in the first stage, and also because PolyViNE is not making any negotiations for backups. It can also be noted that, in general, the number of messages also increase with substrate resource utilisation as seen by comparing both Fig. 7-5 and 7-7. This can be explained by the need for InPs to forward a

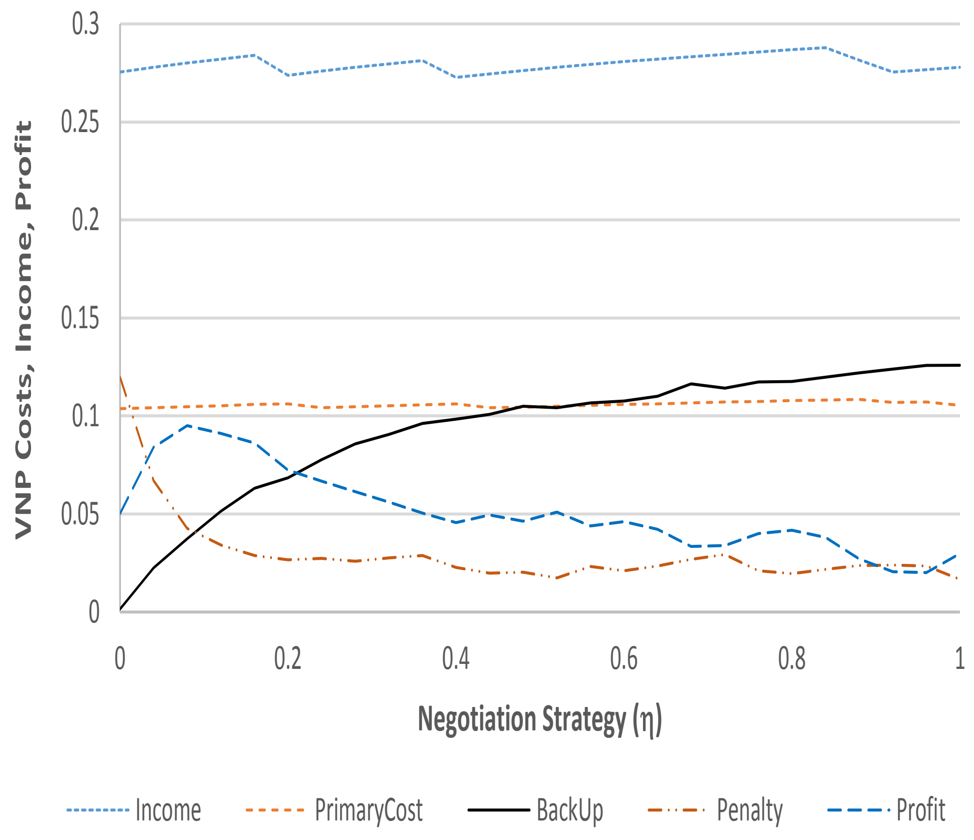


Figure 7-9: Effect of Negotiation Strategy

service request to many InPs possibly due to mapping resource shortages.

In Fig. 7-8 we show the VNP costs (primary mapping costs, back-up costs, QoS violation penalties), income and profit. As it can be noted, while both approaches have on average similar primary resource costs, PolyViNE incurs more costs as a result of QoS violation as compared to the total costs of both back-up resources and QoS violations by MDSViNE, hence ensuring that better profitability for VNPs that are survivability-aware.

In Fig. 7-9, we evaluate the VNP negotiation strategy[7], and its effect on profitability. We observe that by adjusting from being more profitability-biased $\eta = 0$ to being more QoS-aware $\eta = 1$, while the cost of resource back-ups keep rising, the QoS violation penalties decrease sharply and then remain constant, implying that it pays

[7]These simulations are repeated 10 times for each case, and what we present here are average values.

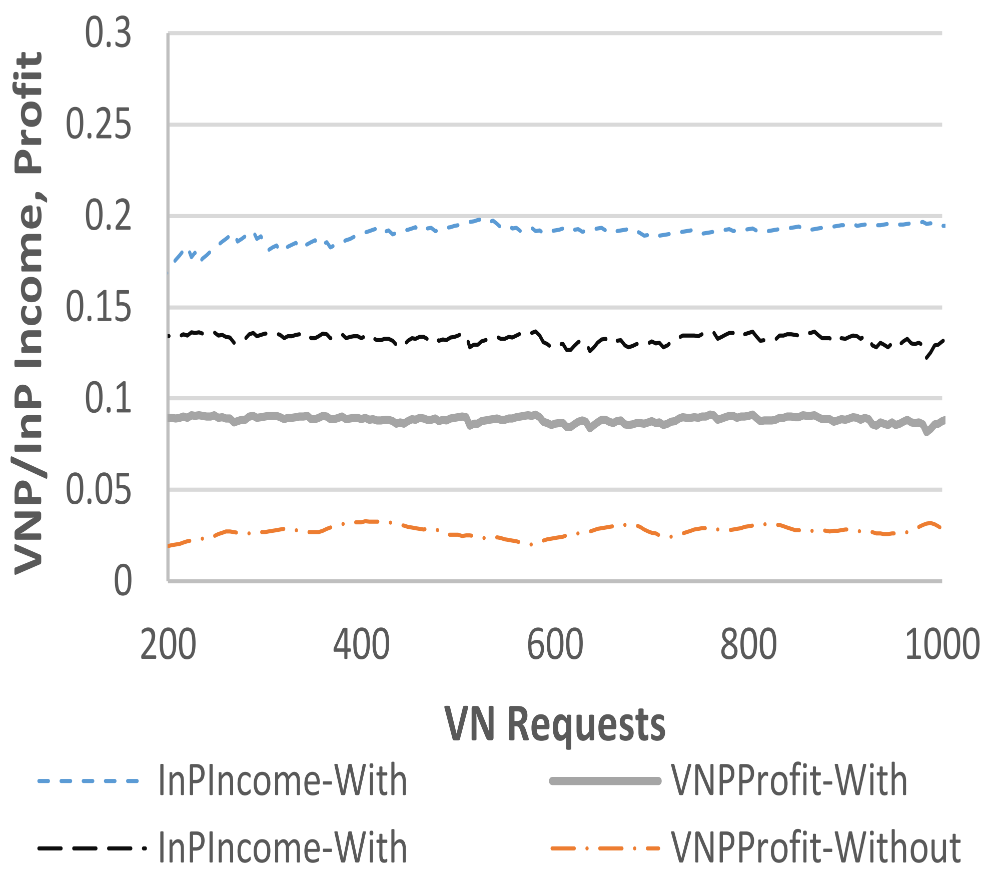


Figure 7-10: Effect of Pricing Model

for the VNP to use a lower value of η. This is attributed to the fact that since the VNP does not have a immediate requirement for backup resources, it may chose to periodically send the resource back up request until it receives a proposal below its reserve evaluations.

Finally, Fig. 7-10 shows the effect of dynamic pricing on the income and profitability of InPs and VNPs. Its clear that a dynamic pricing model leads to better profitability for both InPs and VNPs. For the VNPs a dynamic pricing scheme allows them to wait for periods when resource pricings are low so as to request for resource backups, and this also gives more income to the InPs by avoiding that resources are left idle when they could be priced lower in periods of lower load.

7.5 Conclusion

This Chapter has proposed a distributed, competitive and dynamic multi-attribute negotiation solution that allows for a survivability-aware virtual network embedding in multi-domain environments. We represented each infrastructure provider and virtual network provider as agents with conflicting objectives. While the VNP agents aimed at achieving survivable embeddings at lower costs, the InP agents tried to maximise their profits, keeping their resource deployment and pricing strategies private.

