ASSURING VIRTUAL NETWORK RELIABILITY AND RESILIENCE

By

Baker Alruabaiey

This thesis submitted in total fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Information Technology
Faculty of Science, Engineering and Built Environment

Deakin University
June 2016
I am the author of the thesis entitled “ASSURING VIRTUAL NETWORK RELIABILITY AND RESILIENCE”. Submitted for the degree of Doctor of Philosophy

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .................... Baker Alrubaiey ............................

Signed: .......................... ...........................

Date: ..............................   30/06/2016  ...................................
I certify the following about the thesis entitled “ASSURING VIRTUAL NETWORK RELIABILITY AND RESILIENCE”. Submitted for the degree of Doctor of Philosophy.

a. I am the creator of all or part of the whole work(s) (including content and layout) and that where reference is made to the work of others, due acknowledgment is given.

b. The work(s) are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person.

c. That if the work(s) have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

I also certify that any material in the thesis, which has been accepted, for a degree or diploma by any university or institution is identified in the text.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name: .................... Baker Alrubaiey............................

Signed: ..........................                                  ..........................

Dates: ............................. 30/06/2016  .............................
Acknowledgments

I would like to express my sincere gratitude to my supervisor Professor Jemal Abawajy for his patience, motivation, enormous knowledge and continuous support throughout my PhD studies. His assistance helped me in all aspects of the research and writing of this thesis.

I would like to thank my family for their continued spiritual support of me throughout my PhD studies, writing this thesis and my life in general. In particular, my heartiest thanks to my wife Hadeel and my daughters Russel and Youser for their support and patience while I was busy with my studies.

Last, but not the least, I thank my God (ALLAH) for allowing me to finish my degree and helping me with the difficulties throughout my PhD studies.
Publications

The following papers that I either authored or co-authored have been published or are currently under consideration for publication. These papers are reprinted in this dissertation with the full permission of all co-authors.

Journals


Conference Papers


Abstract

Network virtualisation is an enabling technology that will allow the future Internet to overcome the obstacles of the current Internet to architecture change. The future Internet architecture will be separated into virtual networks that can concurrently run network services and architectures over a shared substrate network. Although virtual networks offer enormous advantages in terms of cost and accessibility, virtual networks are vulnerable to failure due to different factors. Therefore, reliability in a virtual network environment (VNE) is an important issue that needs to be addressed before a virtual network can be used. The aim of this thesis was to improve the virtual network reliability by designing a reliable VNE that can operate normally, even in the event of link or node substrate failure. A framework developed that uses reliability block diagrams and continuous-time Markov chains to model and analyse the reliability and availability of a VNE. The framework can be used for the design and construction of more reliable VNE. In addition, to minimise the unpredicted failures and reduce the impact of failure on a virtual network, a dynamic solution proposed for detecting a failure before it occurs in the VNE. The detection mechanism is based on a conservative time-synchronisation algorithm with a message passing interface. Moreover, to predict failure and establish a tolerable maintenance plan before failure occurs in the VNE, a failure prediction method developed based on time series and support vector regression models. The proposed prediction mechanism for VNE can be used to minimise the unpredicted failures, reduce backup redundancy and maximise system performance. The results show that the framework can use reliability as a level of a service required by the client to allocate resources for virtual networks according to the quality of service. A framework for evaluating reliability and availability achieved high performance compared with previous work. In addition, the failure detection mechanism showed a very small number of messages exchanged in event of failure. Our approach achieved
a high performance compared with previous work in the detection of failure in VNE. Finally, the failure prediction method achieved very high accuracy in prediction the future failures in VNE because the predicted results were very close to the observed values.
# Table of Contents

Acknowledgments ........................................................................................................................................... i
Publications .................................................................................................................................................. ii
Abstract ..................................................................................................................................................... iv
Table of Contents ........................................................................................................................................ vi
List of Figures .............................................................................................................................................. viii
List of Tables ................................................................................................................................................ ix
Abbreviations .............................................................................................................................................. x

Chapter 1: Introduction ................................................................................................................................. 1
  1.1. Background ........................................................................................................................................ 1
  1.2. Thesis Aims ......................................................................................................................................... 2
  1.3. Motivation .......................................................................................................................................... 3
  1.4. Research Problems and Major Contributions ......................................................................................... 6
    1.4.1. Handling Substrate Link Failures ................................................................................................. 7
    1.4.2. Handling Substrate Node Failures ............................................................................................... 7
    1.4.3. Handling Correlated Substrate Link and Node Failures ............................................................... 8
  1.5. Significance of Contributions ................................................................................................................ 8
  1.6. Research Methodology ........................................................................................................................ 9
  1.7. Thesis Organisation ............................................................................................................................ 12

Chapter 2: Literature Review ......................................................................................................................... 13
  2.1. Introduction ........................................................................................................................................ 13
  2.2. Conceptual Virtual Network Architecture ........................................................................................... 16
    2.2.1. Internet In a Slice Architecture .................................................................................................. 16
    2.2.2. CABO Architecture of Future Internet ...................................................................................... 17
    2.2.3. AGAVE ....................................................................................................................................... 18
    2.2.4. FEDERICA ................................................................................................................................. 18
  2.3. Review of Reliability of Virtual Network Due to Substrate Link Failure .............................................. 19
  2.4. Review of Reliability of Virtual Network Due to Substrate Node Failure ........................................... 25
  2.5. Review of Reliability of Virtual Network Due to Substrate Link and Node Failures ......................... 29

Chapter 3: Virtual Network Dependability Assessment Framework .................................................................... 34
  3.1. Introduction ........................................................................................................................................ 34
  3.2. Models ................................................................................................................................................. 37
    3.2.1. System Model ............................................................................................................................. 37
    3.2.2. Problem overview ..................................................................................................................... 38
  3.3. Dependability Assessment Framework .................................................................................................. 41
  3.4. Reliability Block Diagram Representation of Substrate Network ......................................................... 43
  3.5. Continuous-Time Markov Chain Representation of Dynamic Substrate Network ......................... 45
    3.5.1. Simple Mapping ......................................................................................................................... 46
    3.5.2. Passive Mapping ......................................................................................................................... 47
    3.5.3. Active Mapping .......................................................................................................................... 49
  3.6. Performance Analysis ........................................................................................................................... 51
Abstract

3.6.1. Experimental Set-up........................................................................................................... 51
3.6.2. Results and Discussion ..................................................................................................... 51
3.7. Chapter Summary ................................................................................................................. 57

Chapter 4: Failure Detection in Virtual Network Infrastructure .................................................. 59
4.1. Introduction .......................................................................................................................... 59
4.2. Problem Overview ............................................................................................................... 61
4.3. System Model ...................................................................................................................... 64
  4.3.1. VNE Topology .................................................................................................................. 65
  4.3.2. Fault Detection Model ....................................................................................................... 67
  4.3.3. Data Collection Model ...................................................................................................... 69
  4.3.4. Metrics Used ................................................................................................................... 70
4.4. Performance Analysis ........................................................................................................... 71
  4.4.1. Experimental Set-up ......................................................................................................... 71
  4.4.2. Results and Discussion .................................................................................................... 71
  4.4.2.1. Accuracy ...................................................................................................................... 72
  4.4.2.2. Average Failure Detection Time .................................................................................... 75
  4.4.2.3. Average Number of Messages Exchanged ................................................................. 76
  4.4.3. SVM Model Detection Results ....................................................................................... 78
4.5. Chapter Summary ................................................................................................................ 81

Chapter 5: Prediction of Virtual Network Substrate Failures ....................................................... 83
5.1. Introduction .......................................................................................................................... 83
5.2. Problem Overview ............................................................................................................... 84
5.3. Support Vector Regression .................................................................................................. 87
5.4. Predicting Failure in VNE ................................................................................................... 90
5.5. Performance Analysis ........................................................................................................... 94
  5.5.1. Experimental Set-up ......................................................................................................... 94
  5.5.2. Data Sets ......................................................................................................................... 96
  5.5.3. Results and Discussion .................................................................................................... 97
  5.5.3.1. Prediction Failure in Virtual Networks .......................................................................... 97
  5.5.3.2. Prediction Failure in Physical Nodes ............................................................................ 99
  5.5.3.3. Prediction Failure in Physical Link .............................................................................. 100
  5.5.4. Validation ....................................................................................................................... 102
  5.5.5. Failure Prediction Performance ....................................................................................... 103
5.6. Chapter Summary ................................................................................................................ 105

Chapter 6: Conclusions and Future Directions ............................................................................ 106
6.1. Accomplishments ............................................................................................................... 106
6.2. Directions for Future Work ................................................................................................ 109

References .................................................................................................................................... 112
List of Figures

Figure 1-1 Virtual Network Embedding Model ................................................................. 5
Figure 2-1 Network Virtualisation Framework ................................................................. 15
Figure 3-1 Framework for Dependability Metrics Evaluation ......................................... 42
Figure 3-2 Single Mapping ......................................................................................... 43
Figure 3-3 Passive Mapping ....................................................................................... 44
Figure 3-4 Active Mapping ....................................................................................... 44
Figure 3-5 Single Mapping Model ............................................................................... 46
Figure 3-6 Passive Mapping Model ............................................................................ 49
Figure 3-7 Active Mapping Model ............................................................................. 50
Figure 3-8 Reliability Metric for Virtual Network Allocation ....................................... 54
Figure 3-9 Availability Results for Virtual Network Allocation .................................... 55
Figure 3-10 Availability Results ............................................................................... 56
Figure 4-1 Virtual Network Requests ......................................................................... 62
Figure 4-2 Virtual Network Maps onto Substrate Network ......................................... 63
Figure 4-3 VNE Hierarchical Topology .................................................................... 66
Figure 4-4 Partitioned Network Topology with Two LPs ............................................. 67
Figure 4-5 Time-stamp Message Sequence Transmission .......................................... 69
Figure 4-6 Failures Detected with a Failure Rate of \( \lambda = 0.001 \) ................................. 73
Figure 4-7 Failures Detected with a Failure Rate of \( \lambda = 0.003 \) ................................. Error! Bookmark not defined.
Figure 4-8 Accuracy with Different Look-ahead Values ............................................. 74
Figure 4-9 Average Failure Detection Time Using Different Numbers of Clusters ...... 75
Figure 4-10 Number of Messages Exchanged with Different Numbers of Clusters ..... 76
Figure 4-11 Number of Messages Exchanged with Different Number of Nodes ......... 77
Figure 4-12 Receiver Operating Characteristic Curve Results for SVM and Naive Bayesian Models ...... 80
Figure 4-13 Receiver Operating Characteristic Curve Results for SVM and Decision Tree Models..... 81
Figure 5-1 Epsilon Intensive Band – Loss Function ...................................................... 90
Figure 5-2 Architecture of Failure Prediction Model in VNE ........................................ 91
Figure 5-3 Virtual Network Topology ....................................................................... 95
Figure 5-4 Prediction of One Step Ahead TTF of the Virtual Network ....................... 98
Figure 5-5 Prediction of Two Steps Ahead TTF of the Virtual Network ....................... 98
Figure 5-6 Prediction of One Step Ahead TTF the Physical Nodes ............................. 99
Figure 5-7 Prediction of Two Step Ahead TTF of the Physical Nodes ......................... 100
Figure 5-8 Prediction of One Step Ahead TTF of the Physical Links .......................... 101
Figure 5-9 Prediction of Two Step Ahead TTF of the Physical Links .......................... 101
List of Tables

Table 2-1. Differences between VINI, CABO, AGAVE and FEDERICA .................................................... 19
Table 3-1 Component MTTF and MTTR ................................................................................................. 52
Table 3-2 Availability Measurements for Virtual Network Mapping .................................................... 57
Table 4-1 True Positive Rate and False Negative Rate .......................................................................... 74
Table 4-2 Success Rate Percentage in SVR Model .................................................................................. 79
Table 5-1 Training Pattern in SVR Model .............................................................................................. 93
Table 5-2 TTF for Virtual Infrastructure Components ........................................................................... 96
Table 5-3 Training Parameters for SVR ................................................................................................. 97
Table 5-4 RMSE for SVR Model of Virtual Network Component ........................................................ 102
Table 5-5 NRMSE for Virtual Network SVR, MLP and Gaussian Process Models ............................... 103
Table 5-6 NRMSE for Physical Node SVR, MLP and Gaussian Process Models ................................. 104
Table 5-7 NRMSE for Physical Link SVR, MLP and Gaussian Process Models ................................. 104
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGAVE</td>
<td>A liGhtweight Approach for Viable End-to-end IP-based QoS Services</td>
</tr>
<tr>
<td>CABO</td>
<td>Concurrent Architectures are Better than One</td>
</tr>
<tr>
<td>FEDERICA</td>
<td>Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures</td>
</tr>
<tr>
<td>VNM</td>
<td>Virtual Network Mapping</td>
</tr>
<tr>
<td>MTTF</td>
<td>Mean Time To Failure</td>
</tr>
<tr>
<td>MTTR</td>
<td>Mean Time To Repair</td>
</tr>
<tr>
<td>NRMSE</td>
<td>Normalised Root Mean Square Error</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>VINI</td>
<td>Virtual Network Infrastructure</td>
</tr>
<tr>
<td>SVNE</td>
<td>Survival Virtual Network Embedding</td>
</tr>
<tr>
<td>VNE</td>
<td>Virtual Network Environment</td>
</tr>
<tr>
<td>VN</td>
<td>Virtual Network</td>
</tr>
<tr>
<td>SN</td>
<td>Substrate Network</td>
</tr>
<tr>
<td>TTF</td>
<td>Time To Failure</td>
</tr>
<tr>
<td>BRITE</td>
<td>Boston University Representative Internet Topology gEnerator</td>
</tr>
<tr>
<td>NS-3</td>
<td>Network Simulator 3</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

1.1. Background

Internet architecture does not easily accommodate fundamental changes. Network virtualisation has been recognised as an enabling technology for the future Internet [1] and virtual network technology is rapidly evolving. Network virtualisation enables multiple virtual networks to run on a single shared substrate network. Network virtualisation allows users to create an individual virtual network with particular application naming, topology, routing table and resources management mechanisms such as server virtualisation. Network virtualisation also enables users to remotely access computing resources such as their own personal computers. Each virtual network is instantiated and managed independently, which means that a virtual network can use communication protocols designed to a specific service environment. These characteristics provide network operators flexible and dynamic way to manage and modify networks as well as provision flexible service than is currently available on the Internet [2].

The virtual network infrastructures vulnerable to many failures may happen in different layers of the network such as in the physical layer, where a connection failure due to a fiber cut may have a physical disconnectivity. Virtual network infrastructures are susceptible to various component failures such as single link failure [3], node failure and multiple link failures [4]. Seventy per cent of link failures are single link failures [5], and data centres experience 10 times more link failures than node failures [6]. The failure of the link could be due to maintenance, policy change and the substrate links or nodes may not function correctly all the time. Unplanned failures represent 30% shared between the router and optical fibre failures,
and the remaining 70% of unplanned failures are individual link failures due to different type of problems [5].

Because many virtual networks run on a shared physical network with limited network resources, a failure in the physical node, the physical network, or both the node and the network can affect many virtual networks. In addition, because multiple virtual networks share substrate network resources among many infrastructure providers of unknown reputation, it is very common for a client to suspect if the data are secure. This is an important security issue which is extensively researched in the cloud computing [7]. Network virtualisation requires the resolution of many challenges, in particular, reliability assurance. How to assure the virtual infrastructure components are dependable to continue deliver communication in the event of failure in VNE is an important and open problem.

1.2. Thesis Aims

While network virtualisation provides greater flexibility, it poses challenges from a reliability (i.e., fault-tolerance, trust and security) perspective. The overall aims of this thesis are to study the problem of virtual network reliability and develop efficient solutions that assure virtual network reliability.

The specific aims of this research project are to:

i. Analyse the reliability of virtual network links and develop a new approach to enhance virtual network link reliability

ii. Analyse the reliability of virtual network nodes and develop a new approach to enhance virtual network node reliability

iii. Analyse the reliability of the combination of virtual network links and nodes and develop a new approach to enhance the reliability of virtual network links and nodes.
iv. Minimise the unpredicted failures and reduce the impact of failure on a virtual network, by developing a dynamic solution for detecting a failure before it occurs in the VNE.

v. Predict failure and establish a tolerable maintenance plan before failure occurs in the VNE and avoid service interruptions by developing a prediction mechanism to forecast the failure in the VNE.

1.3. Motivation

Network virtualisation allows users to send their own virtual network specifications to their service provider who then maps each user’s request into the infrastructure provider’s hardware. Virtual network embedding is the process of allocating substrate network resources to the virtual network request while taking into account the processing and bandwidth capacity requirements. A virtual network is created by virtualising the network node and network link resources of a substrate network.

We now illustrate through an example of embedding virtual network onto the substrate network as shown in figure 1-1. The system of interest has a set of \( \mathcal{P} = \{PN_1, PN_2, ..., PN_n\} \) physical nodes and a set of \( \mathcal{L} = \{PL_1, PL_2, ..., PL_m\} \) physical links. A virtual network (VN) is created by virtualizing the \( \mathcal{P} \) nodes and the \( \mathcal{L} \) links of a substrate network (SN). As demonstrated in figure 1-1, The process of virtualization of the nodes and the links will create a set of \( \mathcal{V} = \{VN_1, VN_2, ..., VN_x\} \) virtual nodes and a set of \( \mathcal{W} = \{VL_1, VL_2, ..., VL_y\} \) virtual links. Each \( VL_i \in \mathcal{W} \) and each \( VN_k \in \mathcal{V} \) is mapped to a substrate node and substrate path respectively [8], [9], [10]. The physical nodes are represented as (rounded rectangle) and denotes \( [PN_1, PN_2, PN_3, PN_4] \) and the physical links are represents as (solid black lines) and denotes \( [PL_1, PL_2, PL_3, PL_4, PL_5, PL_6] \). In addition, the virtual nodes are represents as (router symbol) and denotes \( [VN_1, VN_2, VN_3, VN_4] \) and the virtual links are represents as
Introduction

(dashed lines) and denotes \([VL_1, VL_2, VL_3, VL_4, VL_5]\). For example, virtual node \([VN_2]\) is mapped to physical node \([PN_1]\), virtual node \([VN_2]\) is mapped to physical node \([PN_2]\) and the two virtual networks communicate between each other’s by using the virtual link \([VL_1]\), which is mapped to physical link \([PL_1]\). The mapping of virtual nodes to the physical node \([PN_1]\) is valid if and only if the total CPU capacities of VNs less than or equal to the CPU capacity of physical node. In addition, the mapping of virtual links \([VL_1, VL_2, VL_3]\) to the physical link \([PL_1]\) is valid if and only if the bandwidth capacity of virtual links less than or equal to the bandwidth capacity of physical link. In another words the mapping is valid when the capacity constraints of both virtual network requests do not exceed the capacities of the physical network.