We proposed that after a given multi-domain VNE, the VNP should initiate negotiations for provisioning of backup paths for each mapped virtual link. To this end, the VNP started by determining a set of InPs to which the mapping request is sent. The objective of restricting the number of agents that receive mapping requests is to minimize communication message overhead. Then upon receiving a mapping request, an InP either performs the mapping on its own or if this cannot be done, the request is forwarded to other InPs. For each InP, after a partial or complete mapping, the substrate resources involved in the mapping were dynamically priced using a pricing model that bases on available substrate resources to determine resource prices, and a proposal forwarded to the VNP. The VNP used knowledge of possible QoS violation penalties and actual costs of backup resources to either accept or reject the all the received mapping proposals. Finally, for all virtual links for which a backup provision was not effected immediately after the initial embedding, other attempts are made to do this following a back-off mechanism.

We compared our proposal to a state-of-the-art multi-domain embedding approach, and through simulations, we have confirmed that our approach improves the utilisation level of substrate link resources by about 20%, while achieving a comparable acceptance ratio and minimal extra message exchange. We have also shown that our approach improves the profitability of InPs by over 1000%, and that the pricing model improves the InP profit by about 200%. The results in this Chapter confirm hypothesises 5 and 6 as outlined in Section 1.2.

Chapter 8

Conclusions and Future Work

8.1 Introduction

One of the fundamental requirements in network virtualisation is the assignment of physical network resources to virtual networks. Because it determines how many virtual networks can share a given set of physical resources at any given point, resource management directly affects the profitability and hence attractiveness of network virtualisation to infrastructure providers.

This thesis set out to make contributions to this very important part of network virtualisation. To this end, the resource management problem was split into three clear sub-problems; (1) virtual network embedding, (2) dynamic resource allocation, and (3) virtual network survivability. For each of these sub-problems, the state-of-the-art was analysed with an aim of determining areas that needed more research.

After, comprehensively analysing state-of-art approaches, it was observed that; (1) there was need to not only perform the embedding in one shot, but also to devise a way of improving the time complexity of the formulated problem, as it is computationally intractable, (2) most state of the art approaches were static, i.e. allocating a fixed amount of resources to virtual networks through out their lifetime, which would potentially be wasteful and hence lead to inefficient resource utilisation, and finally, that (3) current approaches to virtual network survivability have only focussed on intra-domain virtual network environments, yet practical reasons mean that virtual networks can span more than one domain.

For each of these three identified gaps in the state-of-art approaches, this thesis set out to answer the following respective questions:

1. Can virtual network embedding be performed in one shot without incurring the exponential computational time of the optimal formulation ?

2. Can resources be dynamically allocated to virtual nodes and links without a negative impact on their QoS ?

3. What would be the impact of a dynamic pricing approach and a survivability-aware resource allocation approach on the profitability of infrastructure providers ?

These questions led to the formulation of the six hypothesises presented in Section 1.2, all of which have been confirmed in this thesis. In the rest of this Chapter, we summarise the main contributions and results with regard to each of the three sub-problems identified, provide answers to the above respective questions, and finally, an outlook for future work in line with each of the sub-problems. We conclude the chapter and thesis by giving the expected practical applicability of the thesis results, followed by a list of publications and other contributions to the scientific community.

8.2 Summary of Results

The main results of this thesis are chapter specific and were summarised at the end of each of the Chapters 3 - 7. In this section, we summarise these results, grouped into the three sub-problems which have been the subject of this thesis. In what follows, each sub-section is divided into three parts, the main research question is re-stated, the contributions of the thesis are presented, and the answer resulting from them finally given.

8.2.1 Virtual Network Embedding

Question

Can virtual network embedding be performed in one shot without incurring the exponential computational time of the optimal formulation ?

Thesis Contributions

This thesis has applied column generation to the one shot virtual network embedding problem. The objective was to achieve the embedding efficiency that results from a one-shot mapping, while greatly enhancing its time complexity. Two mathematical programs, a primal and dual, were formulated. A third mathematical program representing an initial solution was also formulated. Solving these three programs in turn yielded the one-shot virtual network embedding.

In this regard, the main contributions of this thesis are as follows:

- A near optimal one-shot virtual network embedding approach that improves substrate resource utilization compared to existing solutions.
- A path generation-based approach that significantly improves the time complexity of the solution compared to the optimal solution.

To the best of our knowledge, this is the first path-based mathematical programming solution to the one shot virtual network embedding problem. It is also the first application of path generation to a multi−commodity flow problem in which the source and end nodes of each commodity must also be determined.

Answer

Simulation results confirmed that:

- Our proposal does not only achieve an average acceptance ratio close to that obtained by the optimal solution, but also outperforms state-of-the-art solutions in this regard. It was also observed that the proposed approach achieves a better utilization ratio for substrate node and link resources compared to other solutions.

- Importantly, it was established that for substrate networks up to about 60 nodes, our proposal had time computations similar to state-of-art approaches. While the complexity slightly diverged for bigger substrate networks, it still remained comparable. In all these circumstances however, our proposal significantly out performed the optimal solution.

8.2.2 Dynamic Resource Allocation

Question

Can resources be dynamically allocated to virtual nodes and links without a negative impact on their QoS ?

Thesis Contributions

For dynamic resource allocations, this thesis proposed machine learning techniques, with the objective of incorporating self-management capabilities in the allocation of physical resources to virtual nodes and links. Three incremental approaches were proposed. The first based on reinforcement learning, the second using artificial neural networks, and the final one neuro-fuzzy systems. The overall idea in each of these proposals was to take advantage of idle virtual network resources by allocating then to other virtual networks that needed them. This had to be done while at the same time ensuring that the QoS requirements of virtual networks were not being violated. For each of the three approaches proposed in this respect, the following contributions were made by this thesis:

1. *Reinforcement Learning:* A distributed reinforcement learning algorithm that allocates resources to virtual nodes and links dynamically and an initialisation scheme that biases the learning policy to improve the rate of convergence.

2. *Artificial Neural Networks:* A neural network algorithm that improves the sensitivity of the reinforcement approach, due to perception and action on continuous spaces. The use of reinforcement learning to training the neural network, doing way with the usual need of training examples in typical neural network applications.

3. *Neuro-Fuzzy Systems:* An adaptive neuro-fuzzy system in which a hybrid learning mechanism uses supervised learning to initialise the rule base and then uses unsupervised learning to adapt the rule base and fuzzy sets of each rule to achieve efficient resource allocation. A cooperation scheme that allows the substrate network agents to coordinate their actions so as to avoid conflicts and to share their knowledge so as to enhance their learning speed and improve action selection efficiency.

Answer

In addition to the respective improvements of each approach over the preceding one, the major results from these contributions can be summarised as below:

- Opportunistically using virtual network resources significantly improves the acceptance ratio of virtual networks, and the overall number of virtual networks that can be hosted on a substrate network at any point.

- The enhanced acceptance of virtual network requests can be obtained without negatively impacting the QoS of the already embedded virtual networks, when the allocating entities have learnt optimal resource allocation policies.

To the best of our knowledge, this is the first application of artificial intelligence-based techniques to the self-management of resources in network virtualisation environments. It is also the first QoS-aware proposal for opportunistic use of virtual network resources.

8.2.3 Virtual Network Survivability

Finally, after observing that most current approaches to virtual network survivability have only focussed on intra-domain virtual network environments, this thesis proposed a system of negotiating entities. While the overall objective is to minimise the penalties resulting from QoS violations on the side of InPs, it inevitably also ensures that the VNP do not suffer QoS violations.

Question

What would be the impact of a dynamic pricing approach and a survivability-aware resource allocation approach on the profitability of infrastructure providers ?

Thesis Contributions

In this regard, the major contributions of this thesis are as follows: a negotiation protocol that ensures virtual network survivability with minimum communication message overhead; VNP and InP negotiation strategies that minimise QoS violation penalties, hence ensuring both VNP and InP profitability; and a dynamic substrate resource pricing model that ensures efficient utilisation of resources. To the best of our knowledge, this is the first endeavor to propose survivability in independent multi-domain virtual network environments.