A virtual network topology is created by connecting multiple virtual nodes through multiple virtual links (see figure 1-1, dashed lines). Multiple virtual topologies with varying characteristics are created and co-hosted by the same substrate network. Thus, virtual nodes are interconnected through virtual links forming a virtual topology. This allows many VN topologies with different characteristics to be established and coexist on the same physical hardware.
Multiple virtual networks can run on a shared substrate network with constrained network resources such as bandwidth and CPU capacity, as well as different configurations and requirements. Consequently, a failure in either the physical node or physical network or both the node and the network can affect many virtual networks. Because multiple virtual networks run on a single shared substrate network, a failure in the substrate network will affect the many virtual networks mapped onto the failed physical network. For example, figure 1-1, shows three virtual links (i.e., VL1, VL2, and VL3) are mapped to PL1. In event of physical link failure PL1, all of the virtual links that share that physical path will fail. Hence, failure of a single link substrate could affect all of the virtual networks that depend upon that link. Similarly, in event of failure of one physical node, all virtual nodes that share that physical node will fail.

A failure in the infrastructure virtual environment could contribute to substantial loss of important data and the additional use of many other resources such as time and cost [11]. Because the failed virtual network requires remapping to a substrate network, the virtual network needs to be reconfigured according to the particular requirements. This may ensue in
an economic penalty for infrastructure providers, due to a breach of service level agreements with service providers [12]. For example, in 2010 the online businesses in North America lost more than 26.5 billion in revenue due to service downtime [13]. Infrastructure providers try to reduce the cost of hosting an individual virtual network by accepting more virtual network requests. A failure of a physical network will not simply minimise the long-term revenue by accepting more virtual network requests. A physical network failure could drastically cut the profit of an infrastructure provider because the time available for virtual network hosting will be lessened [14].

Since a failure in the substrate entity (link or node) can affect all of the virtual networks that share the failed substrate entity. Therefore, it is important and challenging problem to provide dependable virtual network in event of failures occurred in the substrate network. Substrate network failure reduces the reliability of virtual network, this mean that reliability assurance is missing from virtual network embedding and it is required when a failure occurs in the virtual network infrastructures. Reliability in virtual network is an important issue for service provider and infrastructure providers. Reliability refers to the probability of all critical components of a virtual network remaining in operation.

1.4. **Research Problems and Major Contributions**

This thesis studies the substrate node and link failure problem and develops a reliable solution that guarantees a virtual network’s resilience and reliability in the event of substrate node or link failure. The major contributions of the thesis are as follows:
1.4.1. Handling Substrate Link Failures

Substrate link failures can occur in different layers of a network. For example, at the physical layer, link failure due to a fibre cut may cause a physical disconnection. The failure of the link could also be due to maintenance, policy change or the substrate links or nodes not operating properly all the time. Twenty per cent of all failures are due to scheduled network maintenance activities [3]. Of the unplanned failures, 30% are due to router or optical fibre failures, and 70% are individual link failures affected by a diversity of problems [3]. Several virtual networks share the substrate resources, and therefore, failure in a single link substrate will affect all of the virtual networks that depend upon that link. Thus, assurance of reliability is required when a physical link failure occurs.

1.4.2. Handling Substrate Node Failures

Usually, several virtual nodes are mapped onto a substrate physical node. Thus, failure in a substrate node will affect all of the virtual nodes that have been mapped onto that substrate node. Node failures in data centres due to maintenance and link failures happen about 10 times more often than node failures [4]. The consequences of substrate failure are that the network operator will incur more overhead costs because all of the failed virtual nodes need to be re-mapped to a different substrate network. Substrate node failure also reduces the operational time for hosting virtual network nodes, thereby reducing the reliability of the virtual networks. Existing studies have failed to optimise the use of resources because most use additional resources as a backup node to recover the fail node. Gill et al. [6] show that redundancy in the network is only 40% effective in reducing the impact of failure. A resilience approach that can deal with physical substrate node failures is required to ensure reliability of a virtual network.
1.4.3. Handling Correlated Substrate Link and Node Failures

The combination of link and node failures in a substrate network is also an important and a real problem. Failures in both substrate links and nodes will affect all of the virtual networks mapped onto that substrate. The failure will affect both the infrastructure provider and the network provider. The infrastructure providers may incur economic penalty due to the failure to provide the required quality of service requested by the network providers. In addition, the network provider fails to provide the required level of service to the end user whose virtual network service is unreliable. Therefore, assurance of reliability for virtual network is required when both physical link and node failure occurs.

1.5. Significance of Contributions

This thesis makes a significant contribution to the network virtualisation knowledgebase in general and our understanding of reliability and resilience in particular. The main research questions that are investigated in Chapter 3, Chapter 4 and Chapter 5 of this thesis are as follows:

i. What is the probability that the substrate network functions?

ii. How to make the physical network reliable with the least resources?

iii. How to detect if a component of a substrate network functions properly?

iv. How to predict when the failure occurs in the substrate network?

The first contribution of this thesis is a framework to estimate the probability of the substrate network failure by measuring the mean time to failure (MTTF) of the underlying infrastructure. A virtual network without protection from node or link failures could lead to virtual network service interruption. Therefore, an ideal virtual network platform proposed that supports efficient mapping in the presence of failures (node failures and network failures) and
offer high-availability. By adopting different redundancy mapping techniques, such as simple mapping, passive mapping and active mapping, we can achieve optimal reliability design for the virtual network allocation onto the physical network. The framework approach allows the virtual network provider to specify the level of service according to the reliability level of the virtual network.

The second contribution of this thesis is methodology check if the components of a substrate network function properly. A dynamic solution developed for detecting a failure before it failure occurs in the substrate network. The failure detection mechanism detects a fault in virtual infrastructure components by exchanging a message between two neighbouring nodes. In the detection mechanism, a small number of messages are exchanged for failure detection because the virtual network environment (VNE) topology is partitioned into clusters. Our detection mechanism achieved very efficient time to detection of a failure and high accuracy in detection of failure in VNE components.

The third contribution of this thesis is methodology to check when the failure occurs in VNE components. A forecasting mechanism developed that predicts failure in a substrate node or link to avoid service interruptions. The failure prediction method accurately predicts failure in the substrate link, substrate node and virtual network.

1.6. Research Methodology

To evaluate the probability that the substrate network functions, our proposed framework uses reliability block diagrams and continuous-time Markov chains to identify the best design that can achieve more reliable VNE. The reliability block diagram technique considers the configuration differences in the virtual network based on the series and parallel component connections in the VNE to evaluate the dependability risks in the VNE. For example, in series, the system required \( M \) out of \( M \) components of the substrate (nodes + links) to function, while
in parallel, the system required 1 out of $M$ components to function. Nodes and links can be connected in many different ways, such as series, parallel and a combination of series and parallel connections. In series connections, the physical network fails if any one of its components fails, while in parallel connections, the physical network fails if all of its components fail. We use series and parallel arrangements in a reliability block diagram to represent the three mappings: single mapping, passive mapping or active mapping. In single mapping, each virtual network maps onto a single physical network without redundancy backup. In passive mapping, each virtual network maps to active primary physical network and an idle secondary physical network and it can be activated if the primary virtual network fails. In active mapping, each virtual network is mapped onto both a primary and a secondary physical network which they are simultaneously active. Reliability block diagram used to compute the reliability metric to determine the reliability of the physical components to guarantee providing the required level of reliability.

A self-healing approach is developed to overcome a failure in a virtual network. By mapping a virtual network component onto multiple substrate network components, the virtual network can be activated on a stand-by substrate component in the event of failure of a substrate network component. Thus, the virtual network avoids service interruptions before the substrate node or link failures occurs. A continuous-time Markov chain model is used to represent redundant and non-redundant architecture in a VNE. In addition, the continuous-time Markov chain model is used to evaluate VNE reliability, with and without redundancy. The reliability of the system is evaluated quantitatively and qualitatively by measuring the MTTF of the underlying infrastructure components. The lifetime of a virtual network can be increased by mapping virtual network components onto more than one substrate network components. If the time $T$ to failure of a physical network is exponentially distributed with the failure rate $\lambda$, then the reliability of the physical network is increased when the MTTF is increased for each
physical component. Thus, we can improve the reliability of the physical network by increasing the reliability of the physical network components to function from time $t$ to $t + \delta$. The proposed framework allows a virtual network provider to specify the required level of service because the reliability of the virtual network becomes a service provided by the infrastructure provider. Thus, we can determine the reliability that satisfies virtual network provider according to how many resources have to be allocated to provide the required level of reliability to guarantee virtual networks resilience in the event of substrate node and link failures.

To overcome the failure in a VNE and improve a virtual network reliability, a dynamic detection system developed to detect failures in a VNE. To check if a component in the system is functional or non-functional, the fail-stop behaviour is chosen in the fault model detection system to represent failure in the VNE. Failure is detected using a message-passing interface that exchanges messages between neighbouring nodes to check whether they are working. In addition, the conservative time-synchronisation algorithm is used to determine the time-out before considering that an event failure has occurred in a VNE. The failure detection mechanism can cope with a large-scale failure and can prevent overloading the virtual network by reducing the number of messages for failure detection. The failure detection mechanism can be used to study the cause of the failure and analyse the effectiveness of redundancy. It uses fewer resources to recover the failure and study the impact of substrate node failure in a virtual network.

To minimise unforeseen failures in VNE, a prediction mechanism developed to predict failure occurrences in a VNE. The prediction mechanism that forecasts the time to failure (TTF) of virtual infrastructure components. The prediction mechanism is based on time series and support vector regression (SVR) models to forecast failure in a substrate node or link and avoid service interruptions. The accuracy of the prediction mechanism using SVR model is very high because the error rates are very low, as measured by root mean square errors (RMSE).
1.7. Thesis Organisation

The rest of the thesis is organised as follows:

Chapter 2 explains the different architectures of a virtual network (i.e., the virtual network could be created by virtualising one or more layers such as the physical layer, link layer, network layer or application layer). It also provides a brief literature review about improving virtual network reliability in the event of failure.

Chapter 3 introduces a framework to estimate dependability risks in VNEs that considers variations in virtual network configurations. The framework uses the reliability block diagrams and the continuous-time Markov chain model to analyse the dependability level of a virtual network.

Chapter 4 presents a mechanism to detect and overcome a failure in a virtual network. Failure is detected using a message-passing interface and the conservative time-synchronisation algorithm. A message-passing interface is used for probing connections between point-to-point nodes by message exchange and the conservative time-synchronisation algorithm is used to determine the time-out before considering than an event failure has occurred in the VNE.

Chapter 5 introduces a prediction mechanism to predict the future failure before the failure occurs in VNE. The main concept of this approach is to forecast the TTF of VNE components by using time series and SVR models.

Chapter 6 summarises the major findings, discusses the accomplishments of this work and highlights possible future research directions.
Chapter 2: Literature Review

Virtual network reliability and resilience is an important issue because a failure in the substrate network can affect multiple virtual networks that run on a single shared physical network. Network virtualisation requires resolving many challenges, and one of the main challenges is reliability. In this chapter, a literature review is presented of the reliability and resilience of virtual network when failure occurs. Different approaches and algorithms are assessed in terms of efficiency and robustness against a failure.

2.1. Introduction

A virtual network is created by virtualising the network node and network link resources of a substrate network, as shown in figure 2-1. Substrate refers to hardware (e.g., 10G line), software (e.g., open shortest path first protocol), and logical or virtual resources (e.g., addresses). In a virtual network, virtual nodes are interconnected through virtual links to form a virtual topology. This allows many virtual networks with unpredictable characteristics to be established and coexist on the same physical hardware. By dynamically mapping virtual resources onto physical hardware, the advantages of the hardware can be maximised.

There are many potential sources of failure in a physical network such as link failure and node failure, and node failure causes all adjacent links to fail. In addition, failure can occur in either the physical layer or the virtual layer. A failure in the physical layer will propagate to the virtual layer, while a failure in the virtual layer will only affect the virtual layer. A failure in the infrastructure of a virtual environment could lead to a loss of important data and be time consuming and costly. A failed virtual network requires re-mapping to a substrate network and reconfigure the virtual network according to the particular user’s need. Any failure in a virtual network will minimise the profit of an infrastructure provider because the time available for
hosting a virtual network will be decreased. Therefore, virtual network reliability assurance is a significant and unresolved problem.

Mapping a virtual network is challenging because of node and link constraints such as CPU resources and geographical location (for the virtual node and delay for the virtual link). Because virtual network mapping is a NP-hard problem [15],[16], a variety of heuristics have been developed in the literatures [9, 17-21]. As noted in [22], restrictions such as the performance requirements of the virtual resources should be considered during the mapping process, for example, a 1,000 MBit/s virtual link cannot be mapped to a 1,00 Mbit/s substrate link. Admission control by infrastructure providers to reject or accept a virtual network mapping request is based on the limited resources of the substrate network. The virtual network request for CPU capacity for each node should be less than the CPU capacity for the substrate node. In addition, for the virtual network request to be successfully mapped, the bandwidth for each virtual link should be less than the bandwidth of the substrate link or the substrate path. A virtual network mapping into substrate network could be static mapping (i.e. without any change in substrate network) or dynamic mapping (i.e. take into consideration any change in virtual networks and substrate network). Online virtual network mapping is more complex than off-line mapping because online mapping is unpredictable and it is difficult to search the entire substrate network to allocate the resources required for the virtual network. A virtual network requests arrive online and are embedded while others expire and release their resources from the substrate network. In a static virtual network embedding approaches do not consider the probability of remapping one of more virtual network request. In static virtual network embedding, fragmentation of substrate network resources occurs because the new arrival virtual network cannot be embedded into released substrate network resources from previous mapped virtual network [23]. Thus, several effects lead to a need for relocation virtual network to different substrate network resources. If the resources of the substrate network are not first
defragmented by reconfiguring the already mapped VN, the ratio of accepted virtual network requests diminishes and the long-term revenue reduced. This can be amended if the fragmented resources are consolidated by using dynamic virtual network embedding approaches to reconfigure the mapped virtual network requests in order to rearrange the resource allocation and optimize the utilization of substrate network resources[24].

All the above-mentioned factors hardware failure, embedding method, fragmentation the substrate network resources resulting rejecting many virtual network requests and hence reducing the virtual network reliability. Virtual network reliability refers to the ability of the completely virtual network provide continuous service even in the event of failure of a component in the VNE. In this chapter, the conceptual virtual network architecture is reviewed, followed by a review of the reliability of virtual networks due to link or node failure and due to combined link and node failures.
2.2. Conceptual Virtual Network Architecture

A virtual network is a set of virtual nodes and virtual links that uses a single physical infrastructure to provide multiple logical networks [25]. Each logical network supports its users through a customised set of protocols and functionalities. Network virtualisation uses software-based abstraction to separate network traffic from the physical components of the network [26]. Virtualisation could be implemented in one or more layers such as the physical layer, link layer, network layer and application layer. We will highlight the following four virtual network architectures: Virtual Network Infrastructure (VINI), Concurrent Architecture Better Than One (CABO), A Lightweight Approach for Viable End-to-end IP-based QoS Services (AGAVE) and Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures (FEDERICA). VINI and FEDRICA are link layer virtualisation architectures, CABO is full virtualisation architecture and AGAVE is a network layer virtualisation architecture.

2.2.1. Internet In a Slice Architecture

Internet in a Slice is an example of network architecture that was implemented by PlanetLabs on an initial VINI by combining a collection of available software components [27]. Internet in a Slice can be contemplated as a particular instantiation of an overlay network that runs software routers and permits multiple overlays to be in parallel. Internet in a Slice consists of the following five modules: forwarding engine for the packets, control plane, a method for clients to communicate with the overlay, processes for exchanging packets with servers and a group distributed machines on which the overlay is implemented. Internet in a Slice operates by using many open-source components, including the XORP open-source routing protocol for its control plane [28], the Click modular software router for packet forwarding and network
address translation [29] and OpenVPN servers to connect with end users. VINI is an overlay network that runs software routers and lets many overlays to work in parallel, but it is considered an unreliable virtual network because when the software crashes, the entire virtual network will fail. Nevertheless, VINI is used as a practical platform for evaluation and managing new protocol and services in virtualization network prepared by researchers.

### 2.2.2. CABO Architecture of Future Internet

CABO is a high-level design hardware-based network virtualisation architecture. The CABO architecture provides separation between infrastructure providers and service providers that eases the manageability of a virtual network. CABO is an example of the future Internet in which functionalities in a networking setting are decoupled through dividing the role of the traditional Internet service provider into two roles [2]. The first role is the infrastructure providers who own and maintain the network equipment (e.g., routers and links). The second role is the service providers who construct virtual networks by combining resources from multiple infrastructure providers and offering end-to-end network service to users [2]. This new Internet architecture allows service providers to choose the service in a cost-effective manner from different infrastructure providers without needing to invest in physical infrastructure. This decoupling provides the service providers with the flexibility to develop multiple heterogeneous networks as a virtual network to be hosted on a shared physical network. This allows service providers to provide multiple Internet access technologies for each user. CABO architecture enables a reliable virtual network by supporting guaranteed migration of virtual routers from one substrate node to another in event of a failure occurs in the substrate node. CABO architecture is the full virtualisation network, which allows the user to choose a virtual network from different infrastructure providers. There are some disadvantages to CABO architecture, such as not offering wide network control and management planes.
2.2.3. AGAVE

AGAVE architecture offers end-to-end provisioning of quality of service-aware services over IP networks. AGAVE is based on the idea of network planes which allow various IP network providers to build and offer parallel Internets designed according to the required service by the end user. Network planes are designed to meet the service providers’ requirements for different services and have engineering processes for routing protocols and adapting the capability of traffic with different end-to-end quality of service expectations. Network planes are interconnected with parallel Internets that enable end-to-end services over multi-provider Internet network providers [7]. This architecture leads to a more reliable virtual network because it allows the virtual network to connect to multiple IP network providers. AGAVE increases reliability by replacing a node-centric configuration approach with a more centralised network-based configuration that ensures consistency between participating IP network providers and decreases misconfiguration errors.

2.2.4. FEDERICA

FEDERICA architecture is a link of the virtualisation layer [30]. FEDERICA node facilities contain the programmable routers or switches that allow a logical router or switch to connect at the core nodes. FEDERICA architecture provides some level of security because of its centralised admission control through a dedicated proxy which is maintains user slices secure from unauthorised access. However, complete user control to the lowest possible layer introduces a vulnerability to the virtual network. FEDERICA is not very reliable because each virtual node and link maps to the substrate node and link respectively. In event of a virtual node failure, a new virtual node is created on the same substrate node or different substrate node in the same cluster. In event of a substrate node failure, the virtual node is migrated to different substrate node failure. Table 2-1 shows the differences between the VINI, CABO, AGAVE and FEDERICA architectures.
2.3. Review of Reliability of Virtual Network Due to Substrate Link Failure

The different approaches to improving the reliability of a virtual network include re-mapping the virtual network with a backup or a recovery mechanism before or after the failure occurs in a substrate link. Because multiple simultaneous failures not often occur in the real world, the following approaches have been introduced to protect against any single substrate link failure before or after a failure occurs.