Answer

We have confirmed that our approach improves the utilisation level of substrate link resources, while achieving a comparable acceptance ratio and minimal extra message exchange compared to a state-of-art approach. We have also been able to confirm that provisioning resources for survivability in multi domain virtual networks improves the profitability of InPs by over 10 times, and that the dynamic pricing model improves the InP profit 2-fold.

8.3 Future Work

The scale of the resource management problem in network virtualisation is extensive and multifaceted even when considered at the level of each of the three identified sub-problems. What follows is our expectations and recommendations for future work in each one of them.

8.3.1 Virtual Network Embedding

There is still more work that can be done to further enhance the computational complexity of the one-shot virtual network embedding. One possibility is to start by formulating linear relaxations of the mathematical programs formulated in Chapter 3, and then proceeding to propose heuristics to determine final solutions. The challenge in this aspect would be ensuring that the solution from a linear relaxation can be adapted to ensure that it still meets the embedding constraints such as capacity or node location. One possible direction to take in this foray would be to explore possible combinations between optimisation theory and some techniques from artificial intelligence such as particle swarm optimisation. Future research in this area will study the feasibility of this solution approach

8.3.2 Dynamic Resource Allocation

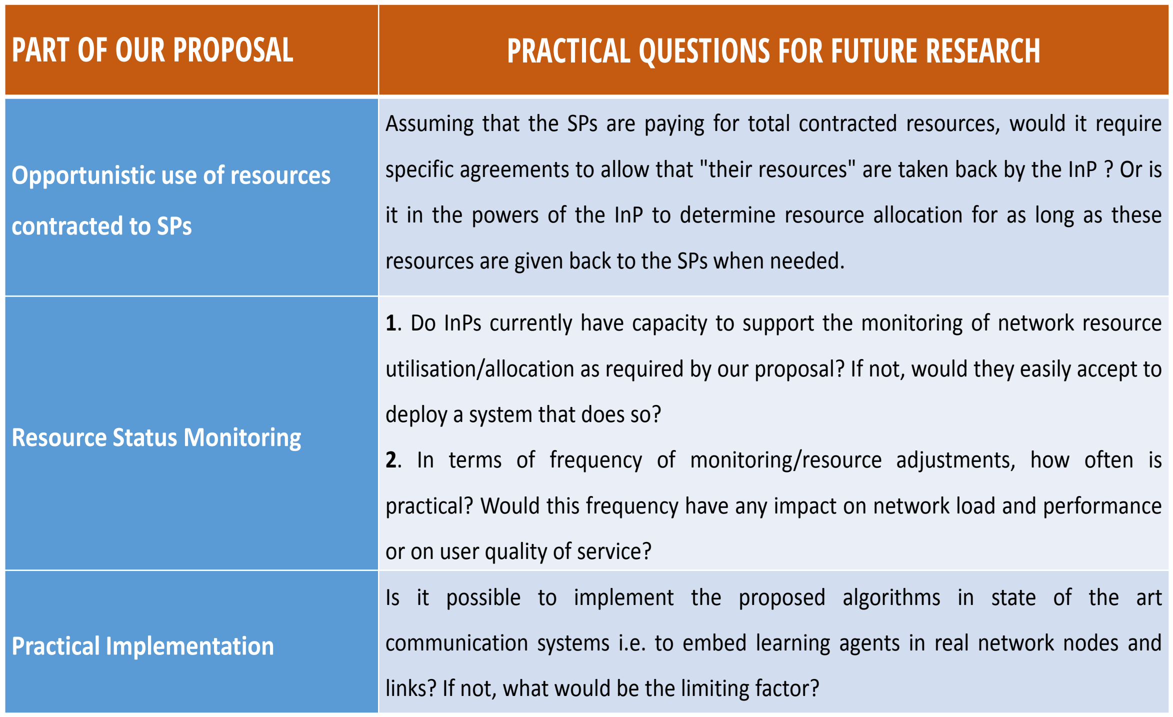
One issue that has appeared across all the three variants of the dynamic resource allocation proposals in this thesis have been the slightly bad performance at the beginning of the learning process. Indeed, most machine learning techniques usually suffer from low convergence speed. While this thesis attempted to initialise the learning to enhance the convergence speed, more work can still be done in the same direction. Specifically, it is possible to create an initial offline learning step, such that actual online resource allocations start from optimal policies. It will also be interesting to extend the dynamic resource allocation to multi-domain virtual networks, in which, instead of the link agents only cooperating, they compete amongst each other and their respective secrecy considerations. It is also worth some effort to study how the rules/policies/weights in each agent vary in a given agent over time, with the aim of determining possibilities for enhancing the effect of agent cooperation.

8.3.3 Virtual Network Survivability

This thesis only considered single substrate link failures. Future works will involve extending the approach to consider multiple link failures, and hence node failures. We will also extend the interfaces between virtualisation players to include scenarios where VNPs are able to use policy-based considerations to re-lease resources to other VNPs. In addition, we intend to explore possible improvements in the mechanism of combining the negotiation attributes by, say, using a non-linear combination. We will also study possibilities of making both VNP and InP entities more autonomic to allowing them to take more decisions, say, changing minimum and maximum prices by considering policies and resource utilisation, by employing learning techniques, ensuring scalability and convergence to optimal results.

8.3.4 Other Related Areas

In addition to the three sub-problems considered in this thesis, there are still several key challenges in the management of resources in network virtualisation. For example, current virtualisation systems assume a cooperative environment where all virtual network operators, and users collaborate. As virtualisation matures and becomes more widely deployed, it is important to consider the effects of and possible defenses against malicious operators and users, especially given that these users share the same medium. It will also be important to defined more elaborate frameworks to ensure and manage QoS especially in multi-domain virtual network environments. Finally, recent research has introduced new concepts such as software defined networking and network function virtualisation, both of which have requirements similar to those of network virtualisation, and specifically to virtual network embedding. Resource management in these future Internet technologies will also become important future research areas.



PART OF OUR PROPOSAL	PRACTICAL QUESTIONS FOR FUTURE RESEARCH
Opportunistic use of resources contracted to SPs	Assuming that the SPs are paying for total contracted resources, would it require specific agreements to allow that "their resources" are taken back by the InP ? Or is it in the powers of the InP to determine resource allocation for as long as these resources are given back to the SPs when needed.
Resource Status Monitoring	**1**. Do InPs currently have capacity to support the monitoring of network resource utilisation/allocation as required by our proposal? If not, would they easily accept to deploy a system that does so? **2**. In terms of frequency of monitoring/resource adjustments, how often is practical? Would this frequency have any impact on network load and performance or on user quality of service?
Practical Implementation	Is it possible to implement the proposed algorithms in state of the art communication systems i.e. to embed learning agents in real network nodes and links? If not, what would be the limiting factor?

Figure 8-1: Practical Questions to be Considered for Future Research

8.4 Practical Application

The contributions of this thesis may be used in existing network virtualisation projects and/or test beds so as to achieve autonomic and efficient management of substrate network resources. They could also be used by telecommunications infrastructure providers. There is also a growing need for efficient resource management in Cloud computing [171] so that different applications can be dynamically allocated resources from the pool [172]. In this regard, our dynamic resource allocation proposals can be adapted and/or extended to fit into the needs of Cloud service providers.

In particular, as an initial step, due to our participation in some European collaborative projects such as Flamingo [173], we can have access to virtualisation testbeds such as the virtual wall [174]. Our next more practical steps in this regard will be to see how our proposals, especially the dynamic resource allocation algorithms can be used in such testbeds.

Finally, with the help of Flamingo[1], we are already attempting to validate the proposals of our dynamic allocation schemes. This process involves discussing various aspects of our proposal with a real operator, such as an infrastructure provider. The idea is to determine the requirements towards having such proposals applied in real environments. To this end, we have already formulated the questions shown Fig. 8-1, that will be discussed with a real operator in Europe (yet to be determined). Based on the feedback, two possible actions will be taken; (1) improve our proposals based on their advise, (2) implement a simple prototype based on our proposals. These will all be part of future work.

[1]Part of the contributions of this thesis were done with in the Flamingo EU project.

Publications and Contributions to Scientific Research

Journals

- Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Meng Shen, Ke Xu, Kun Yang, “A Neuro-Fuzzy Approach to Self-Management of Virtual Network Resources”, **Submitted to** Journal of Expert Systems With Applications, (February 2014). Code: **NFSA**.

- Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Raouf Boutaba, “Path Generation-based Virtual Network Embedding”, **Submitted to** IEEE/ACM Transactions on Networking (March 2013). Code: **PaGeViNE**.

Conferences

- Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Javier Rubio-Loyola, Ramón Agüero, “Survivability-oriented Negotiation Algorithms for Multi-domain Virtual Networks”, **Submitted to** 10th International Conference on Network and Service Management (CNSM) 2014. Code: **SONA**

- Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Maxim Claeys, Jeroen Famaey, Filip De Turck, “Neural Network-based Autonomous Allocation of Resources in Virtual Networks”, European Conference on Networks and Communications (EuCNC), 2014. Code: **NNAA**

- Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, “Contributions to Efficient Resource Management in Virtual Networks”, IFIP 8th International Conference on Autonomous Infrastructure, Management and Security (AIMS), 2014. Code: **CERM**

- Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Maxim Claeys, Steven Latre, Filip De Turck, “Design and Evaluation of Learning Algorithms for Dynamic Resource Management in Virtual Networks”, IEEE/IFIP Networks Operations and Management Symposium, (NOMS), 2014. **Best Student Paper Award.** Code: **DELA**

Workshops

- Rashid Mijumbi, Juan Luis Gorricho and Joan Serrat, "Learning Algorithms for Dynamic Resource Allocation in Virtualised Networks", Management of Large Scale Virtualized Infrastructures: Smart Data Acquisition, Analysis and Network and Service Management in the Future Internet, 2014. **Invited Extended Abstract**. Code: **DARVN**

- Rashid Mijumbi, Joan Serrat and Juan Luis Gorricho, "Multi-Agent Based Resource Allocation in Next Generation Virtual Networks", Barcelona Forum on Ph.D. Research in Communication and Information Technologies, 2012. Code: **MARA**

- Rashid Mijumbi, Joan Serrat and Juan Luis Gorricho, "Autonomic Resource Management in Virtual Networks", Scalable and Adaptive Internet Solutions (SAIL) Summer School Work in Progress Session, Santander, Cantabria, 2012. **Best Work in Progress Award**. Code: **ARMVN**

Participation in Research Projects

- Connectivity as a service. Access of the Future Internet (COSAIF). Funded by Ministerio de Economa y Competitividad (TEC2012-38574-C02-02). Period: Jan 2013 - Dec 2015.

- Management of the Future Internet (Flamingo). Network of Excellence, Funded by European Union contract number: 318488. Period: Nov 2012 - Oct 2016.

- End-to-end Virtual Resource Management across Heterogeneous Networks and Services (EVANS). Funded by European Union contract number: 269323. Period: May 2011- April 2014.

- Service Delivery and Service Level Management in Grid Infrastructures (gSLM). Funded by European Union contract number 261547. Period: Sep 2010 Aug 2012.

- Cognitive, Cooperative Communications and autonomous SErvice Management (C3SEM). Funded by Spanish Ministry of Science and Innovation (TEC2009-14598-C02-02). Period: Jan 2010 - Dec 2012.

Contributions to Technical Documents

- Anna Sperotto et. al., “Network and Service Monitoring”, D5.1 Flamingo EU Project, October-2013.

- Gabi Dreo Rodsek et. al., “Automated Configuration and Repair”, D6.1 Flamingo EU Project, October-2013.

- Radhika Garg et. al., “Economic, Legal and Regulative Constraints”, D7.1 Flamingo EU Project, October-2013.

- Zheng Hu et. al., “Reports on Vertical Management of Virtualised Resources”, D3.1 and D3.2 of Project EVANS, September-2012 and November-2013.

- Ning Wang et. al., “Reports on Horizontal Management of Virtualised Resources”, D4.1 and D4.2 of Project EVANS, September-2012 and November-2013.

- Joan Serrat et. al., “SLM model and use cases”, D4.2 of Project gSLM, September-2011.

Invited Talks

- “Learning Algorithms for Dynamic Resource Allocation in Virtualised Networks”, Workshop on Management of Large Scale Virtualized Infrastructures: Smart Data Acquisition, Analysis and Network and Service Management in the

Future Internet, co-located with European Conference on Networks and Communications (EuCNC), Bologna, Italy. June 2014. (Invited).

- "Application of Learning Techniques to Resource Management in Virtual Networks", Future Internet Technologies, Tsinghua University, Beijing, PR China. December 2013.

Research Visits

- March 22, 2014 – May 02, 2014. Information Technology Laboratory, Center for Research and Advanced Studies of the National Polytechnic Institute of Mexico (CINVESTAV), 87130 Ciudad Victoria, Tamaulipas, Mexico.

- October 02, 2013 – January 25, 2014. Tsinghua University, 100084, Beijing, PR China.

- May 6, 2013 – May 18, 2013. Ghent University – iMinds, B-9050 Ghent, Belgium.

Awards

- Best Student Paper Award: IFIP/IEEE Network Operations and Management Symposium, (NOMS), 2014, Krakow, Poland.

- Student Travel Grant: IFIP/IEEE Network Operations and Management Symposium, (NOMS), 2014, Krakow, Poland.

- Best Work in Progress Award: Scalable and Adaptive Internet Solutions (SAIL), 2012, Santander, Cantabria.

- FPI Grant (FPI2010): Ministry of Economy and Competitiveness (MINECO), Government of Spain, 2010 - 2014, Madrid, Spain.

Other Services

- Reviewer, Journal of the Network and Systems Management, Springer.
- Reviewer, 14th IEEE/IFIP Network Operations and Management Symposium (NOMS 2014), 5- 9 May 2014, Krakow, Poland.
- Local Organising Committee, 7th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2013), June 25-28, 2013, UPC Barcelona, Spain.
- Reviewer, IFIP/IEEE International Symposium on Integrated Network Management, 2013.
- Reviewer, 9th IEEE International Workshop on Managing Ubiquitous Communications and Services, 2012.

Appendix A

Notation

Unless otherwise defined within the thesis, the symbols below have the following meanings.

α	Learning Rate
λ	Discount Factor
π	Profit
n_a	Node Agent
l_a	Link Agent
ϵ	Probability with which random selecting actions
G_v	Virtual Network Graph
G_s	Substrate Network Graph
N_v	Set of Virtual Nodes
L_v	Set of Virtual Links
N_s	Set of Substrate Nodes
L_s	Set of Substrate Links
i, j	Virtual Nodes
l_{ij}	Virtual Link Connecting Virtual Nodes i and j
D_{ij}	Delay of Virtual Link l_{ij}
B_{ij}	Bandwidth of Virtual Link l_{ij}

u, v	Substrate Nodes
$Q(s, a)$	State-Action Value Function
s	State
a	Action
l_{uv}	Substrate Link Connecting Substrate Nodes u and v
D_{uv}	Delay of Substrate Link l_{uv}
B_{uv}	Bandwidth of Substrate Link l_{uv}
$L_i(x, y)$	Location of Virtual Node i
$L_u(x, y)$	Location of Substrate Node u
Q_i	Queue Size of Virtual Node i
Q_u	Queue Size of Substrate Node u
s_p	Previous State
s_n	Next State
a_p	Previous Action
r_p	Previous Reward