The first approach uses two shared backup network provision mechanisms for virtual network embedding [31]. The first backup is shared on-demand and the second backup is shared pre-allocation. The first backup is used as a backup resource allocation after receiving a virtual network request and the second backup is requested during configuration and before any virtual network request. Both backups are constructed with supporting bandwidth when the virtual network request is mapped. The advantage of the first backup is that it is a good technique for sharing bandwidth in the case of link failure. The first backup minimises the...
usage of communication resources and maximises the profit of the infrastructure provider by increasing the time that the virtual network is available to the service provider. The disadvantage of the second backup is that it is inefficient at low virtual network request loads because it always holds the backup bandwidth regardless of a virtual network request.

Survivable virtual network embedding is a reactive backup mechanism that has been prepared for virtual network mapping to protect against single substrate link failure [32]. In the reactive backup mechanism, the bandwidth of a substrate link is shared between primary flows and backup flow, primary flow is reserved for transport in the normal situation and the backup flow is reserved for transport upon failure occurs in primary flow. When a failure occurs in the substrate link failure, a reactive backup mechanism is used to rerouting the affected traffic by using the allocated backup bandwidth of other links. The disadvantage of this mechanism is that more resources are needed because each substrate link requires a backup path to protect against any failure. The backup mechanism cannot assure 100% recovery with an increase in traffic load and a great amount of data loss due to failure occurrence in VNE. In addition, the bandwidth resources are used for new virtual network requests and there may be insufficient resources left on for recovery. While the outcome of various bandwidth sharing for the substrate links has been assessed, it did not offer a procedure to find the best sharing.

An algorithm proposed for restoration of a single link failure involves adopting an intelligent bandwidth sharing mechanism [33]. The algorithm uses existing embedding techniques [19, 34] for mapping virtual networks to substrate networks with the restoration path selection to be used as a backup path in the case of a single substrate link failure. Online virtual network service resource allocation is used to minimise the joint failure probability between the primary path and the backup path. The advantage of this work is that it offered a solution to the complex of minimising network resource usage while allocating sufficient resources to handle the failure. The disadvantage is that it minimises network resource usage
and this could increase the number of rejections of virtual network mapping requests.

Protecting the substrate link failure by using $N$ facility nodes as primary mapping for the virtual nodes and $x$ facility nodes as backup for virtual nodes, after the substrate failure occurs in the $N$ primary node, the virtual node migrates to one of the $x$ backup nodes [35]. In addition, the proposal introduced in [35] used $N + x$, $x \geq 1$ facility nodes and $E + y$, $y \geq 1$ substrate paths to protect virtual network against a link failure in the substrate network. The advantage of this method is that it minimises the resources used by the substrate facility node when a failure occurs in the substrate node and the virtual node is allocated to another substrate node. The disadvantage of this method is that allocating redundant links to enhance the virtual network may consume a lot of bandwidth that may not be used if a failure does not occur.

Another approach has been proposed to tolerate substrate link failure by optimising network and computing resources and extending the shared protection mechanism by combining a node migration method [36]. Node migration is used to move a mapped node onto another facility node in the event of a substrate link failure. The advantage is that the migratory shared protection mechanism is safer than a traditional backup technique. The relocated node saves resources because it needs a shorter backup path length to the destination node before the migration. The disadvantage is that because of the cost of using computing and communication resources, node migration backup protection is more expensive than tradition backup protection. Therefore, traditional backup protection is preferred over migration backup protection.

An embedding algorithm prepares to recover the substrate link failure first by mapping the virtual node to a specific substrate node and then mapping the virtual link over multiple substrate paths with flexible path splitting ratios [19]. Online request mapping is introduced by path splitting and migration of an inefficient substrate path to a different path using different splitting ratios for each path. The advantage of this approach is that it introduces a solution to
the link failure by path splitting and path migration over multiple substrate links with flexible path splitting ratios. In addition, the algorithm introduces optimisation for cost-effective virtual network embedding by allowing substrate path splitting and migration for better resource usage. The disadvantage of this approach is that because the mapping task is achieved in two steps, it scales down the operation of the virtual network mapping and requires more time. Furthermore, the algorithm is concerned with link remapping without any solution of its relation to the node remapping in the event of failure.

A failure could involve an entire computing cluster or just one or more processors that are executing a specific task with no spare processors left in the same cluster. The failure could be due to a power outage or occur in either the hardware or the software. A technique has been proposed to recover a link failure in a wavelength-division multiplexing network [37]. This technique is used for fault tolerance and involves the migration of the task to a spare cluster with sufficient light path connectivity and the existence of other clusters processing in the same distributed computing job. In [37] the problem formulated as integer linear programming to find an optimal virtual private network that can satisfy the traffic requirements. However, when the link failure in the virtual private network remains connected there is no guarantee that the remaining virtual private network connection can support the required traffic matrix.

An effective resilience virtual network mapping against substrate link failure while providing enhanced quality of service can be achieved by allocating backup paths that do not share common links in the substrate network with their related operating paths [38]. The algorithm introduced in [38] maps virtual nodes onto substrate nodes sequentially by selecting substrate nodes with higher quality. After mapping each virtual node, the virtual links are mapped onto substrate paths with backup paths. The heuristic in [38] is similar to the heuristics in [39, 40] but has some different features: firstly, the number of intermediate substrate candidate nodes for link mapping is limited to two; secondly, backtracking is required for
virtual nodes mapped previously when the current virtual node cannot be mapped using the sequential mapping procedure. Moreover, the heuristic in [38] is different from the other heuristics in [39, 40] because they provide improved quality of service and resilience against substrate network failures. The disadvantage of this heuristic is that it suffers a high run-time if backtracking is uncontrolled if there is no solution occurs.

Increasing the reliability of a virtual network in the event of failure, an alternative mechanism constructs high-quality one-hop routes via intermediary virtual nodes [39-42]. To obtain a high quality of service mapping of virtual nodes to the substrate nodes, only the direct path between two substrate nodes is taken. The alternative routes serve as a backup for direct virtual network routes and provide improved reliability against changing network conditions. The quality of both paths (direct and indirect) is high enough to meet or exceed application quality of service constraints, and an application can use either of these paths without disrupting quality of service requirements for loss rate and message delay. This approach combines quality of service with the resilience of a virtual network, but it is not an efficient mechanism for using substrate resources with specific quality of service demands while leaving the other resources unusable.

Table 2-2 summarises the previous work on increasing the reliability of virtual networks in the event of physical link failure, which is the predominant failure type in virtual networks.
Table 2-2 Assuring Resilience of Physical Link Failure in a Virtual Network

<table>
<thead>
<tr>
<th>Reference</th>
<th>Resilience Mechanism</th>
<th>Research Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared backup network provision for virtual network embedding [31]</td>
<td>Resilience link failure before a failure occurs by provision two shared backup</td>
<td>It is inefficient mechanism since reserve virtual infrastructure resources as a backup before virtual network request arrive.</td>
</tr>
<tr>
<td>Survivable virtual network embedding [32]</td>
<td>A restoration mechanism to protect against a single substrate link failure</td>
<td>The restoration mechanism cannot guarantee 100% recovery because the backup activated after the failure occurs.</td>
</tr>
<tr>
<td>Resilient virtual network service provision in network virtualization environment’s [32]</td>
<td>Reactive after failure (restoration) with optimisation objective minimise the path failure probability</td>
<td>The objective to minimise the network resources could decrease the number of virtual network requests.</td>
</tr>
<tr>
<td>Migration based protection for virtual infrastructure survivability for link failure [35]</td>
<td>Proactive before failure with optimisation objective minimise sum of costs</td>
<td>The cost of using computing and communication resources for migration as a backup protection is higher than traditional backup protection.</td>
</tr>
<tr>
<td>A novel virtual node migration approach to survive a substrate link failure [36]</td>
<td>Proactive before failure with optimisation objective minimise the substrate resources usage</td>
<td>Allocating redundant links to enhance virtual network consume a lot of bandwidth and may be not used in case of no failure occurs.</td>
</tr>
<tr>
<td>Rethinking virtual network embedding: substrate support for path splitting and migration [19]</td>
<td>splitting path over multiple substrate links with flexible path splitting ratios to recover link failure</td>
<td>The mapping task is achieved in two steps, which reduces the performance of virtual network mapping because it requires more time.</td>
</tr>
<tr>
<td>Multi-layer resilient design for Layer-1 VPNs [37]</td>
<td>This technique for fault tolerance is migration the task to spare cluster with a sufficient light path connectivity</td>
<td>When the link failure occurred, the virtual private network remains connected but there is no guarantee that the remaining connection of the virtual private network can support the required traffic matrix.</td>
</tr>
<tr>
<td>Achieving effective resilience for QoS-aware application mapping [38]</td>
<td>Allocating a backup substrate path for virtual network which doesn’t share common links with their corresponding working path</td>
<td>Required high run-time for backtracking if there is no solution exists.</td>
</tr>
<tr>
<td>Efficient and dependable overlay networks [39-42]</td>
<td>Constructs high-quality one-hop routes via intermediary virtual nodes. The alternative routes serve as a backup for direct virtual network routes and provide improved reliability against changing network conditions</td>
<td>It is not an efficient mechanism because it uses substrate resources with specific quality of service demands and leaving the other resources unusable.</td>
</tr>
</tbody>
</table>
2.4. Review of Reliability of Virtual Network Due to Substrate Node Failure

The following approaches have been introduced to improve the reliability of a virtual network, by remapping the virtual nodes into the substrate nodes with a backup or recovery mechanism before or after the failure occur at a substrate node.

A proposed two-step solution was introduced to restore virtual infrastructure from substrate node failure [43]. The first step is enhancing the virtual infrastructure with backup of the virtual nodes and links with spare computing and communication resources. The second step is mapping the enhanced virtual infrastructure to a substrate network. The virtual infrastructure is enhanced by two approaches 1-redundant and $K$-redundant virtual infrastructure with $N + 1$ or $N + K$ nodes, respectively. When a facility node fails, the virtual infrastructure node mapped to it is migrated to a backup facility node and the associated virtual links required to be migrated as well. In the 1-redundant scheme solution, one additional virtual infrastructure node is added. When the virtual nodes failed, then it will be migrated to the backup node as well as the connection of the failed node required to be migrated. In the $K$-redundant solution, each critical node has a corresponding backup node and the $K$-redundant virtual infrastructure nodes are then mapped onto the substrate nodes. The advantage of this method that it is very efficient in the event of failure because each critical virtual infrastructure node has a backup node that can be used to replace the failed node. This two-step solution has a significant impact on conserving backup resources and may improve resource usage by using redundant links when the facility node fails. The disadvantage is that by minimising the network resources, costs may increase because more resources are allocated for both active and backup nodes. In addition, the $K$-solution needs to reserve a backup node for every critical node and link to every adjacent node.
Introduced location constraint in virtual network mapping and an optimal resources allocation for active and backup to protecting any single substrate node failure in VNE [44]. The integer linear programming model was formulated to determine the optimal solution for resource allocation for operations and backup demand. For online mapping, a sequential survivable embedding algorithm has been proposed to resolve the problem in two steps. In the first step, the working address is mapped by adopting the embedding algorithm proposed in [17], and the second step is backup request mapping. The integer linear programming model was based on constructing a graph to map each virtual node to substrate nodes while satisfying location and capacity constraints. The disadvantage of the linear programming model is that it consumes many resources to check that all virtual nodes have been allocated to backup nodes. In addition, introducing the location constraint with the existing capacity constraint makes virtual network embedding more complicated.

A recovery mechanism called enhanced virtual network has been proposed for a single failure in a facility node due to power outage, virus attack, disk failure or software crash [45]. The enhanced virtual network uses a two-step approaches: the first step creates an enhanced virtual network by adding service nodes $N$ and additional service links $e$ such that $e \leq N$ to the virtual network. The second step involves mapping the enhanced virtual network to $N + 1$ facility nodes and $E + e$ paths in the substrate. When the service node is affected by a failure, the service node needs to be migrated to a backup facility node at a different geographical location. When any node fails, the role of the failed node will be taken up by other nodes after a rearrangement of all the nodes including the backup node. Graphical transformation or decomposition and bipartite graph matching is used to find the optimal path with the least computing and communication resources. The advantages of the enhanced virtual network design are that it requires a fewer virtual resources, such as bandwidth resources for links or computing resources for service nodes, after mapping the enhanced virtual network to the
substrate. The enhanced virtual network mapping is efficient because it shares resources among other nodes in the event of a failure. The disadvantage is that, if a failure occurs, a large number of virtual nodes require migration to the working nodes, which makes the approach less feasible in a large network.

A solution has been presented for solving the problem of survival virtual network mapping against any failure in facility nodes in a single region of a federated computing and networking system [46]. Facility nodes from a data centre are interconnected in a federated computing and networking system and need to be backed up to achieve a survival virtual network mapping. In [46] redundant facility nodes are used at different geographical locations and redundant links and has the provision to map to virtual infrastructure in case of failure. Two failure-dependent survival virtual network mapping algorithms have been developed. The first solves the non-survivable virtual network mapping problem with a heuristic, the second extends the heuristic to solve the survival virtual network mapping problem. The first heuristic is called separate optimisation with unconstrained mapping which is separating the problem into non-survival problems for each probable regional failure and one for primary functioning mappings. This minimises the costs of the resources used. The second approach is called incremental optimisation with constrained mapping and first maps the primary functioning mapping, and then maps each regional failure. The advantage of incremental optimisation with constrained mapping is that it is a more effective algorithm and minimises cost by using less resources. Separate optimisation with unconstrained mapping provides better failure recovery probability because it uses additional computing resources to overcome the failure. The disadvantage of a federated computing and networking system is that it has a constraint with computing and communication resources, and therefore, certain failures cannot be recovered. Moreover, the separate optimisation with unconstrained mapping algorithm requires recomputing virtual mapping of unaffected nodes, which takes time and costs more.
A service-aware approach groups multiple virtual machines and their backups to form a survival virtual infrastructure for a service \cite{47}. The problem is classified into two-sub problems. The virtual machine placement sub-problem uses an efficient backtracking algorithm based on a depth first search to calculate the virtual link mapping using a linear program. For the virtual machine placement sub-problem, the optimal mapping of survival virtual infrastructure to the physical data centre network, which is cost-effective subject to constraints in computing and communication resources use. For the virtual link mapping sub-problem a polynomial time algorithm is used to solve the bandwidth demands of virtual machines that can be guaranteed before and after the failure. The advantage of this approach is that the reserved bandwidth can be used as a backup in the event of link failure and may also share links. The disadvantage of this approach is that it has a high computing overhead due to the virtual machine placement problem that requires extensive calculations for virtual link mapping for a possible solution. This high computing overhead for a large network may not be guaranteed to get close to the optimum solution.

Table 2-3 shows the previous study done on increasing the reliability of virtual network in the case of physical node failure in a VNE.
2.5. Review of Reliability of Virtual Network Due to Substrate Link and Node Failures

There are different methods to improve the reliability of virtual network, for example, re-mapping the virtual network with backup or a recovery mechanism before or after the failure occurs at a substrate network.

A proposal has been developed to improve the reliability of virtual infrastructures by allocating sufficient computing resources when a failure occurs in either a substrate node or a link [21]. The opportunistic redundancy pooling mechanism overcomes consuming a large amount of physical infrastructure for backup because resources are pooled and shared across

### Table 2-3 Assuring Resilience of Physical Node Failure in a Virtual Network

<table>
<thead>
<tr>
<th>Reference</th>
<th>Resilience Mechanism</th>
<th>Research Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost-efficient design of SVI to recover from facility node failures [43]</td>
<td>A backup substrate node is reserved for all critical nodes as well as a backup substrate links to all neighboured nodes before failure the failure occurs.</td>
<td>Too many resources allocated for both active and backup for substrate links and nodes.</td>
</tr>
<tr>
<td>Location-constrained survivable network virtualization [44]</td>
<td>Offers survivability before a failure occurs by allocating backup nodes with location constraint</td>
<td>Consume many resources to check all virtual nodes have allocated backup nodes with location constraint make the mapping more complicated.</td>
</tr>
<tr>
<td>A novel two-step approach to surviving facility failures [45]</td>
<td>Migration of service node to backup facility node located at a different geographically location in the event of a failure</td>
<td>A failure in a large amount of virtual node required a large migration of working nodes makes this approach less applicable in the large network.</td>
</tr>
<tr>
<td>Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures [46]</td>
<td>Migration service node to a backup facility node with a backup link at different geographical location</td>
<td>Since the mechanism has a constraint with computing and communication resources, therefore certain failure maybe not recovered. The mechanism required computing virtual network remapping which mapping results in more cost and more time wasted.</td>
</tr>
<tr>
<td>Survivable virtual infrastructure mapping in virtualized data centres [47]</td>
<td>A service-aware approach by grouping multiple virtual machines and their backup to form a survival virtual infrastructure for a service</td>
<td>Required high computing overhead for calculation virtual machine placement.</td>
</tr>
</tbody>
</table>
Literature Review

multiple virtual infrastructures. Opportunistic redundancy pooling ensures that virtual infrastructures limit the connection of redundant nodes in the links. Reliability is increased when the number of backup nodes increases. Opportunistic redundancy pooling shares these redundancies for both independent and cascading types of failure by reducing the number of backup nodes and increases reliability by sharing backup resources with other virtual infrastructures. The advantage of opportunistic redundancy pooling is that it minimises redundant resources for backup by reducing the computing and communication resources that are used by the virtual infrastructures. The disadvantage is that the mechanism for backup recovery is not efficient because it allocates backup resources before a failure has occurred and does not provide a solution for unexpected failure.

There are three types of resource failures: virtual node failure, substrate node failure and link failure. A distributed fault-tolerant embedding algorithm has been proposed to detect and identify local changes through monitoring node or link failure and finding new resources to maintain virtual network topologies [48]. The monitoring is based on a multi-agent approach to guarantee distributed negotiation and synchronisation between the substrate nodes [49]. In the event of failure in each substrate node, the agent selected the substrate node attributes that should be matched to the virtual node attributes. Each agent computes a dissimilarity metric between non-functional attributes requested by virtual node and the non-functional attributes of its associated substrate node. The non-functional attributes may be different types, such as binary, nominal or interval [50]. The advantage of this approach is that it handles failure of a virtual node, substrate node or link, as well as monitoring and detecting any failure autonomously and informing other substrate nodes about the failure. The distributed fault-tolerant embedding algorithm replaces the failed node or link using available resources. The disadvantage of this mechanism is that in the event of failure, the search procedure for finding match resources for the virtual network is repeated, which makes the algorithm inefficient with
increased overhead.