Appendix B

Acronyms and Definitions

ANN	Artificial Neural Network
BW	BandWidth
CPU	Central Processing Unit
DRA	Dynamic Resource Allocation
DRM	Dynamic Resource Management
EU	European Union
IaaS	Infrastructure as a Service
ILP	Integer Linear Programming
InP	Infrastructure Provider
IP	Internet Protocol
ISP	Internet Service Provider
LP	Linear Programming
MAS	Multi Agent Systems
MCF	Multi Commodity Flow
MP	Mathematical Programming
NFS	Neuro-Fuzzy System
NN	Neural Network
NVE	Network Virtualisation Environment
OLA	Operational Level Agreement
PLR	Packet Loss Rate

QoS	Quality of Service
RL	Reinforcement Learning
SBSP	Sealed Bid Second Price
SLA	Service Level Agreement
SN	Substrate Network
SP	Service Provider
SVNE	Survivable Virtual Network Embedding
UPC	Technical University of Catalonia
VN	Virtual Network
VNO	Virtual Network Operator
VNP	Virtual Network Provider
VNE	Virtual Network Embedding
VNS	Virtual Network Survivability

Appendix C

Link-based Optimal One-Shot VNE Formulation (ViNE-OPT)

This is the link based formulation of the one shot optimal virtual network embedding problem. We define f_{uv}^{ij} as the flow of a virtual link $l_{ij} \in L_v$ on the link $l_{uv} \in (L_s \cup L_x)$. L_x is the set of all meta links in the augmented substrate network.

$$\textbf{minimise} \sum_{l_{ij} \in L_v} \sum_{l_{uv} \in (L_s \cup L_x)} \frac{1}{A_{uv}} f_{uv}^{ij} \quad + \sum_{n_v \in N_v} \sum_{n_s \in N_s} \frac{1}{A_{n_s}} \chi_{n_s}^{n_v}$$

subject to

Node Mapping Constraints

$$\sum_{n_s \in N_s} \chi_{n_s}^{n_v} = 1 \qquad \forall n_v \in N_v$$

$$\sum_{n_v \in N_v} \chi_{n_s}^{n_v} \leq 1 \qquad \forall n_s \in N_s$$

$$f_{uv}^{ij} - D_{ij} \chi_u^i \leq 0 \qquad \forall uv \in L_x, \forall l_{ij} \in L_v$$

$$f_{uv}^{ij} - D_{ij} \chi_v^j \leq 0 \qquad \forall uv \in L_x, \forall l_{ij} \in L_v$$

Capacity Constraints

$$\sum_{ij \in L_v} f_{uv}^{ij} \leq A_{uv} \qquad \forall l_{uv} \in (L_s \cup L_x)$$

$$\sum_{uv \in L_v} f_{uv}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v$$

Flow Conservation Constraints

Source Nodes

$$\sum_{k \in N_s} f_{ik}^{ij} - \sum_{k \in N_s} f_{ki}^{ij} = D_{ij} \qquad \forall l_{ij} \in L_v$$

Sink Nodes

$$\sum_{k \in N_s} f_{jk}^{ij} - \sum_{k \in N_s} f_{kj}^{ij} = -D_{ij} \qquad \forall l_{ij} \in L_v$$

Intermediate Nodes

$$\sum_{u \in N_s} f_{uv}^{ij} - \sum_{u \in N_s} f_{uv}^{ij} = 0 \qquad \forall l_{ij} \in L_v, \forall v \in N_s$$

Domain Constraints

$$f_{uv}^{ij} = [0, D_{ij}] \qquad \forall l_{ij} \in L_v, \forall l_{uv} \in (L_s \cup L_x)$$

$$\chi_u^i = [0, 1] \qquad \forall i \in N_v, \forall u \in N_s$$

Bibliography

[1] Antonio Carzaniga and Alexander L. Wolf. Content-based networking: A new communication infrastructure. In Birgitta Knig-Ries, Kia Makki, S. A. M. Makki, Niki Pissinou, and Peter Scheuermann, editors, *Infrastructure for Mobile and Wireless Systems*, volume 2538 of *Lecture Notes in Computer Science*, pages 59–68. Springer, 2001.

[2] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, April 2010.

[3] Jonathan S. Turner and David E. Taylor. Diversifying the internet. In *GLOBECOM*, page 6. IEEE, 2005.

[4] Christophe Diot, Brian Neil, Levine Bryan, and Kassem Doug Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network*, 14:78–88, 2000.

[5] J.G. Jayanthi and S.A. Rabara. Ipv6 addressing architecture in ipv4 network. In *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pages 461–465, Feb 2010.

[6] Ashiq Khan, Alf Zugenmaier, Dan Jurca, and Wolfgang Kellerer. Network virtualization: a hypervisor for the internet? *IEEE Communications Magazine*, 50(1):136–143, 2012.

[7] Thomas E. Anderson, Larry L. Peterson, Scott Shenker, and Jonathan S. Turner. Overcoming the internet impasse through virtualization. *IEEE Computer*, 38(4):34–41, 2005.

[8] M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *Networking, IEEE/ACM Transactions on*, 20(1):206 –219, feb. 2012.

[9] Aun Haider, Richard Potter, and Akihiro Nakao. Challenges in resource allocation in network virtualization. In *Proceedings of 20th ITC Specialist Seminar*, Hoi An, Vietnam, 2009.

[10] L. Wenzhi, L. Shuai, X. Yang, and T. Xiongyan. Dynamically adaptive bandwidth allocation in network virtualization environment. *Advances in information Sciences and Service Sciences(AISS)*, 4(1):10 – 18, 2012.

[11] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.

[12] Steven Davy, Joan Serrat, Antonio Astorga, Brendan Jennings, and Javier Rubio-Loyola. Policy-assisted planning and deployment of virtual networks. In *CNSM*, pages 1–8. IEEE, 2011.

[13] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[14] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):pp. 316–329, 1998.

[15] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[16] J E Dayhoff and J M DeLeo. Artificial neural networks: opening the black box. *Cancer*, 91(8 Suppl):1615–1635, April 2001.

[17] A. Nrnberger, D. Nauck, and R. Kruse. Neuro-fuzzy control based on the nefcon-model: recent developments, 1999.

[18] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[19] David L. Sallach. Strategic negotiation in multiagent environments by sarit kraus. *J. Artificial Societies and Social Simulation*, 6(1), 2003.

[20] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, March 2008.

[21] Jing Lu and Jonathan Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical report, Washington University in St. Louis, 2006.

[22] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A distributed virtual network mapping algorithm. In *ICC*, pages 5634–5640. IEEE, 2008.

[23] Viktor K. Prasanna Sethavidh Gertphol. Iterative integer programming formulation for robust resource allocation in dynamic real-time systems. In *18th International Symposium on Parallel and Distributed Processing*, 2004.

[24] Renchao Xie, F. Richard Yu, and Hong Ji. Dynamic resource allocation for heterogeneous services in cognitive radio networks with imperfect channel sensing. *IEEE T. Vehicular Technology*, 61(2):770–780, 2012.

[25] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer. Energy efficient virtual network embedding. *Communications Letters, IEEE*, 16(5):756–759, May 2012.

[26] Muntasir Raihan Rahman and Raouf Boutaba. Svne: Survivable virtual network embedding algorithms for network virtualization. *Network and Service Management, IEEE Transactions on*, 10(2):105–118, 2013.

[27] Cynthia Barnhart, Niranjan Krishnan, and PamelaH. Vance. Multicommodity flow problems. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2354–2362. Springer US, 2009.

[28] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[29] Jacques Desrosiers and Marco E. Lbbecke. A primer in column generation, 2005.

[30] D. Santos, A. de Sousa, F. Alvelos, and M. Pioro. Link load balancing optimization of telecommunication networks: A column generation based heuristic approach. In *Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2010 14th International*, pages 1–6, Sept 2010.

[31] Abdallah Jarray and Ahmed Karmouch. Column generation approach for one-shot virtual network embedding. In *GLOBECOM Workshops*, pages 863–868, 2012.