A concurrent failure can occur in a computing cluster due to power outrage, virus attack or link failure due to a fibre cut. Two technique have been developed to recover concurrent multi-layer failures in the cluster or a link [51]. The first technique is called cluster and path protection and the second technique is called virtual network protection. Cluster and path protection is a mechanism to protect each logical connection from a link failure by establishing two disjointed paths and two clusters to survive any single cluster failure. Virtual network protection uses three disjoint clusters and makes provision to survive one link failure and one cluster failure. The advantage of the cluster and path protection method is that it is a first offer recovery mechanism for a multiple clusters or links and has introduced a concurrent recovery facility to the substrate node and link. The disadvantage of cluster and path protection is that it takes more bandwidth resources because a logical link in the cluster and path protection can share physical links with virtual network protection that requires more CPU resources. Consequently, these mechanisms make mapping more complex due to different resource isolation and the study did not determine which approaches perform better for an existing virtual network.

A hierarchical and heterogeneous modelling to depict redundant architectures and compare their availability taking in account computers acquisition costs [52, 53]. A hierarchical and heterogeneous are based on RBD and Markov chains, a high-level model based on RBD denotes the Eucalyptus platform subsystems and a low-level model based on Markov chains represents the respective subsystems employing warm standby replication. In the analytical models, the failure in hardware and software are considered in the cloud computing [52, 53].

A framework is proposed to specify the virtualized infrastructures allocation that takes into consideration the reliability support in virtual networks [54]. The framework has a specification language, which describes the reliability metric to be adopted in a resource allocation
algorithm. The disadvantage of this study is that it does not offer dependability model for evaluation the general assessment risk and the maintenance is not considered.

A cloud dependability model that uses system-level virtualisation is proposed in [55], but this work focuses on cloud security and evaluates the virtualised component dependability properties at the system level. The proposed reliability block diagrams to assess the system reliability of cloud computing. The drawback is that dependability is assessed only at the host level and the model is too simple to describe the complex behaviour of underlying hardware as well as software components.

A framework proposed to model and evaluate the dependability of a virtual network based on the reliability block diagrams and continuous-time Markov chains [56]. The proposed framework will be helpful to the design and construction of more dependable. The important characteristic of continuous-time Markov chain models is the representation of system behaviour along the time scale. The continuous-time Markov chain model was chosen for it is greater simplicity than discrete time models. If time is discrete, the model has to consider that multiple events may occur between two consecutive time marks and search the effects of all possible combinations of these events. Continuous time scale models use appropriate probabilistic assumptions and it is possible to take only one event into consideration [57].

Table 2-4 summarises previous work on increasing the reliability of virtual network in the case of combination physical link and node failure in virtual network.
### Table 2-4 Assuring Resilience of Physical Link & Node Failure in a Virtual Network

<table>
<thead>
<tr>
<th>Reference</th>
<th>Resilience Mechanism</th>
<th>Research Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designing and embedding reliable virtual infrastructures [21]</td>
<td>Recover a failure occurs in either a substrate node or a link by allocating sufficient computing resources by using opportunistic redundancy pooling to pool and share across multiple virtual infrastructures</td>
<td>It is not an efficient mechanism since backup resources allocated before failures occur. Moreover it recovered only one node failure thus, the mechanism cannot be applied when more than one node failure</td>
</tr>
<tr>
<td>Adaptive virtual network provisioning [48]</td>
<td>Monitoring substrate node and link failure and finding new resources to maintain the virtual network topology</td>
<td>The matching virtual network procedure that repeated again that made the algorithm inefficient.</td>
</tr>
<tr>
<td>Robust application specific and agile private (ASAP) networks withstanding multi-layer failures [51]</td>
<td>A mechanism to protect against link and cluster failure by establishing two disjoint paths and two clusters to survive from any single cluster failure</td>
<td>The mechanism makes mapping more complex due to the different resource isolation.</td>
</tr>
<tr>
<td>An Availability Model for Eucalyptus Platform, Models for Dependability Analysis of Cloud Computing Architectures for Eucalyptus Platform [52, 53]</td>
<td>a warm-standby replication mechanism is considered to protect both hardware and software failure in cloud computing environment</td>
<td>The study only consider the dependability and cost in designing cloud infrastructure without evaluation the performance which is very important metric in cloud computing</td>
</tr>
<tr>
<td>Reliability Support in Virtual Infrastructures [54]</td>
<td>A framework is proposed to efficiently specify and control the reliability of the virtualized infrastructure components at runtime.</td>
<td>The framework is not considered the general assessment risk and the maintenance in the evaluation model.</td>
</tr>
<tr>
<td>A dependability model to enhance security of cloud environment using system-level virtualization techniques [55]</td>
<td>Dependability model to evaluate the virtualised component dependability properties at the system level</td>
<td>The dependability is evaluated only at the host level</td>
</tr>
</tbody>
</table>
Chapter 3: Virtual Network Dependability Assessment Framework

Advances in virtualisation technology have enabled the development of network virtualisation that complements server virtualisation by enabling continuous workload agility irrespective of the network addressing and protocol of the underlying physical network. Despite huge benefits in both cost and accessibility, network virtualisation is susceptible to failure from a wide variety of factors. Therefore, dependability in a VNE is a significant issue that needs to be addressed before the full benefits of network virtualisation can be exploited. In this chapter, we propose a framework to estimate dependability risks in VNEs by considering variations in the virtual network configurations. The proposed framework uses the reliability block diagrams and continuous-time Markov chains to model and analyse the dependability of a virtual network. The proposed framework will be helpful to the design and construction of more dependable VNEs.

3.1. Introduction

In the past few years, server virtualisation has become the standard method for managing server infrastructure. However, virtualisation of the network is also required to realise the advantages of the server virtualisation. As a result, network virtualisation has attracted significant attention from the research community during the last few years. Network virtualisation allows multiple logical networks – each with autonomous service models, network topologies and addressing mechanisms – to run on a single shared physical network [58]. Network virtualisation also allows agility and segregation of traffic by disassociating the virtual networks from the physical network.
Although network virtualisation provides flexibility, diversity, isolation and increased system manageability, there are many technical issues that need to be addressed before fully realising the benefits that network virtualisation provides. Virtual network technology depends on the underlying physical network infrastructure such as links and nodes (e.g., routers, switches and servers) and virtualisation software. These physical network resources are prone to failure and can lead to the failure of all of the virtual networks hosted on the failed physical network infrastructure. How to efficiently allocate and schedule physical resources to the virtual network requests is a major issue that is being actively addressed. Although the focus has been on how to optimise usage of the resources of the substrate network hosting the virtual networks, recent work in this area advocates for dependability to be considered because it affects the quality of service provided by the virtualised network [59].

Dependability in a VNE is an important issue that needs to be researched in order to get the full benefits of network virtualisation. In this context, dependability modelling is an important and open problem. Although some studies have proposed evaluating dependability metrics in virtual computing systems [60], existing work tends to be preliminary and not an in-depth analysis. Dependability is the ability of the system to distribute a set of services that can be justifiably trusted [61]. Dependability also refers to the reliability of the system in providing the required functionalities [15]. Dependability can be related to disciplines such as fault tolerance, availability and reliability [62], [63]. Measurement or analytical modelling can be used for system dependability evaluation. Modelling is the preferred technique, especially when the system is very complicated or does not yet exist. Combinatorial (e.g., reliability block diagrams and fault trees) and state-based stochastic (e.g., continuous-time Markov chain and stochastic Petri net) [64, 65] models are used to represent the VNE and evaluate the dependability metrics.
In this chapter, we propose a framework to estimate dependability risks in VNEs by considering variations in the virtual network configurations. The proposed framework uses reliability block diagrams and continuous-time Markov chains for modelling and analysing the dependability level of a virtual network. The proposed framework will be helpful to the design and construction of more dependable VNEs. As a case study, we perform reliability analysis of multiple design options when single and multiple physical nodes are used to host multiple virtual networks. The contributions of the work describe in this chapter are as follows:

- The lifetime of a virtual network can be increased by mapping the virtual network onto more than substrate network components.
- We have developed a self-healing approach to overcome failures in the virtual network. We adopt a different approach to virtual network mapping onto substrate network components according to virtual network allocation and the quality of services required by the client.
- Our approach allows a virtual network provider to specify the required reliability level because the reliability of the virtual network becomes a service provided by the infrastructure.
- We have investigated the impact of substrate node failure in a virtual network, analysed the effectiveness of redundancy, and the use of fewer resources to recover the failure.
Virtual Network Dependability Assessment Framework

3.2. Models

In this section, we present the system model of interest, an overview of the problem addressed and a discussion of related work.

3.2.1. System Model

As in [66], we model the substrate network as an undirected weighted graph $G^S = (V^S, E^S, C^S, B^S)$, where $V^S$ represents the set of substrate nodes and $E^S$ is the set of substrate links. The parameter $C^S = (PC, VMF, VMR)$ represents the attributes of the substrate nodes, where $PC$ represents the available processing capacity, and $VMF$ and $VMR$ represent the MTTF and the mean time to repair (MTTR), respectively. Similarly, the parameter $B^S = (BC, EMF, EMR)$ represents the attributes of the substrate link, where $BC$ represents the available bandwidth capacity, and $EMF$ and $EMR$ represent the MTTF and the MTTR, respectively.

Virtual network requests are submitted by the system users and are modelled as $G^V = (V^V, E^V, C^V, B^V)$, where $V^V$ is the set of virtual nodes requested, $E^V$ is the set of virtual links requested, $C^V$ is the processing capacity required and $E^V$ is the bandwidth requested. Each substrate node can host a set of virtual nodes, $V^S = \{V^V_1, V^V_2, ..., V^V_n\}$, such that the total capacity of the $n$ virtual nodes is less than or equal to the substrate node processing capacity $C^S \geq \sum_{i=1}^{n} C^V_i$. Similarly, each substrate link hosts a set of virtual links $E^S = \{E^V_1, E^V_2, ..., E^V_m\}$ such that the total bandwidth capacity of the $m$ virtual links is less than or equal to the substrate link bandwidth capacity $B^S \geq \sum_{i=1}^{m} B^V_i$. If a substrate node $v^S \in V^S$ fails, all of the virtual nodes mapped onto the failed substrate node will also fail. Similarly, if a substrate link $e^S \in E^S$ fails, all of the virtual links mapped onto the $e^S$ link will also fail. In this context, dependability modelling is an important task [66].
3.2.2. Problem overview

Once the virtual nodes and virtual links have been mapped to the physical network substrate resources, the virtual network must provide services to the client in a reliable manner. The physical network components and software are prone to failure. This makes VNE dependability analysis paramount to realising the desired quality of service level.

The main problem addressed in this chapter is assessment of the dependability attributes of the virtual network infrastructure. Dependability attributes of a system refers to the reliability of the system in providing the required functionalities [15] and the ability of the system to distribute a set of services that can be justifiably trusted [61]. Dependability can be used to measure availability, reliability, safety, confidentiality, integrity and maintainability [67, 68]. Assessment of dependability attributes can be used to measure and evaluate the risks in a VNE as well as controlling and managing failures in a VNE.

Dependability evaluation in a VNE is a vital factor in the establishment of a service-level agreement between a virtual network provider, a virtual network operator and users [69]. Dependability evaluation could also be used to provide optimal resource allocation and provisioning of components at the physical network or at the virtual network provider and virtual network operator levels [69]. System dependability evaluation can be achieved using measurement or analytical modelling. Modelling is the preferred technique, especially when the system is very complicated or does not yet exist. Combinatorial (e.g., reliability block diagram and fault trees) and state-based stochastic (e.g., continuous-time Markov chain and stochastic Petri net) [64, 65] models are used to represent the VNE and dependability metrics evaluation for each model.

Dependability metrics assessment can be classified as non-state based and state based models [66]. Non-state based models deal with system availability (i.e., the system is operational or faulty) [70]. However, non-state based models have weaknesses to represent the
dynamic behaviour when the system switch from primary component to the backup component in the event of failure. Therefore, State-based models more suitable for modelling complex interactions between system components to represent dynamic behaviour by its states and event occurrences [61].

There is little research in the area of VNE dependability assessment. A cloud dependability model that uses system-level virtualisation is proposed in [55], but this work focuses on cloud security and evaluates the virtualised component dependability properties at the system level. The problem of the survivable virtual network is discussed in [32] and a heuristic that considers redundant links is discussed. However, that work does not consider dependability metrics. A framework that takes into account reliability parameters to be adopted in resource allocation for a virtual network is discussed in [54]. The drawback of this study is that it only considered one dependability metric, namely reliability. In addition, the study does not take into account a real system that consists of hardware resources and software resources, and thus, it cannot be applied for general risk assessment. A technique for computing a dependability metric in a virtual computing system based on stochastic Petri net models is discussed in [71], and a continuous-time Markov chain model for evaluating dependability metrics is discussed in [72] and [73]. In [74], a continuous-time Markov chain model is used for analysing the availability of a cluster system with multiple nodes. A hierarchical heterogeneous modelling based on reliability block diagram and continuous-time Markov chain to represent a redundant architecture and compare its availability to that of a non-redundant architecture in a Eucalyptus cloud computing environment is proposed in [52]. These works differ from our work in that they focus on the assessment of the dependability metric for virtual machines whereas we focus on VNE.

Our work is motivated and directly related to the work of [60, 66, 70], where reliability block diagrams [70] and stochastic Petri nets models were used in evaluating the dependability
of the virtual network. The work discussed in [75] only considers one simple configuration. A hybrid reliability block diagram and general stochastic Petri net model to analyse the relationship between the dependability metrics and consolidation ratio for the virtual data centre of cloud computing is presented in [60]. Our work is based on reliability block diagrams and the continuous-time Markov chain model. The important characteristic of continuous-time Markov chain models is the representation of system behaviour along the time scale. The continuous-time Markov chain model was chosen for it is greater simplicity than discrete time models. If time is discrete, the model has to consider that multiple events may occur between two consecutive time marks and search the effects of all possible combinations of these events. Continuous time scale models use appropriate probabilistic assumptions and it is possible to take only one event into consideration [57].

A continuous-time Markov chain model is more suitable for representing VNE behaviour in event of failure because the probability distributions for future developments depend only on the current state and not on the process that resulted in that state [15]. The continuous-time Markov chain model is used for two transitions to fire at exactly the same instant, while the stochastic Petri net model evolves by firing transitions one by one. Thus, the continuous-time Markov chain model is more flexible than the stochastic Petri net model because the former can fire more than one transition at a time. [60, 76, 77] proposed reliability block diagrams to assess the system reliability of cloud computing. The drawback is that dependability is assessed only at the host level and the model is too simple to describe the complex behaviour of underlying hardware as well as software components.

The proposed dependability model is realised using reliability block diagrams [70] to capture how the components of the virtual network are connected from a reliability point of view as well as to determine the reliability, availability and downtime of the system. Specifically, reliability block diagrams are used to model the various configurations (i.e.,
Virtual Network Dependability Assessment Framework

series-parallel and complex block combinations) that result in system success. However, reliability block diagrams cannot be used to represent the dynamic behaviour of the VNE in the event of a failure in the virtual network switched into spare stand-by components. To address this problem, we adopt the continuous-time Markov chain model to capture the dynamic structure of the system in the event of failure.

### 3.3. Dependability Assessment Framework

In this section, we describe the proposed dependability assessment model for VNE, where reliability block diagrams and continuous-time Markov chain models are used to represent the complex behaviour of the physical network, the virtual network and their interdependencies. Therefore, we first present the overall methodology and this is followed by discussion of the various components of the framework. In addition, scenarios will be provided to demonstrate the functionality of the framework. Generally, dependability includes several attributes that include reliability, availability, security and safety. In this chapter, we focus on two core attributes, namely reliability and availability.

Figure 3-1 shows the components of the dependability attribute assessment framework. The framework is divided into three main components: level 1, level 2 and level 3. Level 1 of the framework deal with mapping the virtual network request to the substrate infrastructure. The users specify their desired virtual network, with or without replication and the type of replication. For each request, a decision to accept or reject the request is based on the availability of resources and any constraints. For accepted requests, a mapping of the virtual network to the substrate network is performed based on the requirements of the request. The outcome of level 1 of the framework is single mapping (no redundancy), passive replication mapping or active replication mapping of the request [66]. There are many works on virtual network mapping [9, 17-21], which is outside the scope of our work.
The other two components of the virtual network dependability framework are level 2, which implements the reliability block diagram [70], and level 3, which implements the continuous-time Markov chain model. The input to the virtual network dependability framework is the virtual network mapping specification (i.e., single mapping, passive replication mapping or active replication mapping). The reliability block diagram is used to represent the mappings from level 1, and the continuous-time Markov chain model is used to capture and model the system behaviour in the event of virtual network component failures. The validation in level 3 deals with gaining confidence that a certain dependability goal (requirement) has been attained. Each of these tasks is discussed in subsequent sections. In this chapter, we assume that in both passive and active mappings, each virtual node is mapped onto primary and secondary physical infrastructure (i.e., node and link).
3.4. Reliability Block Diagram Representation of Substrate Network

A reliability block diagram is used to assess the reliability of the system and sub-system by capturing the structural relationship between system components. As noted above, the outcome of the virtual to physical mapping will be a single mapping, passive mapping or active mapping for each request [66]. In this section, we use a reliability block diagram to represent the three mappings produced by the mapping algorithm. The system’s operational state is given by its working components [61], and this work adopts series and parallel arrangements.

In a reliability block diagram representation of a mapping, rectangles represent the components and lines represent the logical relations (links). In the single mapping in figure 3-2, there is no replication of the physical resources and each virtual network maps onto a single physical network. In the passive mapping in figure 3-3, each virtual network maps onto both a primary and a secondary physical network. Only the primary virtual network will be active and the secondary virtual network will only be activated if the primary virtual network fails. Similarly, each virtual network is mapped onto both a primary and a secondary physical network in the case of the active mapping in figure 3-4. Here, both the primary virtual network and the secondary virtual network are simultaneously active. In both passive mappings and active mappings, each virtual link is mapped onto four physical links.

Figure 3-2 Single Mapping
Let $R(t)$ be the probability that the system will conform to its specification throughout duration $t$ (i.e., reliability). The failure probability $Q(t)$ is the probability that the system will not conform to its specification throughout duration $t$. Therefore, the reliability ($R_S(t)$) and the failure probability ($Q_S(t)$) for a single mapping are given in Eq. (3.1) and Eq. (3.2), respectively:

$$R_S(t) = \prod_{i=1}^{n} R_i(t)$$
$$Q_S(t) = 1 - R_S(t)$$

In the single mapping case, the system is a single-point failure. It assumes that the application requires the entire physical infrastructure to run. Any failure in physical infrastructure (e.g., router or link) will lead to failure of the entire mapping.

The reliability ($R_P(t)$) and failure probability ($Q_P(t)$) for a passive mapping are given in Eq. (3.3) and Eq. (3.4), respectively. The model consists of $m$ modules that are connected in parallel, and module $i$, for $1 \leq i \leq m$, consists of $n_i$ modules that are connected in series. Thus the reliability of a passive mapping is:
The reliability \( R_A(t) \) and failure probability \( Q_A(t) \) for an active mapping are given in Eq. (3.5) and Eq. (3.6), respectively. The model consists of \( m \) modules that are connected in series, and module \( i \), for \( 1 \leq i \leq m \), consists of \( n_i \) modules that are connected in parallel. Thus the reliability of an active mapping is:

\[
R_A(t) = \prod_{i=1}^{m} \left( 1 - \prod_{j=1}^{n_i} (1 - R_i(t)) \right) 
\]

(3.5)

\[
Q_A(t) = 1 - R_A(t) 
\]

(3.6)

3.5. Continuous-Time Markov Chain Representation of Dynamic Substrate Network

Because reliability block diagrams cannot have used to represent the dynamic behaviour of a VNE, we use the continuous-time Markov chain model to capture the dynamic behaviour of the system in the event of failure. A stochastic process with discrete events and continuous time \( T = (t \mid t \geq 0) \) is a continuous-time Markov chain if and only if:

\[
P = (N(t + h) = j \mid N(t) = i) = p_{ij}(h) \text{ for all } t, h \geq 0
\]

(3.7)

where \( P = (N(t + h) = j \mid N(t) = i) \) is the probability of the process making a transition from state \( i \) at time \( t \) to state \( j \) at time \( t + h \) for \( h \geq 0 \) is dependent only on \( j \) and the time increment \( h \).

From the above definition, two important components used to check the behaviour of a continuous-time Markov chain model are the sojourn time spent in state \( i \) (random variable) and the probability \( p_{ij} \) of transition from state \( i \) to state \( j \). We consider the time \( T \) spent in state
to be a continuous random variable with an exponentially distributed event rate parameters (e.g., failure rate ($\lambda$) or repair rate ($\mu$)) which are used as an input parameters in the Markov chain model. Thus, the continuous distribution function is:

$$F_T (t) = P(T < t) = \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

(3.8)

### 3.5.1. Simple Mapping

The continuous-time Markov chain is a three-tuple $(S, Q, L)$, where $S$ is a finite set of states, $Q$ is a transition rate matrix between the states and $L$ is the labelling function that assigns reward for each state.

![Figure 3-5 Single Mapping Model](image)

Figure 3-5 illustrates a single mapping case. State $S0$ indicates that all components are working. States $S1$, $S2$ and $S3$ represent a failure in a physical router, a failure in a physical link and a failure in a virtual machine monitor, respectively. These failures lead to failure of the system. At time 0, the system is in the working state. The system goes into a failed state as soon as a component fails. The labelling function is assigned to the state to represent the number of virtual networks hosted by the physical network. For example, 1 is assigned to the states for one virtual network hosted by substrate network, and 2 is assigned to the states for
two virtual networks hosted by the substrate network. The labelling function assigns 1 to states $S1$, $S2$ and $S3$, and assigns 3 to state $S0$. From the above model, we can compute the steady-state unavailability of the system by computing the steady-state unavailability of all reward states:

$$UA_s = \sum_{s \in S} P_s L_s$$  \hspace{1cm} (3.9)

where $P_s$ is the steady-state probability of being in state $S$, and $L_s$ is the labelling function assigned to state $S$.

In addition, we can compute the probability of any component failure in the system. The following expression can be used to calculate the unavailability, the probability of physical router fails and the availability in the system, respectively:

$$UA = (P\{S1\} + P\{S2\} + P\{S3\})$$

$$ProbRouterFailur = (R\{S1\} * P\{S1\})$$

$$A = 1 - (P\{S0\})$$

### 3.5.2. Passive Mapping

Figure 3-6 illustrates a passive mapping case. We assume the virtual network is mapped onto two physical routers, and when the primary physical router fails, the stand-by physical router begins working. The state $VN_1H_{1U}H_{2I}$ indicates that the primary host $H_{1U}$ is up, the stand-by host $H_{2I}$ is idle and the virtual network $VN_1$ is hosted by both hosts. The state $VN_1H_{1D}H_{2I}$ indicates that the primary host $H_{1D}$ has failed with the rate $\lambda$. The state $VN_1H_{1D}H_{2R}$ indicates that the primary failure is detected with rate $\delta$, and then the stand-by host restarted. Failure is detected using the failure detection mechanisms (e.g., heart beat mechanism every 30 seconds) [78]. The state $VN_1H_{1D}H_{2U}$ indicates that the $VN_1$ started on the stand-by host $H_{2U}$ which takes the mean time $r_e = 5$ min. This is called a virtual machine high availability service in VMware [79].
When a system in state $VN_1H_1D H_2U$, it may go to state $VN_1H_1U H_2U$ after repairing the primary host $H_1D$ with repair rate $\mu$. The state $VN_1H_1D H_2D$ indicates that the stand-by host $H_2D$ has failed with rate $\lambda$ and then the virtual network failed. At time 0, the primary active component (either the physical router or the link) is in the working state while the secondary component is in the stand-by state. The system goes into a failed state as soon as a failure occurs in both components. From the above model, the MTTF for the system increases by combining the MTTF of primary components and the redundancy active component:

$$T_s = T_1 + T_2$$

$$R_s(t) = R_1(t) + \int_0^1 f_1(x) R_2(t - x)dx$$

Because the probability ($P(T_s > t)$) of the union of these events is the reliability function of the system ($R_s(t)$), the reliability of the system is increased, as illustrated in Eq. (3.11). The labelling function assigns 1 to states ($VN_1H_1U H_2U, VN_1H_1D H_2D, VN_1H_1D H_2R$ and $VN_1H_1D H_2U$) and 0 to state $VN_1H_1D H_2D$. The following expressions are used to calculate the availability and the unavailability of the system, respectively.

$$A = (P\{VN_1H_1U H_2U\} + P\{VN_1H_1D H_2D\}) - (P\{VN_1H_1D H_2D\})$$

$$UA = 1 - (P\{VN_1H_1U H_2U\} + P\{VN_1H_1D H_2D\}) - (P\{VN_1H_1D H_2D\})$$
3.5.3. Active Mapping

Figure 3-7 illustrates an active mapping case. The state \( UUUU \) indicates that all components are working (the state \( UUUU \) represents the primary physical router, active redundancy physical router, primary physical link and active redundancy physical link, respectively). State \( UDUUU \) indicates that the primary physical router failed at time \( x_1 \) \((0 \leq x_1 < t)\) and the active redundancy physical router worked properly for a period longer than \( t - x_1 \). The state \( UUU_DU \) indicates that the primary physical link failed at time \( x_1 \) \((0 \leq x_1 < t)\) and the active redundancy physical link worked properly for a period longer than \( t - x_1 \). The state \( DU_DUU \) indicates that the primary physical router failed at time \( x_1 \) \((0 \leq x_1 < t)\) and the active redundancy physical link failed after a period of \( x_2 \) \((0 \leq x_2 < t - x_1)\), after which the system failed.

Similarly, state \( UUU_DU \) indicates that the primary physical link failed at time \( x_1 \) \((0 \leq x_1 < t)\) and the active redundancy physical link failed after a period of \( x_2 \) \((0 \leq x_2 < t - x_1)\), after which the system failed. The system goes into a failed state as soon as a failure
occurs in both components. The labelling function assigns 1 to states $UUUU, U_DUUU$ and $UUUU_D$, and assigns a 0 to states $U_DUU$ and $UUUU_D$. 

In an active mapping case, the primary component and the active redundancy component are both active at time $t_0$. When the primary component fails at time $x$, the active redundancy component survives beyond time point $t$, where $0 \leq x < t$. The system failed when the active redundancy component fails. From the above model, we can compute the unavailability and the availability in the system, respectively using the following expressions:

$$UA = 1 - (P\{UUUU\} + P\{U_DUU\} + P\{UUUU_D\})$$

$$A = 1 - (P\{U_DUU\} + P\{UUUU_D\})$$
3.6. Performance Analysis

In this section, we evaluate the performance of the proposed framework and compare it with the dependability model discussed in [66].

3.6.1. Experimental Set-up

We have constructed reliability block diagrams and continuous-time Markov chain models using the Mercury/Astro environment [80]. Reliability block diagrams and continuous-time Markov chain models are used for evaluating dependability metrics for system and subsystem components. To construct the network topology, we used the embedding techniques presented in [17]. GT-ITM tools [81] were used to generate a substrate network with 50 nodes randomly connected with probability 0.5. The CPU capacity for each node and the bandwidth capacity for each link were real numbers and uniformly distributed between 50 and 100. The virtual network requests arrived in a Poisson process with a mean rate of four virtual networks per 100 time units. The number of virtual nodes was randomly distributed for each virtual network request, following similar set-up in the previous work [66], the number of virtual networks requests were 800 over a period of 50,000 hours and each virtual network request had an exponentially distributed lifetime of 1,000 time units.

3.6.2. Results and Discussion

In this section, we describe the proposed approach for evaluating the virtual networks generated by the mapping algorithm presented in [17]. We used the approach discussed in [82], and the objective of the algorithm is to provide different virtual network mappings into the substrate network satisfying CPU, bandwidth and cost constraints. The exponential distribution for the MTTF and MTTR of the hardware and software components is adopted for each allocation for
analysing the dependability metrics. Table 3-1 presents the MTTF and MTTR for each component based on [75]. In our study, we used the algorithm for resource allocation and evaluated the reliability and availability dependability metrics.

Table 3-1 Component MTTF and MTTR

<table>
<thead>
<tr>
<th>Node</th>
<th>MTTF (h)</th>
<th>MTTR (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Switch/Router</td>
<td>320,000</td>
<td>1</td>
</tr>
<tr>
<td>Virtual Machine Monitor</td>
<td>2,880</td>
<td>2</td>
</tr>
<tr>
<td>Network Interface Card</td>
<td>6,200,000</td>
<td>1</td>
</tr>
<tr>
<td>CPU</td>
<td>2,500,000</td>
<td>1</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>200,000</td>
<td>1</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>1,440</td>
<td>2</td>
</tr>
<tr>
<td>Memory</td>
<td>480,000</td>
<td>1</td>
</tr>
<tr>
<td>Optical Link</td>
<td>19,996</td>
<td>12</td>
</tr>
</tbody>
</table>

To evaluate reliability and availability for virtual network allocation, we make two assumptions: the virtual network is allocated into physical network components with and without common mode failure. The former is modelled as a continuous-time Markov chain and the latter is modelled as a reliability block diagram model. The reliability block diagram is used to evaluate the reliability metric, while the continuous-time Markov chain model is used to evaluate the availability metric for virtual network allocation. In addition, we assume that the MTTF for each physical network component decreases when the number of virtual networks hosted by physical network increases.

Figure 3-8 illustrates the reliability values for different allocations of a virtual network into physical network components with the assumption of independence of failure. The reliability of the simple mapping decreased dramatically because the components were connected in series. The failure rate of a series system is equal to the sum of the component failure rates, \( \lambda_s = \sum_{i=1}^{n} \lambda_i \). The failure rate of the system is higher than the failure rate of the
component when the system size is large. The reliability of the system decreased because the MTTF for the series system is equal to

\[ MTTF_s = \frac{1}{\sum_{i=1}^{n} \lambda_i}. \]

The reliability of the passive mapping increased because the physical network components were connected in parallel at the system level. This means that the reliability of the parallel system

\[ R_p = 1 - \left(1 - e^{-\lambda t}\right)^n \]

increased because the MTTF for the system is increased. The reliability of the active mapping increased significantly because the physical network components were connected in parallel. From the above analysis, reliability is an important factor for virtual network allocation to physical network components. High reliability mapping is achieved by choosing components with high MTTF for virtual network allocation. The reliability of active mapping is higher than the reliability of passive mapping, because in active mapping, the MTTF for the system increases by combining the MTTF of primary components and the secondary active. While in passive mapping, the MTTF is equal to the MTTF for only active component since the secondary component is idle during the normal operation of primary component.
Because the physical network components consist of hardware components (i.e., routers, switches and fibre optic cable) used by the virtual network, failure in the physical network will cause all of the hosted virtual networks to fail. For example, if the router fails (e.g., CPU or memory), the virtual node will fail. Similarly, if the fibre optic cable fails, the virtual link will fail and the system becomes unavailable. The proposed dependability model allows the evaluation of fault-tolerance techniques by adopting mapping with and without redundancy.
The mapping with redundancy improved the reliability and the availability of the system in the event of a failure. The availability results illustrated in figure 3-9 shows that passive mapping and active mapping for virtual network allocation achieved higher availability than simple mapping because redundancy was used with the former and no redundancy was used with the latter. Passive mapping achieved higher performance than active mapping because the stand-by redundancy in the passive mapping started when the primary components failed, while in the active mapping, the redundancy ran simultaneously with the primary components. The availability in passive mapping is higher than in active mapping, this is because in passive mapping only the primary virtual network active and the secondary virtual network is idle and it will be activated if the primary virtual network fails. Thus, in passive mapping the virtual network has a spare component with high availability. While in active mapping, both components (primary and secondary) run simultaneously, therefore there are a chance for both components at the same time.

Figure 3-9 Availability Results for Virtual Network Allocation
The proposed dependability model achieved more reliable results in measuring dependability metrics than the dependability model in [66]. The dependability evaluation achieved in previous work [66] is illustrated in figure 3-10 shows that hot stand-by achieved higher availability than the cold stand-by in measuring reliability. In the hot stand-by model (in our model equivalent to active mapping), the primary and secondary components run simultaneously and may fail at the same time. In the cold stand-by model (in our model equivalent to passive mapping), the stand-by component starts after the primary component fails and the lifetime of the system is increased significantly. Thus, the cold stand-by should achieved higher availability than the hot stand-by model as we show in our results in figure 3-9.

Figure 3-10 Availability Results
Evaluating the availability is used to assess quality of service according to virtual network allocation. For example, we achieved very high availability by adopting different redundancy techniques. The results in table 3-2 confirm an enhanced dependability of the proposed redundancy system that is verified by increasing the availability from two to five times. In addition, the annual downtime is decreased from 11.30 hours to only 0.028 minutes.

<table>
<thead>
<tr>
<th>Model</th>
<th>Availability</th>
<th>Unavailability</th>
<th>Annual Downtime (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Mapping</td>
<td>0.998</td>
<td>0.00129</td>
<td>11.30</td>
</tr>
<tr>
<td>Passive Mapping</td>
<td>0.999996</td>
<td>0.000003267</td>
<td>0.028</td>
</tr>
<tr>
<td>Active Mapping</td>
<td>0.9997</td>
<td>0.0002</td>
<td>1.75</td>
</tr>
</tbody>
</table>

### 3.7. Chapter Summary

In this chapter, we presented a modelling method and evaluation techniques for computing dependability metrics of a VNE. Analytical modelling is preferred over measurement techniques for evaluation of system dependability when the system is very complicated or might not yet exist. The dependability of a system is delivering a set of trustable services without failure. A failure occurs in the system when the system fails to deliver its identified functionality. A fault in the system is defined as the failure of a component of the system. Therefore, we used an analytical modelling technique for evaluation of the faults in the components, subsystem and the system as a failure or non-failure. In our approach, we used the dependability metrics to evaluate the system reliability and availability. Reliability is the probability that the system is working up to time $t$, while availability is the probability that the system is working at time $t$. MTTF and MTTR of the VNE components were adopted for analysing the reliability and availability metrics, respectively. Dependability metrics were
evaluated by using reliability block diagrams and continuous-time Markov chain models. Reliability block diagrams were used to represent different mappings of the virtual network onto the substrate network and to assess the reliability of the virtual infrastructure components. The virtual network was mapped onto the substrate network without redundancy as simple mapping or with redundancy as passive or active mapping. In passive mapping, the backup mapping redundancy is idle during the operation of primary mapping, and the backup is activating when the primary mapping fails. In active mapping, the backup and primary mapping run simultaneously. A continuous-time Markov chain was used to model the complicated interaction between the VNE components. Continuous-time Markov chain models capture the dynamic behaviour in event of failure in hardware and software components of the VNE. In addition, we used continuous-time Markov chain models to study the performance in event of failure in VNE and compared the availability of the VNE with and without stand-by redundancy. The proposed framework was used for evaluating the reliability and availability according to virtual network allocation and the quality of services with the client. In addition, the framework was used to assess the optimal reliability design for the virtual network allocation in a physical network. The experimental results show that our proposed modelling achieved very high performance in measuring dependability metrics. Chapter 4 will concentrate on the detection of failure in virtual infrastructure components.
Chapter 4: Failure Detection in Virtual Network Infrastructure

In this chapter, we use a detection mechanism based on a conservative time-synchronisation algorithm and message passing interface to detect normal and anomalous behaviours in a VNE. A substrate network and its software are prone to failure, which leads to failure of all the virtual resources hosted by that substrate network and the need to remap the virtual network to different substrate network resources. Detecting failure in a VNE is an important issue to overcome the failure in a VNE and improve a virtual network reliability.

4.1. Introduction

A virtual network is a subset of the underlying substrate network resources. A combination of virtual nodes and virtual links is created on top of a substrate network by virtualising the substrate node and link resources. A virtual network is mapped onto substrate network resources using existing mapping proposals [17, 19, 20, 83]. The virtual topology is created by using virtual links to connect multiple virtual nodes [22, 84, 85]. In addition, multiple virtual topologies can be created and co-hosted on the same substrate network, each with specific application naming, topology routing and resource management mechanisms [22]. Because each virtual network is instantiated and managed independently, the virtual networks can employ communication protocols that are tailored to their service environment [2]. These features lead to greater service provision flexibility than is currently available on the Internet [1, 26, 86].

The virtual network embedding problem has been addressed by many researchers [10, 17, 87] who have studied efficient virtual network embedding into physical network without consideration of failure in the physical resources. A failure in either a physical node, a physical...
link or both can affect the many virtual networks that run on a shared substrate network with limited network resources such as bandwidth and CPU capacity.

A virtual network requires resolving many challenges, specifically those related to reliability. Virtual network reliability refers to the ability of the overall network to provide communication in the event of a failure in the physical network. Virtual network reliability is an important and open research question. In this chapter, we develop a mechanism to detect and overcome failure in a virtual network to improve virtual network reliability.