[32] Wenping Pan, Dejun Mu, Hangxing Wu, and Lei Yao. Feedback control-based qos guarantees in web application servers. In *HPCC*, pages 328–334. IEEE, 2008.

[33] Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. A multi-model framework to implement self-managing control systems for qos management. In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '11, pages 218–227, New York, NY, USA, 2011. ACM.

[34] Rui Han, Li Guo, M.M. Ghanem, and Yike Guo. Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644–651, May 2012.

[35] Wei-Sheng Lai, Muh-En Chiang, Shen-Chung Lee, and Ta-Sung Lee. Game theoretic distributed dynamic resource allocation with interference avoidance in cognitive femtocell networks. In *WCNC*, pages 3364–3369. IEEE, 2013.

[36] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu. Resource provisioning for cloud computing. In *Conference of the Center for Advanced Studies on Collaborative Research*, pages 101 –111, 2009.

[37] Fareed Jokhio, Adnan Ashraf, Sebastien Lafond, Ivan Porres, and Johan Lilius. Prediction-based dynamic resource allocation for video transcoding in cloud computing. In *Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, PDP '13, pages 254–261, Washington, DC, USA, 2013. IEEE Computer Society.

[38] Jiayue He, Rui Zhang-shen, Ying Li, Cheng yen Lee, Jennifer Rexford, and Mung Chiang. Davinci: Dynamically adaptive virtual networks for a customized internet. In *in Proc. CoNEXT*, 2008.

[39] C.C. Marquezan, L.Z. Granville, G. Nunzi, and M. Brunner. Distributed autonomic resource management for network virtualization. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 463–470, April 2010.

[40] Zhiping Cai, Fang Liu, Nong Xiao, Qiang Liu, and Zhiying Wang. Virtual network embedding for evolving networks. In *GLOBECOM*, pages 1–5. IEEE, 2010.

[41] Ilhem Fajjari, Nadjib Aitsaadi, Guy Pujolle, and Hubert Zimmermann. Vnr algorithm: A greedy approach for virtual networks reconfigurations. In *GLOBECOM*, pages 1–6. IEEE, 2011.

[42] Gang Sun, Hongfang Yu, Vishal Anand, and Lemin Li. A cost efficient framework and algorithm for embedding dynamic virtual network requests. *Future Gener. Comput. Syst.*, 29(5):1265–1277, July 2013.

[43] M.F. Zhani, Qi Zhang, G. Simon, and R. Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 18–25, May 2013.

[44] Nabeel Farooq Butt, Mosharaf Chowdhury, and Raouf Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. In Mark Crovella, LauraMarie Feeney, Dan Rubenstein, and S.V. Raghavan, editors, *NETWORKING 2010*, volume 6091 of *Lecture Notes in Computer Science*, pages 27–39. Springer Berlin Heidelberg, 2010.

[45] Gang Sun, Hongfang Yu, Lemin Li, Vishal Anand, Yanyang Cai, and Hao Di. Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter. *Photonic Network Communications*, 23(2):109–122, 2012.

[46] Sheng Zhang, Zhuzhong Qian, Bin Tang, Jie Wu, and Sanglu Lu. Opportunistic bandwidth sharing for virtual network mapping. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5, Dec 2011.

[47] Junhong Nie and S. Haykin. A Q-learning-based dynamic channel assignment technique for mobile communication systems. *Vehicular Technology, IEEE Transactions on*, 48(5):1676–1687, 1999.

[48] Andrei Lucian Stefan, Mandalika Ramkumar, Rasmus H. Nielsen, Neeli R. Prasad, and Ramjee Prasad. A qos aware reinforcement learning algorithm for macro-femto interference in dynamic environments. In *ICUMT*, pages 1–7. IEEE, 2011.

[49] F. Bernardo, R. Agusti, J. Perez-Romero, and O. Sallent. Distributed spectrum management based on reinforcement learning. In *Cognitive Radio Oriented Wireless Networks and Communications, 2009. CROWNCOM '09. 4th International Conference on*, pages 1–6, June 2009.

[50] Tao Jiang, David Grace, and Yiming Liu. Two-stage reinforcement-learning-based cognitive radio with exploration control. *IET Communications*, 5(5):644–651, 2011.

[51] Dorian Minarolli and Bernd Freisleben. Utility-driven allocation of multiple types of resources to virtual machines in clouds. In *Proceedings of the 2011 IEEE 13th Conference on Commerce and Enterprise Computing*, CEC '11, pages 137–144, Washington, DC, USA, 2011. IEEE Computer Society.

[52] El-Sayed M. El-Alfy, Yu-Dong Yao, and Harry Heffes. Autonomous call admission control with prioritized handoff in cellular networks. In *ICC*, pages 1386–1390. IEEE, 2001.

[53] Yih-Shen Chen, Chung-Ju Chang, and Fang-Ching Ren. Q-learning-based multirate transmission control scheme for rrm in multimedia wcdma systems. *IEEE T. Vehicular Technology*, 53(1):38–48, 2004.

[54] S. M. Perlaza, S. Lasaulce, H. Tembine, and M. Debbah. Learning to use the spectrum in self-configuring heterogenous networks: A logit equilibrium approach. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11, pages 565–571, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[55] Afef Feki and Vronique Capdevielle. Autonomous resource allocation for dense lte networks: A multi armed bandit formulation. In Kaveh Pahlavan, Shahrokh Valaee, and Elvino Silveira Sousa, editors, *PIMRC*, pages 66–70. IEEE, 2011.

[56] Steven Walczak. Neural network models for a resource allocation problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(2):276–284, 1998.

[57] Stephen C. Stubberud and Kathleen A. Kramer. A game theoretic sensor resource allocation using fuzzy logic. *Adv. Fuzzy Systems*, 2013, 2013.

[58] Lorenza Giupponi, Ramn Agust, Jordi Prez-Romero, and Oriol Sallent. A novel approach for joint radio resource management based on fuzzy neural methodology. *IEEE T. Vehicular Technology*, 57(3):1789–1805, 2008.

[59] Hongfang Yu, Vishal Anand, Chunming Qiao, and Hao Di. Migration based protection for virtual infrastructure survivability for link failure. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2011*, page OTuR2. Optical Society of America, 2011.

[60] Ines Houidi, Wajdi Louati, Djamal Zeghlache, Panagiotis Papadimitriou, and Laurent Mathy. Adaptive virtual network provisioning. In *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '10, pages 41–48, New York, NY, USA, 2010. ACM.

[61] A. Khan S. Herker and X. An. Survey on survivable virtual network embedding problem and solutions. In *International Conference on Networking and Services*, ICNS 2013, March 2013.

[62] Z. Despotovic, J.-C. Usunier, and K. Aberer. Towards peer-to-peer double auctioning. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 8 pp.–, Jan 2004.

[63] Z. Despotovic, J.-C. Usunier, and K. Aberer. Towards peer-to-peer double auctioning. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 8 pp.–, Jan 2004.

[64] J. Rubio-Loyola, C. Merida-Campos, S. Willmott, A. Astorga, J. Serrat, and A. Galis. Service coalitions for future internet services. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, June 2009.

[65] J. Rubio-Loyola, C. Merida-Campos, J. Serrat, D.F. Macedo, S. Davy, Z. Movahedi, and G. Pujolle. A service-centric orchestration protocol for self-organizing autonomic management systems. *Network, IEEE*, 25(6):16–23, Nov 2011.

[66] Carrie Beam and Arie Segev. Automated negotiations: A survey of the state of the art. *Wirtschaftsinformatik*, 39(3):263–268, 1997.

[67] Oracle. Oracle solaris administration: Network interfaces and network virtualization, 2011.

[68] Brad Hedlund. What is network virtualization? `http://bradhedlund.com/2013/05/28/what-is-network-virtualization/`, March 2013. Accessed: 2014-04-26.