Failure is detected by a fault detection mechanism in the event of the complete failure of a virtual infrastructure component (fail-stop). The failure is detected using a message passing interface that probes connections between point-to-point nodes by message exchange. In addition, conservative time-synchronisation algorithms are used to determine the time-out before considering that a failure event has occurred in a VNE. The contributions of the work described in this chapter are:

- We propose a fault detection mechanism that detects when a component in a VNE has failed and notifies the system about the failure.
- The failure detection system can cope with a large-scale virtual network and prevent overloading the VNE by reducing the number of messages for failure detection.
- Physical network components do not change as rapidly as the virtual networks (in which virtual machines can appear or disappear very frequently). Therefore, we design a failure detection mechanism that can dynamically respond to virtual network resources allocation, is time-efficient in detection of the failure and can run independently without the need for reconfiguration.
- We evaluate the accuracy and completeness of the failure detection system during run time and off-time by running the experimental data through the support vector machine (SVM) classifier and comparing the results with those of existing approaches.
4.2. Problem Overview

In this section, we describe the failure problem in a VNE. We model the substrate network as an undirected weighted graph $G^S = (V^S, E^S)$, where $V^S$ represents the set of substrate nodes and $E^S$ represents the set of substrate links. Similarly, we model the virtual network as an undirected weighted graph $G^V = (V^V, E^V)$, where $V^V$ represents the set of virtual nodes requested and $E^V$ represents the set of virtual links requested, as shown in figure 4-1. Each substrate node can host a set of virtual nodes, $V^S = \{V^V_1, V^V_2, \ldots, V^V_n\}$, and each substrate link can host a set of virtual links, $E^S = \{E^V_1, E^V_2, \ldots, E^V_m\}$, as shown in figure 4-2. Failure in a physical node will affect all of the virtual nodes mapped onto the failed physical node. In addition, failure in a physical link or physical path will affect all of the virtual links mapped onto the failed physical link or physical path. For example, if a physical node $v^S \in V^S$ fails, then all the virtual nodes mapped onto the failed physical node will fail. Similarly, if a physical link or physical path $e^S \in E^S$ fails, then all of the virtual links mapped onto the failed link will fail. A single substrate entity failure will affect all of the virtual entities that are mapped upon it.

Failure in a virtual network will decrease the virtual network’s reliability and increase the operational costs. Reliability can decrease due to numerous types of interruption, such as fibre cut, maintenance and misconfiguration. Operational costs increase due to the need for reconfiguration of the failed virtual network and the infrastructure providers may suffer economic penalty because of the breach of the level of service required by service providers [88].

Most previous studies recover a failure by allocating more resources as a backup. One technique to improve the reliability of a VNE when a failure occurs in either a substrate node or a link is to allocate a backup at a different geographical location with redundant links to be
provisioned to the virtual network after the failure occurs [89], [21]. Some researchers have introduced an approaches to protect against any potential single link failure. For example, one approach is to use two backup resources allocations, the first is allocated on arrival of the virtual network request, and the second is a pre-allocated backup resource during configuration and before any virtual network request arrives [31]. Another approach is introduced before a failure occurs in a single link failure by separating the bandwidth of a substrate link into two shares: the first share is active primary for transport primary flows in the normal operation and the second share is inactive backup used for carrying backup flows in event of failed primary flow [32]. The backup path is used in the event of a single substrate link failure [33], and a migration technique is used to allocate the virtual node into another substrate network [36, 46]. To protect a virtual link against a single physical link failure, multiple substrate paths with flexible path splitting ratios are used to map the virtual link [19]. In addition, some researchers have introduced approaches to protect against node failure. For example, one approach introduces a mechanism to migrate the virtual node onto a backup physical node [43, 45]. The drawback of the abovementioned proposals are that they are inefficient because the resources are wasted until a failure occurs in the VNE. In addition, reinstallation after a failure in a VNE is not a reliable method to recover data that has been lost.

Figure 4-1 Virtual Network Requests
We developed an approach to detect failure in virtual infrastructure components using an efficient detection mechanism solution. This work makes a significant contribution to the network virtualisation knowledge base in general, and to reliability and resilience in particular. The study investigated and developed a mechanism to detect virtual network failures and avoid service interruptions after a failure occurs. An existing failure detection system was proposed in [90-92] for a large computer network centre, but their proposal is designed for fixed or slowly changing infrastructure, such as routers, switches and servers. Another study [93] focused on the workload model and failure correlations in cloud computing. They proposed a framework for monitoring a cloud-based system, collecting unlabelled data and using an ensemble of Bayesian models as an unsupervised method for failure detection based on the history of the
collected data [94]. For detection of failure in a virtual network, previous work has been based on the traffic load, where the traffic rate is detected on the user link and adjusts the allocated bandwidth based on the forecast from traffic history [95]. The drawback of [95] is that the method is dependent only on measuring traffic load for failure detection, but the traffic load could be increased on a specific link due to heavy traffic. A management framework for detection in a the virtual network uses a probe to collect data represents an interesting feature that can be used to measure data to detect abnormal behaviour of a virtual network [96]. In the proposal in [96], the failure detection system is controlled by the hypervisor, and when the hypervisor fails, the failure detection system fails. A proposed prepared adaptive virtual network embedding framework detects node or link failure in a VNE using multi-agents that are integrated into the substrate nodes. The agents detect failure through keep-alive messages that are exchanged periodically between nodes that belong to the same cluster [48]. The drawback of the work proposed in [48] is that it consumes a lot of traffic because the detection mechanism send messages continuously, even when there is no failure. Our approach is different from existing approaches because it introduced time-efficient failure detection and reduces the number of messages for failure detection that can cope with a large-scale virtual network.

4.3. System Model

In this section, we present the system model for failure detection in a VNE. We adopted an efficient failure detection technique that takes into consideration the following issues:

- **Scalability** – the detection system should be designed to work in a large-scale virtual network, and it must quickly detect a failure.
- **Adaptation** – the detection system should be adapted to very high load traffic and avoid overloading the network by reducing the number of messages for failure detection.
• Autonomic – the detection system should keep running and detect the virtual network behaviour independently and without configuration.

• Flexibility – The detection system should correctly detect new virtual machines created in the system and the expired virtual machines.

### 4.3.1. VNE Topology

To solve the previously mentioned issues, such as scalability, adaptation, autonomic and flexibility, we designed a hierarchical topology to represent a VNE. Hierarchical topology is very scalable and can be used in grouping many nodes into one cluster (clustering reduces the number of links needed to connect the virtual nodes). These characteristics of the hierarchical network topology achieve high performance in delivering messages between virtual nodes and increase virtual network reliability [97]. As illustrated in figure 4-3, the autonomous system-level and the router-level are used to represent the physical network and the virtual network, respectively. For each node at the autonomous system-level, there is a router-level to represent the virtual nodes. The virtual nodes interconnect in the router-level topologies according to the connectivity of as the autonomous system-level topology. For example, if we have two nodes \( S_i, S_j \) in as the autonomous system-level and \( (S_i, S_j) \) represents a link in as the autonomous system-level, to connect two nodes in the router-level, we choose a node \( V_u \) in the router-level that is associated with the autonomous system-level node \( S_i \), and we choose a node \( V_m \) in the router-level that is associated with the autonomous system-level node \( S_j \).
To reduce the number of messages between nodes, we designed an efficient failure detection system by partitioning the network topology into groups of nodes and placing each group into a logical process, as illustrated in figure 4-4. Each logical process is assigned a unique number to represent the system identifier, and each logical process has its own events to be processed. During configuration of a network topology, each of the nodes is assigned a number to denote its logical process, and in cases where a link is created between two nodes in different logical process, a remote point-to-point channel is created between them. A message
Failure Detection in Virtual Network Infrastructure

The passing interface is implemented to exchange messages between nodes in same logical process or between nodes in different logical processes by creating remote point-to-point channels.

4.3.2. Fault Detection Model

The proposed model for failure detection in a VNE is based on the multi-agents system from the artificial intelligence field [98]. A message passing interface [99] is used to probe connections between point-to-point nodes by exchanging a time-stamped message. We have chosen a conservative time-synchronisation algorithm [100] to determine out-of-order time-stamp messages in the event of failure in a VNE. The conservative time-synchronisation algorithm determines the threshold value for failure by using a predetermined value called a lookahead value. The lookahead value is the minimum time that must pass before node $V_u$ considers a fail in node $V_m$ or a link fail between $V_u$ and $V_m$ (fail-stop). The lowest bound time-stamp (LBTS) is determined using a null message algorithm [101, 102]. The LBTS on all

![Figure 4-4 Partitioned Network Topology with Two LPs](image-url)
possible events that it may receive is used as the lookahead value. The null message algorithm begins by searching the nodes in each logical process to find external links to remote logical processes. It then groups all of the links into bundles according to which logical process is connected. Next, it determines the minimum propagation delay value for each bundle, which becomes the lookahead between the two logical processes. In the conservative time-synchronisation algorithm, each logical process has to determine whether an event is a failure or a non-failure. A failure event occurs when the logical process receives events from other logical processes with time-stamps that are less than the event being considered. A non-failure event occurs when the logical process receives events with time-stamps in order from other logical processes.

For example, as illustrated in figure 4-5 we assume that node $V_0$ in $LP_0$ is connected to $V_1$ in $LP_1$. The message passing interface sends and receives messages between a remote point-to-point link connecting two logical processes. In addition, we assume the communication between two nodes starts at 3 times and the propagation delay for the link is 10 times (i.e., the lowest bound time-stamp for the link is 10 times). We also assume normal behaviour in the VNE so that the sequence of the time-stamp messages occurs in order. The sequence of the time-stamp messages occurs as follows: the first time-stamp message is $(3, 1)$, where the first component of the message is time (i.e., 3 from the source $V_0$) and the second component (1) is the is the echo-received message by the source node $V_0$. Then the message departs $LP_0$ to $LP_1$ so that the time-stamp message at $LP_1$ is $(13, 1)$, where the first component (13) is the sum of arrival time (3) and delay time (10). Finally, the message arrives at the sink $V_1$ and the time-stamp message is $(23, 1)$, where the first component (23) is the sum of the arrival time (13) and the delay time (10). Thus, from the sequence of the time-stamp messages we can determine when the failure occurs in the VNE.
4.3.3. Data Collection Model

We used the Network Simulator 3 [103] as a data collection framework to model different failure scenarios in virtual networks and extract interesting data to study the behaviour of a VNE in the event of a failure. The data collector is based on the concept of producer (trace source) and consumer (trace sink). The producer and consumer concept is very scalable because the producer is decoupled from the consumer (i.e., space, time and synchronisation decoupling) [104]. The producer is an entity used to generate data for system management, signal an interesting event that happened in the system and provide access to the consumer. The consumer is an entity that reads the source data generated by the producer. The trace source may be connected to one or multiple trace sinks, and when an interesting state change occurs in the system, it will use signal event to pass the changed state to the trace sink.

To connect the producer and the consumer, we used the Network Simulator 3 callback feature, which allows the two modules to communicate through function calls. A trace source is a callback to which several functions may be registered. When a trace sink is interested in
receiving trace events, it adds a callback to the list of callbacks stored by the trace source. When an event of interest occurs, the trace source invokes all of its callbacks in turn and provides the required parameters (such as time-stamp messages received) to the trace sinks. The trace source keeps track of all registered processes and records whenever a time-stamp message arrives. Because the trace source knows the frequency at which time-stamp messages are generated by the registered processes, it can infer missing time-stamp messages. The trace source can create callbacks for a failure event such as a missing time-stamp message or an out-of-order time-stamp message. The conservative time-synchronisation algorithm uses a lookahead threshold value to determine component failure based on how late the time-stamp message is. In summary, the function of the data collection is limited to keeping track of time-stamp messages and invoking callbacks between trace sources and trace sinks.

4.3.4. Metrics Used

We used the following metrics to evaluate the fault detection model:

- The true positive rate, or recall, is the proportion of positive cases that were correctly identified, and is calculated using the following equation:

\[ TPR = \text{Recall} = \frac{TP}{TP + FN} \]  \hspace{1cm} (4.1)

where \( TP \) denotes true positive instances and \( FN \) denotes false negative instances.

- The false positive rate is the proportion of negative cases that were incorrectly classified as positive, and is calculated using the following equation:

\[ FPR = \frac{FP}{FP + TN} \]  \hspace{1cm} (4.2)

where \( TN \) denotes true negative instances and \( FP \) denotes false positive instances.

- The receiver operating characteristic (ROC) curve is used for analysing the performance of a classifier system and is created by plotting the true positive rate on the \( y \)-axis against the false positive rate on the \( x \)-axis. The best prediction method
would yield a value in the upper left corner of the receiver operating characteristic curve.

4.4. Performance Analysis

In this section, we evaluate the performance of the proposed framework and compare it with the failure detection model discussed in [48] and [94].

4.4.1. Experimental Set-up

In our study we used Network Simulator 3 (NS-3) to model different failure scenarios in virtual networks and extract interesting data to study the behaviour of a VNE in the event of a failure [103]. NS 3 is a discrete-event network simulator platform that can be used for failure detection, to analyse network features and to extract interesting data to detect failure in a VNE [105]. The simulations were run in an Ubuntu 14.04.2 LTS Virtual Machine with 8 GB RAM and a 2.60 GHz CPU. Boston University Representative Internet Topology gEnerator (BRITE)[106] was used to generate a hierarchical topology to represent the VNE. BRITE is an ideal topology generator to represent the substrate network and the virtual network topologies using hierarchical structure as illustrated in figure 4-3. In addition, BRITE is very efficient and flexible such that can be used to generate very large scale topology (e.g. number of nodes > 100,000 in VNE) with reasonable CPU and memory consumption. Moreover, widely used simulators such as NS-3 can process the generated topologies by BRITE.

4.4.2. Results and Discussion

We measured two properties, cost and accuracy, for evaluation of the failure detection system. Reducing cost requires minimising the overhead in the network traffic by reducing the number
of messages generated by the failure detection system. Accuracy measures how quickly the failure is reported with a low false positive rate by failure detection system.

### 4.4.2.1. Accuracy

For detection of the behaviour of a virtual network in the event of failure in a VNE, we modelled failure as a fail-stop model (i.e. the virtual infrastructure components stop completely from normal operation). Failure was injected into the virtual infrastructure components with the failure rates ($\lambda$) of 0.001 and 0.003. The results show that failure detection system achieved high accuracy of the detection of processes because the number of failures detected increased when the failure rate increases from 0.001 to 0.003. Figure 4-6 shows that the number of failures detected by the failure detection system with a failure rate of 0.003 was higher than the number of failures detected by the failure detection system with a failure rate of 0.001 because the MTTF for virtual networks components are increased (i.e. $MTTF = 1/\lambda$), when the MTTF increased, the lifetime of the component increases significantly, and thus the number of failure occurrences decreases significantly.
The accuracy of our failure detection system was investigated to avoid a false positive failures and a false-negative failure. To avoid false positive failures and false negative failures, a lookahead value should be carefully chosen. Therefore, we investigated several look-ahead values to find the accuracy in failure detection. Accuracy was measured by using the look-ahead values of 2 ms, 1 ms, 0.01 ms and 0.001 ms with the failure rate $\lambda = 0.001$ and 500 virtual nodes. The simulation was run 10 times for 60 minutes with the different look-ahead values. From our experiment, we found the highest accuracy of 95.5% was achieved when the look-ahead value was 2 ms, as shown in figure 4-7.
To measure the true positive rate and the false negative rate we ran our failure detection system with a look-ahead value of 2 ms with different numbers of nodes (1,200 nodes, 1,000 nodes, 800 nodes, 600 nodes, 400 nodes and 200 nodes). Table 4-1 shows that our approach achieved high accuracy with a low false negative rate.

<table>
<thead>
<tr>
<th>Number of Virtual Network Nodes</th>
<th>True Positive Rate</th>
<th>False Negative Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,200</td>
<td>0.955</td>
<td>0.045</td>
</tr>
<tr>
<td>1,000</td>
<td>0.948</td>
<td>0.052</td>
</tr>
<tr>
<td>800</td>
<td>0.964</td>
<td>0.036</td>
</tr>
<tr>
<td>600</td>
<td>0.946</td>
<td>0.054</td>
</tr>
<tr>
<td>400</td>
<td>0.947</td>
<td>0.053</td>
</tr>
<tr>
<td>200</td>
<td>0.948</td>
<td>0.052</td>
</tr>
</tbody>
</table>
4.4.2.2. Average Failure Detection Time

The failure detection time evaluation was based on the time required to detect a failure between the nodes with varying numbers of clusters in the VNE topology. Figure 4-8 shows that the average detection time decreases significantly when the number of clusters is increased because failure detection is restricted to a few substrate nodes. The average failure detection time decreases from 0.9574 seconds for one cluster to 0.04562 seconds for five clusters. We found that our approach achieved high performance in failure detection time when the number of clusters in the VNE topology was increased from one cluster to five.

![Average Detection Time Using Different Numbers of Clusters](image)

Figure 4-8 Average Failure Detection Time Using Different Numbers of Clusters
4.4.2.3. **Average Number of Messages Exchanged**

The number of messages exchanged for failure detection decreases when the number of clusters increases, as shown in figure 4-9. The number of messages exchanged in failure detection failure is very small when the number of clusters increases because a message is exchanged among few substrate nodes in the event of failure. The number of messages decreases from 68 messages for one cluster to seven messages for five clusters. We found that our approach achieved high performance because the overhead from messages exchanges drops significantly with an increased number of clusters.

![Figure 4-9 Number of Messages Exchanged with Different Numbers of Clusters](image-url)
Figure 4-10 shows a comparison of the number of messages exchanged for failure detection using a previous approach [48] and using our conservative time-synchronisation algorithm. Our approach requires the exchange of very small number of messages for failure detection, while the previous approach requires the exchange of a large number of messages. For example, with 100 nodes for failure detection, our approach exchanges 106 messages, while the previous approach exchanges 4,200 messages. This discrepancy is because our failure detection system is based on a producer and consumer approach. When there is an interesting state change in the system, a message is exchanged whereby the producer passes the changed state to the consumer. Conversely, the previous approach continuously exchanges messages between all nodes in the virtual network, even without a failure event.

![Number of Messages Exchange for Failure Detection](image)

Figure 4-10 Number of Messages Exchanged with Different Number of Nodes
4.4.3. SVM Model Detection Results

We used a SVMLIB [107] in Weka [108] to build a SVM model for detection of failure and non-failure in the components of the VNE. We collected the dataset to represent the failure and non-failure occurrences in a VNE from our failure detection system. We then split the dataset into 70% as the training dataset and 30% as the testing dataset. The training dataset was used to train the SVM model to classify the features that indicate whether a given error sequence is a failure-prone or not. The testing dataset was used to evaluate the generalisation performance of the SVM model.