[69] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer. Network virtualization: a hypervisor for the internet? *Communications Magazine, IEEE*, 50(1):136–143, January 2012.

[70] Kurt Tutschku, Thomas Zinner, Akihiro Nakao, and Phuoc Tran-Gia. Network virtualization: Implementation steps towards the future internet. In *KiVS 2009*, Kassel, 3 2009.

[71] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: state of the art and research challenges. *Comm. Mag.*, 47:20–26, July 2009.

[72] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41, April 2005.

[73] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.*, 37(1):61–64, January 2007.

[74] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '09, pages 63–72, New York, NY, USA, 2009. ACM.

[75] Fady Samuel, Mosharaf Chowdhury, and Raouf Boutaba. Polyvine: policy-based virtual network embedding across multiple domains. *Journal of Internet Services and Applications*, 4(1):6, 2013.

[76] Andreas Berl, Andreas Fischer, and Hermann de Meer. Using system virtualization to create virtualized networks. *ECEASST*, 17, 2009.

[77] Roland Bless and Christoph Werle. Control plane issues in the 4ward network virtualization architecture. *ECEASST*, 17, 2009.

[78] David G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, December 2002.

[79] Khaled Elbassioni, Naveen Garg, Divya Gupta, Amit Kumar, Vishal Narula, and Arindam Pal. Approximation Algorithms for the Unsplittable Flow Problem on Paths and Trees. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 267–275, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[80] Stavros G. Kolliopoulos and Clifford Stein. Improved approximation algorithms for unsplittable flow problems. In *FOCS*, pages 426–435. IEEE Computer Society, 1997.

[81] Ines Houidi, Wajdi Louati, Walid Ben Ameur, and Djamal Zeghlache. Virtual network provisioning across multiple substrate networks. *Comput. Netw.*, 55(4):1011–1023, mar 2011.

[82] Hao Di, Hongfang Yu, Vishal Anand, Lemin Li, Gang Sun, and Binhong Dong. Efficient online virtual network mapping using resource evaluation. *J. Netw. Syst. Manage.*, 20(4):468–488, December 2012.

[83] Yufeng Xin, Ilia Baldine, Anirban Mandal, Chris Heermann, Jeff Chase, and Aydan Yumerefendi. Embedding virtual topologies in networked clouds. In

Proceedings of the 6th International Conference on Future Internet Technologies, CFI '11, pages 26–29, New York, NY, USA, 2011. ACM.

[84] Gregor Schaffrath, Stefan Schmid, and Anja Feldmann. Generalized and resource-efficient vnet embeddings with migrations. *CoRR*, abs/1012.4066, 2010.

[85] A. Razzaq, P. Sjodin, and M. Hidell. Minimizing bottleneck nodes of a substrate in virtual network embedding. In *Network of the Future (NOF), 2011 International Conference on the*, pages 35–40, Nov 2011.

[86] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. Virtual network mapping into heterogeneous substrate networks. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 438–444, June 2011.

[87] A. Razzaq and M.S. Rathore. An approach towards resource efficient virtual network embedding. In *Evolving Internet (INTERNET), 2010 Second International Conference on*, pages 68–73, Sept 2010.

[88] Xiujiao Gao, Hongfang Yu, Vishal Anand, Gang Sun, and Hao Di. A new algorithm with coordinated node and link mapping for virtual network embedding based on lp relaxation. In *Communications and Photonics Conference and Exhibition (ACP), 2010 Asia*, pages 152–153, Dec 2010.

[89] A. Pages, J. Perello, S. Spadaro, and G. Junyent. Strategies for virtual optical network allocation. *Communications Letters, IEEE*, 16(2):268–271, February 2012.

[90] Wenzhi Liu, Yang Xiang, Shaowu Ma, and Xiongyan Tang. Completing virtual network embedding all in one mathematical programming. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 183–185, Sept 2011.

[91] T. Ghazar and N. Samaan. Hierarchical approach for efficient virtual network embedding based on exact subgraph matching. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, Dec 2011.

[92] H. Esaki T. Trinh and C. Aswakul. Quality of service using careful overbooking for optimal virtual network resource allocation. In *International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, volume 8, pages 296–299, 2011.

[93] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.N. Chuah, and C. Diot. Characterization of failures in an IP backbone. In *INFOCOM 2004*, 2004.

[94] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee. Survivable wdm mesh networks. *Lightwave Technology, Journal of*, 21(4):870–883, April 2003.

[95] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, August 2011.

[96] A. Jarray and A. Karmouch. Vcg auction-based approach for efficient virtual network embedding. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 609–615, May 2013.

[97] Sen Su, Zhongbao Zhang, Xiang Cheng, Yiwen Wang, Yan Luo, and Jie Wang. Energy-aware virtual network embedding through consolidation. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 127–132, March 2012.

[98] S. Bradley, A. Hax, and T. Magnanti. *Applied Mathematical Programming*. Addison Wesley, 1977.

[99] Olivier Chapelle, Vikas Sindhwani, and Sathiya S. Keerthi. Optimization techniques for semi-supervised support vector machines. *J. Mach. Learn. Res.*, 9:203–233, June 2008.

[100] John W. Chinneck. Practical Optimization: a Gentle Introduction. `www.sce.carleton.ca/faculty/chinneck/po.html`, 2001. Accessed: 2014-04-30.

[101] S. Lahaie. How to take the Dual of a Linear Program. www.cs.columbia.edu/coms6998-3/lpprimer.pdf?, 2008. Accessed: 2014-02-17.

[102] Wikipedia: Column Generation. http://en.wikipedia.org/wiki/Column_generation. Accessed: 2014-04-30.

[103] Michel X. Goemans and David P. Williamson. Approximation algorithms for np-hard problems. chapter The Primal-dual Method for Approximation Algorithms and Its Application to Network Design Problems, pages 144–191. PWS Publishing Co., Boston, MA, USA, 1997.

[104] Dimitri P. Bertsekas and Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, September 1999.

[105] L. R. Ford, Jr. and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Manage. Sci.*, 50(12 Supplement):1778–1780, December 2004.

[106] Jacques Desrosiers and MarcoE. Labbecke. A primer in column generation. In Guy Desaulniers, Jacques Desrosiers, and MariusM. Solomon, editors, *Column Generation*, pages 1–32. Springer US, 2005.

[107] S. Irnich and G. Desaulniers. *Shortest Path Problems with Resource Constraints*, chapter 2, pages 33–65. GERAD 25th Anniversary Series. Springer, 2005.

[108] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[109] M Güzelsoy and T K Ralphs. Integer programming duality. In J. Cochran, editor, *Encyclopedia of Operations Research and Management Science*. Wiley, 2010.

[110] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite: An approach to universal topology generation. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication*

Systems, MASCOTS '01, pages 346–353, Washington, DC, USA, 2001. IEEE Computer Society.

[111] IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/about/`. Accessed: 2014-02-17.

[112] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006.

[113] Gerhard J. Woeginger. Combinatorial optimization - eureka, you shrink! chapter Exact Algorithms for NP-hard Problems: A Survey, pages 185–207. Springer-Verlag New York, Inc., New York, NY, USA, 2003.

[114] Jeff Erickson. Lecture Notes: Efficient Exponential-Time Algorithms. `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/`, 2010. Accessed: 2013-02-01.

[115] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[116] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[117] Andrea Schaerf, Yoav Shoham, and Moshe Tennenholtz. Adaptive load balancing: A study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2:475–500, 1995.

[118] Aram Galstyan, Shashikiran Kolar, and Kristina Lerman. Resource allocation games with changing resource capacities. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 145–152, New York, NY, USA, 2003. ACM.

[119] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource allocation in the grid with learning agents. *J. Grid Comput.*, 3(1-2):91–100, 2005.

[120] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[121] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010.

[122] Csaba Szepesvári. *Algorithms for Reinforcement Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[123] Gergely Neu, András György, Csaba Szepesvári, and András Antos. Online markov decision processes under bandit feedback. In *NIPS*, pages 1804–1812, 2010.

[124] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics).* Wiley-Interscience, 2007.

[125] *Monte Carlo Statistical Methods.* Springer-Verlag, 1 edition, August 1999.

[126] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time dependent problems and difference methods.* Pure and applied mathematics. J. Wiley, New York, Chichester, Brisbane, 1995.

[127] Ana Marıa Galindo Serrano. Self-organized femtocells: a time difference learning approach. 2012.

[128] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[129] Michel Tokic. Adaptive e-greedy exploration in reinforcement learning based on value differences. In *Proceedings of the 33rd Annual German Conference on Advances in Artificial Intelligence*, KI'10, pages 203–210, Berlin, Heidelberg, 2010. Springer-Verlag.

[130] L. P. Pitaevskii and E. M. Lifshitz. *Statistical Physics, Course of Theoretical Physics 5, 3rd Edition.* Oxford: Pergamon Press, January 1976.

[131] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Trans. Sys. Man Cyber Part C*, 38(2):156–172, March 2008.

[132] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A multi-agent systems approach to autonomic computing. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 464–471, Washington, DC, USA, 2004. IEEE Computer Society.

[133] Network Simulator 3. `http://www.nsnam.org/`. Accessed: 2014-02-17.

[134] The CAIDA Anonymized Internet Traces 2012 - 20 December 2012, equinix sanjose.dirB.20121220-140100.UTC.anon.pcap.gz. `http://www.caida.org/data/passive/passive_2012_dataset.xml`. Accessed: 2014-02-17.

[135] F. L. Lewis, S.Q. Zhu, and K. Liu. Function approximation by fuzzy systems. In *American Control Conference, Proceedings of the 1995*, volume 5, pages 3760–3764 vol.5, Jun 1995.

[136] Fu Qi-ming, Liu Quan, Cui Zhi-ming, and Fu Yu-chen. A reinforcement learning algorithm based on minimum state method and average reward. *Computer Science and Information Engineering, World Congress on*, 5:534–538, 2009.

[137] Jeff Heaton. *Introduction to Neural Networks for Java, 2Nd Edition.* Heaton Research, Inc., 2nd edition, 2008.

[138] J.J. Hopfield. Artificial neural networks. *Circuits and Devices Magazine, IEEE*, 4(5):3–10, Sept 1988.

[139] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *Information Theory, IEEE Transactions on*, 39(3):930–945, May 1993.

[140] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[141] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA, 1999.

[142] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[143] Jing Li, Ji-hang Cheng, Jing-yuan Shi, and Fei Huang. Brief introduction of back propagation (bp) neural network algorithm and its improvement. In David Jin and Sally Lin, editors, *Advances in Computer Science and Information Engineering*, volume 169 of *Advances in Intelligent and Soft Computing*, pages 553–558. Springer Berlin Heidelberg, 2012.

[144] Wikipedia: Backpropagation. `http://en.wikipedia.org/wiki/Backpropagation`. Accessed: 2014-05-09.

[145] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[146] Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Maxim Claeys, Filip De Turck, and Steven Latre. Design and evaluation of learning algorithms for dynamic resource management in virtual networks. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, NOMS2014. IEEE, 2014.

[147] Stephen R. Garner. Weka: The waikato environment for knowledge analysis. In *In Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[148] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995.

[149] Junfei Qiao, Ruiyuan Fan, Honggui Han, and Xiaogang Ruan. Q-learning based on dynamical structure neural network for robot navigation in unknown envi-

ronment. In *ISNN (3)*, volume 5553 of *Lecture Notes in Computer Science*, pages 188–196. Springer, 2009.

[150] Shi chao Wang, Zheng xi Song, Hao Ding, and Hao bin Shi. An improved reinforcement q-learning method with bp neural networks in robot soccer. In *ISCID (1)*, pages 177–180. IEEE, 2011.

[151] Nikola K. Kasabov. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering.* MIT Press, Cambridge, MA, USA, 1st edition, 1996.

[152] Chia-Feng Juang and Chin-Teng Lin. An online self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems*, 6(1):12–32, 1998.

[153] Chin-Teng Lin. A neural fuzzy control system with structure and parameter learning. *Fuzzy Sets and Systems*, 70(23):183 – 212, 1995. Modern Fuzzy Control.

[154] Vaclav Dvorak. Neural networks and fuzzy systems : B kosko prentice-hall. *Knowl.-Based Syst.*, 6(3):179, 1993.

[155] Mohamad H. Hassoun. Neural fuzzy systems: A neuro-fuzzy synergism to intelligent systems [book review]. *IEEE Transactions on Neural Networks*, 7(5):1316, 1996.

[156] José Vieira, Fernando Morgado Dias, and Alexandre Mota. Neuro-fuzzy systems: A survey. In *5th WSEAS NNA International Conference on Neural Networks and Applications*, Udine, Italy, 2004.

[157] Detlef Nauck and Rudolf Kruse. A neural fuzzy controller learning by fuzzy error propagation. In *In Proc. NAFIPS'92*, pages 388–397, 1992.

[158] Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Ke Xu, Meng Shen, and Kun Yang. A neuro-fuzzy approach to self-management of virtual network resources. *Journal of Expert Systems With Applications*, Feb 2014. Submitted to.

[159] Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, Maxim Claeys, Filip De Turck, and Jeroen Famaey. Neural network-based autonomous allocation of resources in virtual networks. In *Proceedings of the European Conference on Networks and Communications (EuCNC)*, EuCNC2014, June 2014.

[160] Gianluca Iannaccone, Chen-nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot. Analysis of link failures in an ip backbone. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 237–242, New York, NY, USA, 2002. ACM.

[161] A. Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational ip backbone network. *Networking, IEEE/ACM Transactions on*, 16(4):749–762, Aug 2008.

[162] Huhnkuk Lim and Youngho Lee. Toward reliability guarantee vc services in an advance reservation based network resource provisioning system. In *In Proc. of the Eighth International Conference on Systems and Networks Communications*, ICSNC2013, pages 112 – 120, November 2013.

[163] Nicholas R. Jennings, Peyman Faratin, A. R. Lomuscio, Simon Parsons, Michael Wooldridge, and Carles Sierra. Automated negotiation : prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[164] Martin Beer, Mark D'inverno, Michael Luck, Nick Jennings, Chris Preist, and Michael Schroeder. Negotiation in multi-agent systems. *Knowl. Eng. Rev.*, 14(3):285–289, September 1999.

[165] Mikhail J. Atallah and Susan Fox, editors. *Algorithms and Theory of Computation Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1998.

[166] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.

[167] Yi hua Zhu, Xian-Zhong Tian, and Jun Zheng. Performance analysis of the binary exponential backoff algorithm for ieee 802.11 based mobile ad hoc networks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, June 2011.

[168] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[169] FIPA ACL Message Structure Specification. `http://www.fipa.org/specs/fipa00061/`. Accessed: 2014-04-02.

[170] Robert B Abernethy. *The New Weibull handbook: reliability and statistical analysis for predicting life, safety, supportability, risk, cost and warranty claims; 5th ed.* Dr. Robert B. Abernethy, 2006.

[171] S. Patidar, D. Rane, and P. Jain. A survey paper on cloud computing. In *Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on*, pages 394–398, Jan 2012.

[172] Han Xingye, Li Xinming, and Liu Yinpeng. Research on resource management for cloud computing based information system. In *Computational and Information Sciences (ICCIS), 2010 International Conference on*, pages 491–494, Dec 2010.

[173] Flamingo, Network of Excellence. `http://www.fp7-flamingo.eu/`. Accessed: 2014-05-20.

[174] iMinds, Virtual Wall. `http://www.iminds.be/en/succeed-with-digital-research/technical-testing`. Accessed: 2014-05-20.