The aim of constructing the SVM model was to evaluate the accuracy of our failure detection system. The SVM algorithm is chosen because it can be used for solving a complex problems in classifying a failure and non-failure in VNE, it employs very sophisticated mathematical principles to avoid over-fitting, and gives greater experimental results compared with other models [120]. SVM can be assumed as a technique of data compression because it objectives to find the subset of training data points, which present the whole information held in the dataset. In reality, support vectors are those points that summarize the information of the training dataset and allow detection test dataset in an efficient model [120].

The training dataset was collected from a failure detection system and comprised 11,670 instances that represent a failure and non-failure in a VNE. The results of training the SVM model show that 11,112 instances were classified correctly with true positive rate of 96.04% and 458 instances were misclassified with a false positive rate of 3.96%. We then validated the performance of the SVM model using the testing dataset with splits of 10%, 30%, 50%, 70% and 90%. For example, for the first experiment with 90% training data and 10% testing data, we calculated the average correct and standard deviation from the 10 runs and then used the average correct and standard deviation to calculate the success rate. Table 4-2 shows that the
SVM model performed very well in classifying the failure and non-failure because the success rates were between 90% and 100%.

### Table 4-2 Success Rate Percentage in SVR Model

<table>
<thead>
<tr>
<th>% Training Data</th>
<th>% Testing Data</th>
<th>Correct Average</th>
<th>Incorrect Average</th>
<th>SD</th>
<th>% Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>10</td>
<td>6,845.7</td>
<td>760.3</td>
<td>0.6403</td>
<td>90.00</td>
</tr>
<tr>
<td>70</td>
<td>30</td>
<td>5,324.3</td>
<td>2,281.7</td>
<td>0.6403</td>
<td>100.00</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>3,803.2</td>
<td>3,802.8</td>
<td>0.7483</td>
<td>100.00</td>
</tr>
<tr>
<td>30</td>
<td>70</td>
<td>2,281.9</td>
<td>5,324.1</td>
<td>0.9434</td>
<td>90.00</td>
</tr>
<tr>
<td>10</td>
<td>90</td>
<td>760.7</td>
<td>6,845.3</td>
<td>0.6403</td>
<td>90.00</td>
</tr>
</tbody>
</table>

To evaluate the performance of the SVM model, we ran a ten-fold cross-validation for 10% of the testing dataset, and then calculated the true positive rate and false positive rate from each run. The true positive rates and false positive rates were used to plot receiver operating characteristic curves of the detection accuracy with different threshold values. The results of the SVM model compare with the ensemble Bayesian and decision trees models prepared in [94]. The results in figure 4-11 and figure 4-12 show that the SVM model outperforms ensemble Bayesian and decision trees models, respectively, because the optimal performance for each classifier is at the top left of the receiver operating characteristic curve (i.e., with a high true positive rate and a low false positive rate). The results in figure 4-11 and figure 4-12 show that the SVM model achieved a high true positive rate (94%) and a low false positive rate (0.2%) compares with both ensemble Bayesian and decision trees models.
Figure 4-11 Receiver Operating Characteristic Curve Results for SVM and Naïve Bayesian Models
4.5. Chapter Summary

In this chapter, we designed a detection system to detect abnormal behaviour of a VNE. The detection system is based on a conservative time-synchronisation algorithm with message passing interface used to probe connections by exchanging messages between nodes in logical processes as well as messages within a logical process. A conservative time-synchronisation algorithm was used to determine out-of-order time-stamp messages in the event of failure in a VNE. The order of the message time-stamps is used for detection of a failure. A failure occurs in a VNE when a logical process is receiving an event out of ordered time-stamp messages. In addition, the conservative synchronisation algorithm uses a pre-determined look-ahead value,
Failure Detection in Virtual Network Infrastructure

which is the minimum time that must pass before a failure is considered to occur in a VNE. To increase the scalability of the failure detection system, adopting clustering was adopted and the network was partitioned into multiple logical processes. In addition, we have adopted producer and consumer model in our data collection mechanism to deliver the measurement only in the event of a failure. Results show that a very small number of messages are exchanged in the event of a failure. Therefore, our approach achieved high performance compared with previous work in the detection of failure in a VNE. The failure detection system achieved high accuracy because the results show that the rate of false positive failures is very low during runtime of failure detection. Moreover, the results from the SVM model show that our failure detection system achieved high accuracy in detecting the failure. The advantages of our failure detection system are that it reduces the amount of data by only detecting interesting events, it achieves high accuracy in detection of failure in a VNE and it reduces the overhead on the network by reducing the number of messages exchanged between nodes. Chapter 5 will concentrate on failure prediction in virtual infrastructure components.
Chapter 5: Prediction of Virtual Network Substrate Failures

In a VNE, a failure in the substrate network will affect the many virtual networks hosted by the substrate network. To minimise un-predicted failures, maximise system performance, efficiently use resources and determine how often failures may occur, we must be able to predict failure occurrence. In this chapter, we present a prediction mechanism to forecast the TTF of the VNE components based on time series data. In addition, we use supervised learning based on a SVR model to predict future failures in the VNE. The prediction can be used to establish a tolerable maintenance plan in the event of substrate and virtual network failure.

5.1. Introduction

Because many virtual networks run on a shared substrate network, failure in the substrate network will cause failure many virtual networks. Virtual network failure may results a huge amount of cost and data loss because the entire failed virtual network required to be mapped to different substrate network. Failure prediction is used to forecast failure occurrences in the substrate network using runtime execution states of the system and the history of observed failures. The aim of a failure prediction model is to assess whether there is a risk that the virtual networks cannot operate as expected. The risk assessment depends on system characteristics such as the TTF for each component, whether there a backup in the event of failure and the current load of the system. In addition, failure prediction can be used to predict a critical situation and apply countermeasures to prevent the occurrence of a failure and reduce the time to repair for the upcoming failure. To identify a failure-prone situation in a virtual network, the output prediction is either a binary decision or a continuous measurement and can be used to judge the current situation as more or less failure-prone.
In this chapter, we propose failure prediction method to predict failure in more than one component in a VNE by adopting multiple regression model for time series data and the SVR model. As far as we know, this is the first time that such a modelling technique has been used for the prediction of failure in a VNE. Our contributions are as follows:

- We prepared a failure prediction method that accurately predicts failure of infrastructure components (physical links, physical nodes and virtual networks) in a VNE.
- We used TTF of the physical link, physical node and virtual network to forecasting failure in these components.
- We integrated a time series forecasting modelling technique with the SVR model to predict failure in virtual infrastructure components.
- We evaluated the accuracy of our prediction method by computing the percentage errors between the prediction values and actual values. Our method achieved very high accuracy.
- We evaluated the performance of the SVR model compared with multilayer perceptron (MLP) and Gaussian process. According to our results, the SVR model outperforms the MLP and Gaussian process.

5.2. Problem Overview

In this section, we describe the failure problem in VNE components. The process of instantiating a virtual network by allocating substrate network resources to the virtual network is called virtual network mapping algorithm. Virtual network mapping takes into account the processing and bandwidth capacity requirements of virtual network requests. Multiple virtual networks are mapped onto a shared substrate network with limited network resources such as bandwidth and CPU capacity as well as different configurations and requirements. Therefore,
virtual network mapping is considered an NP-hard problem \cite{62, 68}, and a variety of heuristics have been developed in the literature for efficient mapping.

A single substrate entity failure will affect all virtual entities that are mapped onto it. Therefore, failure occurs in a virtual network when the critical physical node or link fails. There are different scenarios for failure in a VNE, such as maintenance \cite{3, 5} or when the virtual network consumes all of the bandwidth and CPU capacity \cite{6}. The main problem addressed in this chapter is preventing failure before the failure occur in VNE. Adopting preventive failure strategies in a VNE is a promising approach to further enhance system dependability. In addition, predicting failure is becoming an increasingly significant area of research on system dependability to prevent maintenance or reducing time to repair.

Recent research into the prediction of failures in cloud computing has focused on using the unsupervised learning with Bayesian models to deal with unlabelled datasets \cite{109}. One prediction method is based on a Bayesian model for predicting the mean load over a long period to capture trends and patterns of host load in cloud computing \cite{110}. Techniques for predicting node availability were introduced to capture the relationships between the availability of different nodes by using traces taken from distributed system \cite{111}. Predicting failure in a virtual link has been achieved by checking the traffic rate of a user link and adapting the allocated bandwidth based on the predicted traffic \cite{95}. A dynamic meta-learning prediction method adjusts its rules of failure patterns according to accuracy tracing and dynamic re-training with time \cite{112}. Linear traffic predictors have been used to dynamically resize the bandwidth of virtual private network links \cite{113}. Active virtual network management prediction mechanism have been used for active prediction in virtual network \cite{114}. Prediction methods have been used to forecast the future load demand profiles in cloud data centre network by using auto-regressive linear prediction and neural network prediction \cite{115}. The prediction method in \cite{115} is based on Multi-layer neural network perceptrons to predict the
future load of applications in cloud data centers. A framework has been presented to predict demand and provide proactive resources for cloud computation dynamically by using autoregressive integrated moving average [116]. Unsupervised behaviour learning has been used for predicting both anomalies and normal behaviour of virtual machine in the virtualised cloud computing infrastructures. Prediction anomalies behaviours of virtual machines, unsupervised behaviour learning looks for early deviations from normal system behaviour by capturing the pattern of normal virtual machine operation [117]. Failure prediction is essential for developing proactive fault-tolerance mechanisms and self-managing resource problems for system-level dependability and assurance reliable production [118]. Therefore, we develop a prediction mechanism solution to predict the TTF of the virtual infrastructure components based on time series and use Support Vector Machines Regression (SVR) to forecast failure. The reason behind chosen SVR techniques in prediction model because experimental results show the SVR achieved high performance when compared to other powerful techniques such as Artificial Neural Networks (ANNs) [119]. A SVR is achieved high performance than ANN because it is based on the structural risk minimization (SRM) while ANNs is based on empirical risk minimization (ERM). AN ERM is minimizing only the training errors while the SRM is minimizing an upper bound on the generalization error which is required minimum computing while the ERM required high computing since it is deal with large sample sizes [120]. Thus, SVR looks on the generalization performance of the machine to achieve high accurate model by a compromise between model accuracy in the training stage and model ability in forecasting future values, whereas ANNs do not focus at the generalization performance of the machine, which may lead to either overfitting or underfitting problems. This feature lead to increase the SVR efficiency to predict future values.
5.3. Support Vector Regression

The SVR algorithm is important because it can be used to solving simple and complex regression problems, it is robust to very large numbers of attributes with small numbers of instances, it employs very sophisticated mathematical principles to avoid over-fitting, and gives greater experimental results compared with other models [120]. We first give a brief description of the SVR algorithm, and full details can be found in [120] [121]. The SVR formulation can be addressed by minimising an upper bound of the generalisation error rather than minimise the prediction error on the training dataset. This provides the SVR model with greater ability to generalise the input–output correlation realised through its training stage for making good predictions for any given data, not just previously seen data. The SVR maps the input data $x_i$ into a high-dimensional feature space using a non-linear mapping function $\Phi(x_i)$, and then produces and solves a linear regression problem. Thus, The regression function $y = f(x)$ between input vector $x$ and the output for a given a training dataset $D$ that can be approximated using the following function:

$$y_i = f(x) = \omega \Phi(x_i) + b$$ (5.1)

$$K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$$ (5.2)

where $\omega$ and $b$ are coefficients.

The kernel function $K(x_i, x_j)$ is equal to the inner product of the vectors $x_i$, $x_j$ in the high-dimensional feature space, $\Phi(x_i)$ and $\Phi(x_j)$. The kernel function can run any dimension of feature space without the need to accurately calculate $\Phi(x)$ [122]. Any function satisfies the Mercer condition, such as when $K$ takes two points as input and returns a real positive number, it can be used as a kernel function [120]. For example, typical kernel functions are:
Prediction of Virtual Network Substrate Failures

\[ K(x_i, x_j) = (x_i \cdot x_j + \text{coef}0)^q \] \hspace{1cm} \text{Polynomial kernel}

\[ K(x_i, x_j) = e^{\exp\left(-\frac{|x_i-x_j|^2}{2\sigma^2}\right)} \] \hspace{1cm} \text{Gaussian kernel}

where \( q \) represent the degree of the polynomial kernel and \( \sigma \) represent the bandwidth of the Gaussian kernel.

These parameters can be selected accurately by the user to find the best structure of high-dimensional feature space. The SVR achieves the linear regression in the high-dimensional feature space by \( \epsilon \) - insensitive loss function. To prevent over-fitting and improve the capacity for generalisation, the empirical risk and a complexity term \( \|\omega\|^2 \) need to be minimised by a regularised function. Thus, the coefficients \( \omega \) and \( b \) can be estimated by minimising the regularised risk function.

\[
R_{SVR}(C) = R_{emp} + \frac{1}{2}\|w\|^2 = C \cdot \frac{1}{m} \sum_{i=1}^{m} e(y, f(x)) + \frac{1}{2}\|w\|^2 \tag{5.3}
\]

where \( R_{SVR} \) and \( R_{emp} \) denote to the regression model and empirical risks, respectively. \( \frac{1}{2}\|w\|^2 \) is the regularisation term and \( C \) denotes the cost function measuring the empirical risk.

\( R_{SVR} \) is the regression risk dedicated by \( f \) function in predicting the output corresponding to the error in the test dataset. The empirical risk error \( R_{emp} \) or the first term of Eq. 5.3, \( C \cdot \frac{1}{m} \sum_{i=1}^{m} e(y, f(x)) \), denotes to the error in the training dataset estimated by \( \epsilon \) - insensitive function. The \( \epsilon \) - insensitive ignores errors if the difference between the predicted value \( f(x) \) and the observed value \( y \) is smaller than \( \epsilon \), otherwise it equals the absolute value of \( |y - f(x)| - \epsilon \).

\[
e(\ y, f(x)) = \max(0, |y - f(x)| - \epsilon) \tag{5.4}
\]
The parameter $C$ calculates the penalty when an error occurs by regulating the trade-off between the empirical risk and the regularisation term. The $C$ parameter controls the balance between model complexity and the degree to which deviations larger than $\varepsilon$ are tolerated in the optimisation formulation. For example, if $C$ is too large, the empirical risk will be increased in relation to the regularisation term and the optimisation objective is to minimise the empirical risk only.

The penalty is acceptable only if the fitting error is larger than $\varepsilon$, which controls the width of the $\varepsilon$ area that is used to fit the training dataset. The SVR function depends on $\varepsilon$ value, bigger $\varepsilon$ value results in fewer support vectors are selected thus more flat estimates. Thus, the SVR model’s performance depends on parameters $C$ and $\varepsilon$ need to be controlled by the user.

To estimate $\omega$ and $b$, two slack variables $\xi_i$ and $\xi_i^+$ are introduced to minimise the error in the training dataset outside the $\varepsilon$–insensitive zone. The slack variables $\xi_i^+$ and $\xi_i$ measure the positive and negative errors, respectively, in the training dataset and assume non-zero values outside the $-\varepsilon, \varepsilon$ region.

The SVR model fits the function $f(x)$ by minimising the errors in the training dataset. The errors are minimised by minimising $\xi_i$ and $\xi_i^+$ or minimising the regularisation term $\frac{1}{2}||w||^2$ to raise flatness of $f(x)$ function as shown in figure 5-1. Thus, Eq. 5.5 can be formulated as minimising of the following functions.

Minimise

$$R_{SVR}(\omega, \xi_i^+) = \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{m}(\xi_i + \xi_i^+)$$  \hspace{1cm} (5.5)

Subject to the following:

$$\begin{cases}
y_t - \omega\Phi(x_i) - b \leq \varepsilon + \xi_i \\
\omega\Phi(x_i) + b - y_t \leq \varepsilon + \xi_i^+ \\
\xi_i, \xi_i^+ \geq 0 \quad i = 1, 2, \ldots, n
\end{cases}$$
5.4. Predicting Failure in VNE

In this section, we propose a new approach for predicting failure in virtual infrastructure using the time series forecasting modelling technique and the SVR model. Time series data are a set of observations that occur over time or a collection of random variables indexed in time to represent samples of a system’s behaviour over time [123]. The forecast of the system’s behaviour progression over time involves the forecast of the time series explaining the system’s behaviour [124]. The architecture of the failure prediction model components are illustrated in figure 5-2.

The input data of our failure prediction model are the TTF for each component (physical links, physical nodes and virtual networks) in the VNE. The MTTF can be used to measure the probability of failure by integrating the probability distribution function, that is, $MTTF = \int_{t_0}^{\infty} f_T(t) \, dt$. Therefore, TTF is chosen as a feature in our prediction model because it can be used to measure the probability of the physical network failing at or before time $t_0$ [63].
From the TTF input dataset, we then construct lagged variables. Lagged variables are the main mechanism to capture the relationship between the past and current values of a series in support vector machines learning algorithms. To create periodicity, we create a set of lagged input variables within a fixed-length window in the time series. In our model, we use variables lagged between 1 and 24 hours, where 1 is the minimum previous time step to create a lagged variable that holds the target value at time $t - 1$, and 24 is the maximum previous time step to create a lagged variable that holds the target value at time $t - 24$. Thus, the periods between the minimum and maximum lag will become the lagged variables. When the lagged variables have been constructed, the variable can be predicted from itself.
We are interested in predicting failure in more than one component because multiple factors can produce failure in a VNE, for example, physical link failure, physical node failure and virtual network failure. Therefore, we adopted a multiple regression model for the time series data to predict the future failure of each component in the VNE. The lagged variables created from the TTF input dataset are used in the multiple regression model. We used the lagged variables $x_{i,t-1}, x_{i,t-2}, \ldots, x_{i,t-p}$ in the multiple regression model to represent the TTF of the physical links, physical nodes and virtual networks. The aim of multiple regression model is to forecast each entry in the time series accurately by finding a formula that captures the autocorrelation between the lagged values and the current values of the series. Thus, the time-series forecasting is modelled as follows:

$$Y_t = f \left( X_{i,t} \right) = f(x_{i,t-1}, x_{i,t-2}, \ldots, x_{i,t-p})$$  \hspace{1cm} (5.6)

where $Y_t$ is the output observation at time $t$ of the inputs $X_{i,t}$, and $X_{i,t}$ is the input vector of lagged variables $(x_{i,t-1}, x_{i,t-2}, \ldots, x_{i,t-p})$, $i$ is a constant number $i = 1, 2, 3, \ldots, n$ ($i = 1$ represents a vector of lagged variables TTF of physical links, $i = 2$ represents a vector of lagged variables TTF of physical nodes and $i = 3$ represents a vector of lagged variables TTF of virtual networks), $t$ is the number of observations at time, $p$ represents the number of past observations and $f$ is a function to find autocorrelation between the time-lagged value and the current value. Thus, Eq. 5.6 can be written as follows:

$$Y_t = f \left( x_{1,t-1}, x_{1,t-2}, \ldots, x_{1,t-p} \right), \left( x_{2,t-1}, x_{2,t-2}, \ldots, x_{2,t-p} \right), \left( x_{3,t-1}, x_{3,t-2}, \ldots, x_{3,t-p} \right)$$

Thus, the training pattern can be constructed in the SVR model as shown in table 5-1, where $t - p$ is the total number of training data, $p$ is the number of lagged variables, $X_i$ is the lagged variables vector for the VNE components ($i = 1$ for the lagged variables TTF of physical links, $i = 2$ for the lagged variables TTF of physical nodes and $i = 3$ for the lagged variables TTF of virtual networks) and $Y$ is the predicted output.
Table 5-1 Training Pattern in SVR Model

<table>
<thead>
<tr>
<th>$x_{i,1}$</th>
<th>$x_{i,2}$</th>
<th>...</th>
<th>$x_{i,p}$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{i,2}$</td>
<td>$x_{i,3}$</td>
<td>...</td>
<td>$x_{i,p+1}$</td>
<td>$x_{i,p+2}$</td>
</tr>
<tr>
<td>$x_{i,3}$</td>
<td>$x_{i,4}$</td>
<td>...</td>
<td>$x_{i,p+2}$</td>
<td>$x_{i,p+3}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_{i,t-p}$</td>
<td>$x_{i,t-p+1}$</td>
<td>...</td>
<td>$x_{i,t-1}$</td>
<td>$x_{i,t}$</td>
</tr>
</tbody>
</table>

The multiple regression model is a complex and nonlinear problem because there are multiple predictor variables in the model. Therefore, we adopted the SVR model to solve the nonlinearity problem and identify the correct time series model for forecasting failure in a VNE. The inputs used by the SVR model are the lagged variables of the time series, and these variables are used to capture the unknown relationship between the lagged input variables and the output. In addition, to solve the nonlinear problem in the multiple regression model and to forecast future failure, the $f$ function needs to be approximated by the SVR model. The SVR model parameters $C$, $\sigma$ and $\varepsilon$ need to be chosen by the user. Therefore, we train the SVR model with different values of $C$, $\sigma$ and $\varepsilon$ to find the optimal prediction model to capture the correlation between the time-lagged input and the output.

The prediction quality of the SVR on the training dataset can be evaluated using the RMSE metric to measure the difference between the values predicted by the model and the real values of the modelled dataset [125]. If the RMSE is very low, the model is selected, otherwise we choose different values for the SVR parameters ($C$, $\sigma$, $\varepsilon$).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(S_{p,i} - S_{r,i})^2}{N}}$$  \hspace{1cm} (5.7)
where $S_{r,i}$ are the actual values, $S_{p,i}$ are the predicted values at time $i$ and $N$ is the number of forecasts.

Following successful training, the SVR model with the lowest error rate according to the RMSE metric can be selected. The selected SVR model can then be evaluated using the testing dataset to predict the failure $Y_{t+k}$ at different time steps $k$. For example, $k = 1$ uses the $t$-th TTF as input to forecast a one-step ahead $t + 1$-th TTF as output. The second prediction is two steps ahead when $k = 2$ and uses the same input as before and predicts the $t + 2$-th TTF as output.

The results of the SVR model performance was compared with MLP and Gaussian process algorithms by calculating the normalised root mean square error (NRMSE) [126] for each prediction model using the following equation:

$$NRMSE = \frac{RMSE}{S_{max} - S_{min}}$$

where $S_{max}$ is the maximum of the actual values and $S_{min}$ is the minimum of the actual values.

### 5.5. Performance Analysis

In this section, we evaluate the performance of the proposed SVR prediction model and compare it with a variety of techniques, such as MLP and Gaussian process.

#### 5.5.1. Experimental Set-up

We used a discrete-event Network Simulator 3 [103] and Boston University Representative Internet Topology generator [106] to generate a hierarchical topology to represent substrate network topology and virtual network topology, as shown in figure 5-3. The substrate network consists of 50 physical nodes where each node is connected to two neighbour nodes. CPU and bandwidth resources are uniformly assigned for each node and link. The TTF is assigned for
each physical node and link. The virtual network topology was generated using the virtual network mapping proposed in [17]. Up to four virtual nodes can be mapped onto each physical node with an average lifetime of 1,000 time units for each virtual network request through the simulation of 50,000 time units in a substrate network.

Weka version 3.7.13 with forecast package [108] was used to build a SVR model based on the training dataset to find the optimal function $f$ with given values of the SVR parameters $C$, $\varepsilon$ and $\sigma$ to capture the unknown relationship between the time-lagged input and the output.
5.5.2. Data Sets

We used Network Simulator 3 in our research as a platform to be used to analyse network features and collect interesting data (TTF). In our model, we assume that the component failure time decreases linearly according to the number of virtual networks sharing the substrate component. In addition, we assume that the virtual network is mapped onto the physical network without redundancy. When the physical component fails, the virtual network fails. The TTF of the hardware and software components are shown in Table 5-2, which is based on factory specifications and adopted from recent literature [71, 127-130]. Based on table 5-2, random numbers were uniformly generated over the interval [35, 100] to represent the TTF of the infrastructure components adopted by the mapping algorithm. The collected TTF data may be treated as a time series of failure times for components in a VNE.

<table>
<thead>
<tr>
<th>Node</th>
<th>TTF (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Switch/Router</td>
<td>320,000</td>
</tr>
<tr>
<td>Virtual Machine Monitor</td>
<td>2,880</td>
</tr>
<tr>
<td>Network Interface Card</td>
<td>6,200,000</td>
</tr>
<tr>
<td>CPU</td>
<td>2,500,000</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>200,000</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>1,440</td>
</tr>
<tr>
<td>Memory</td>
<td>480,000</td>
</tr>
<tr>
<td>Optical Link</td>
<td>19,996</td>
</tr>
</tbody>
</table>
5.5.3. Results and Discussions

From our experiment results, we found the optimal parameters that best fit our training dataset for building the SVR model to predict the failure in VNE, as shown in table 5-3. We used 9,702 instances for building the SVR model for one-step ahead and two-steps ahead forecasting the TTF in virtual network, physical link and physical node.

<table>
<thead>
<tr>
<th>SVR Parameters</th>
<th>One Step Ahead</th>
<th>Two Steps Ahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>1560</td>
<td>1560</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.00001</td>
<td>0.00001</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>0.00001</td>
<td>0.00001</td>
</tr>
</tbody>
</table>

5.5.3.1. Prediction Failure in Virtual Networks

The first SVR model was built using the TTF for virtual networks as input to predict one-step ahead ($t + 1$) TTF as output (short-term prediction). The second SVR model was built using the same input TTF to predict two steps ahead ($t + 2$) TTF. Figure 5-4 and figure 5-5 show the actual TTF with the results of the one-step and two steps prediction, respectively, for failure occurrences in the virtual network. The prediction results are very close to the actual TTF values.
Prediction of Virtual Network Substrate Failures

Figure 5-4 Prediction of One Step Ahead TTF of the Virtual Network

Figure 5-5 Prediction of Two Steps Ahead TTF of the Virtual Network
5.5.3.2. Prediction Failure in Physical Nodes

To forecast failure in the physical nodes, we built a SVR model using the TTF for physical nodes as input to predict one-step ahead \((t + 1)\) TTF as output (future failure prediction). In addition, we used the same input TTF to make a two steps ahead prediction \((t + 2)\) TTF. Figure 5-6 and figure 5-7 show the actual and predicted TTF for the one-step ahead and two steps ahead prediction, respectively, for failure occurrences. The predicted values and the actual TTF values for the physical nodes are identical. The SVR model achieved very accurate results because the difference between the predicted values and the actual values was very low.

![Figure 5-6 Prediction of One Step Ahead TTF the Physical Nodes](image)

Figure 5-6 Prediction of One Step Ahead TTF the Physical Nodes
5.5.3.3. Prediction Failure in Physical Link

Prediction of the failure in physical links involved forecasting the TTF for each physical link in the VNE. The SVR model uses an input TTF for each physical link to predict \((t + 1)\) TTF as output. Similarly, for two steps ahead prediction, the model uses the same input TTF as before but make a two-step ahead prediction \((t + 2)\) TTF as output. Figure 5-8 and figure 5-9 show the actual TTF with the results of the one-step ahead and two steps ahead, respectively, prediction for failure in the physical links. The results show that the predicted values and the actual values are very close to each other, which means that the prediction results are accurate because the difference between the predicted values and actual values is very low.

Figure 5-7 Prediction of Two Step Ahead TTF of the Physical Nodes
Figure 5-8 Prediction of One Step Ahead TTF of the Physical Links

Figure 5-9 Prediction of Two Step Ahead TTF of the Physical Links
5.5.4. Validation

The RMSE is used for the evaluation of a numerical prediction and measures the average of the square of all the errors between the predicted values and the observed values. The RMSE gives a high weight to large errors. Therefore, the RMSE is useful to measure error rates when large errors are especially unwanted in the evaluation of a numerical prediction [131].

To validate the predicted results from the SVR for virtual networks, physical nodes and physical links, we used a testing set method by splitting the dataset into a training dataset and a test dataset. The proportions used for the testing dataset were 10%, 20% and 30%, which means that the first experiment was run with 90% of the data used for the training dataset and 10% used for the test dataset. From the results of each run, we computed the RMSE, and then calculated the average RMSE for the three runs.

The results in table 5-4 show that our SVR models achieved a very good accuracy because the RMSE values are very low: 0.16%, 3.13% and 1.83 for the VN-SVR, physical node-SVR and physical link-SVR models, respectively. The low value of the RMSE indicates that the SVR models achieved very high accuracy in forecasting failure in the VNE. Since the prediction accuracy could not reach 100% by using the most advanced learning algorithms, however our predictions achieved high accuracy in forecasting the TTF of virtual networks, physical nodes and physical links.

<table>
<thead>
<tr>
<th>% Testing Dataset</th>
<th>Virtual Network – SVR Model</th>
<th>Physical Node – SVR Model</th>
<th>Physical Link – SVR Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 step</td>
<td>2 steps</td>
<td>1 step</td>
</tr>
<tr>
<td>10</td>
<td>0.092</td>
<td>0.093</td>
<td>2.42</td>
</tr>
<tr>
<td>20</td>
<td>0.102</td>
<td>0.112</td>
<td>2.96</td>
</tr>
<tr>
<td>30</td>
<td>0.279</td>
<td>0.285</td>
<td>4.01</td>
</tr>
<tr>
<td>Average RMSE</td>
<td>0.16</td>
<td>0.16</td>
<td>3.13</td>
</tr>
</tbody>
</table>
5.5.5. Failure Prediction Performance

To maximise the performance of the SVR in forecasting the TTF in virtual infrastructure components, three parameters, $C$, $\sigma$ and $\varepsilon$, need to be controlled in setting the SVR model. The SVR model’s performance on the test dataset is measured by computing the NRMSE. The NRMSE provides an indication of how well the predictor is performing. Low values of the NRMSE indicate that the predictor performs well. Two different regression models – MLP and Gaussian process – were used to compare the performance of the SVR model. The performance comparison was based on 10%, 20% and 30% of the dataset set aside as a test dataset.

The results in table 5-5 show that the NRMSE values for both one-step ahead and two steps ahead prediction of failure in virtual network was 0.0008 for the SVR model. In addition, the NRMSE values for the MLP model were 0.0461 for one-step ahead and 0.0893 for two steps ahead prediction. For the Gaussian process model, the NRMSE values were 0.3355 for one-step ahead and 0.3363 for two steps ahead prediction. Because the NRMSE computed by SVR model is lower than the NRMSE values computed by the MLP and Gaussian process models, the SVR outperforms the Gaussian process and MLP models for forecasting the TTF in virtual networks.

<table>
<thead>
<tr>
<th>% Testing Dataset</th>
<th>Virtual Network – SVR</th>
<th>Virtual Network – MLP</th>
<th>Virtual Network – Gaussian Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 step</td>
<td>2 steps</td>
<td>1 step</td>
</tr>
<tr>
<td>10</td>
<td>0.0009</td>
<td>0.0009</td>
<td>0.0598</td>
</tr>
<tr>
<td>20</td>
<td>0.0005</td>
<td>0.0006</td>
<td>0.0235</td>
</tr>
<tr>
<td>30</td>
<td>0.0009</td>
<td>0.0009</td>
<td>0.0550</td>
</tr>
<tr>
<td>Average NRMSE</td>
<td>0.0008</td>
<td>0.0008</td>
<td>0.0461</td>
</tr>
</tbody>
</table>
The results in table 5-6 show that the NRMSE values for one step ahead and for two steps ahead prediction of failure in physical nodes were 0.0015 and 0.0017, respectively, for the SVR model. The other models show that the MLP and Gaussian process models achieved higher NRMSE values. Thus, the SVR model achieves higher performance than the Gaussian process and MLP models for forecasting TTF in the physical node component in a VNE.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 step</td>
<td>2 steps</td>
<td>1 step</td>
</tr>
<tr>
<td>10</td>
<td>0.0021</td>
<td>0.0024</td>
<td>0.0068</td>
</tr>
<tr>
<td>20</td>
<td>0.0013</td>
<td>0.0014</td>
<td>0.0189</td>
</tr>
<tr>
<td>30</td>
<td>0.0012</td>
<td>0.0012</td>
<td>0.1843</td>
</tr>
<tr>
<td>Average NRMSE</td>
<td>0.0015</td>
<td>0.0017</td>
<td>0.0700</td>
</tr>
</tbody>
</table>

Table 5-7 shows that the SVR model achieves the lowest NRMSE values for one-step ahead and two steps ahead prediction of failure in the physical link component in a VNE. Therefore, the SVR outperforming Gaussian process and MLP models for forecasting the TTF of physical link components in VNE.

We conclude that SVR models achieved high performance with a big dataset or small dataset because the predictors depend on their parameters to fit the data into a model.
5.6. Chapter Summary

In the VNE, multiple virtual networks run on a shared physical network, and therefore, a failure in a physical node or a physical link can affect many virtual networks. The consequence of a failure in physical network include the loss of critical data lost, the need for reconfiguration of the filed virtual networks and profit loss due to the failure. Therefore, we need a system to predict failure before it takes place. In this chapter, we designed a prediction mechanism to forecast the failure of the virtual infrastructure components based on time series and SVR models. Each component in a VNE has a factory-specific feature such as TTF. We modelled the time series as a set of TTF observations ordered in time. To predict the TTF for each component, we used SVR based on the input time series as a one-step ahead or two steps ahead. We evaluated the SVR model by using the dataset and comparing it with other technologies such as MLP and Gaussian process. The results show that the NRMSE for the SVR model is very low compared with the NRMSE of the other models. In other words, the SVR model achieved high performance in prediction of failure in a VNE because the predicted results are very close to the actual values.
Chapter 6: Conclusions and Future Directions

This chapter provides an overall summary and discusses the proposed methodologies, results and the conclusions in this thesis. The first section discusses the accomplishments of this work, and the second section highlights possible future research directions.

6.1. Accomplishments

The first question addressed in this thesis was: what is the probability that the substrate network functions? The answer to this question is we have presented a framework to estimate the probability of the system providing the required functionalities, as presented in Chapter 3. The probability that the system is working or failed during time $t$ can be calculated using reliability block diagrams to assess system and sub-system reliability. A reliability block diagram is a combinatorial model used for analysing the reliability of components arranged in series, in parallel or a combination of both series and parallel. The functionality of the system depends on the arrangement of its components. For example, in a series system, if any component fails, then all the whole system will fail, while in a parallel system, the system will fail when all of the components in the system fail. Reliability block diagrams were used to represent the three different mappings and the reliability of the system operational state given by its working components based on its series and parallel arrangements. We adopted series and parallel arrangements to model virtual network mapping onto a substrate network as a single mapping, passive mapping or active mapping. In the single mapping case, the reliability of the system is single-point failure. Therefore, it requires all physical infrastructure components to be working, and any failure in physical infrastructure (router or link) will lead to failure of the entire mapping. The reliability and failure probabilities for passive mapping and active mapping are higher than for single mapping because active and passive mapping uses a combination of
parallel and series component connections. The results in Chapter 3 shows that the reliability decreased significantly from 89% to 33% for simple mapping when the virtual networks increased from 100 to 1000 virtual nodes mapped onto the substrate network. In addition, the results in Chapter 3 show that the reliability for active and passive mapping was higher than for single mapping. For example, the reliability decreased from 99% to 91% and from 97% to 70% for active and passive mapping, respectively, when virtual networks increased from 100 to 1000 virtual nodes mapped onto the substrate network.

The second question addressed in this thesis was: how to make virtual networks reliable with the least resources? This problem was solved using the continuous-time Markov chain model to represent virtual network mapping without redundancy (simple mapping) or with redundancy (passive mapping) for analysing the reliability and availability of virtual network. The lifetime of a virtual network can be estimated from the MTTF and the MTTR for substrate network components. MTTF and MTTR are used for analysing the lifetime for each substrate network component and the lifetime of the system. The lifetime of a virtual network increases by mapping the virtual network onto more than one component of the substrate network. The reliability of the simple mapping decreased dramatically when the virtual network was mapped into one component (i.e., the MTTF for the series system decreased). The reliability of the passive mapping increased when the substrate components were connected in parallel (i.e., the MTTF for the parallel system increased). Thus, we can increase the lifetime of the system by adopting the virtual network mapping with redundancy. In addition, passive mapping achieved very high performance with fewer resources than active mapping because the stand-by redundancy in the passive mapping starts when the primary components fail. In passive mapping, the primary active component is in the working state while the secondary component in stand-by state. Thus, the MTTF for the system is increased by combining the MTTF of primary components and the redundancy of stand-by components. The results in
Conclusions and Future Directions

Chapter 3 show that the availability of the system increased with the least resources (i.e., passive mapping is 100% all the time during 50 hours running the virtual network). While the results show active mapping decreased availability from 99% to 93% and simple mapping decreased availability from 97% to 92% during the 50 hours of running the virtual network.

The third question addressed in this thesis was: how to check if the component of substrate network is functioning? To check whether the component is functioning or not, we developed a failure detection mechanism based on the conservative time-synchronisation algorithm and message passing interface. The message-passing interface is used for probing connections between point-to-point nodes by message exchange and the conservative time-synchronisation algorithm is used to determine the time-out before considering that a failure event has occurred in the VNE. The failure detection system was designed to work in a large-scale virtual network with small numbers of message exchanged and short time for failure detection. The results in Chapter 4 show that the failure detection system achieved a high true positive failure rate (95.5%) and a low false negative failure rate (5%). Because we partitioned the VNE topology into multiple clusters, failure detection is restricted to a few substrate nodes. Therefore, the failure detection approach achieved efficiency in the time to failure detection (0.04562 seconds for five clusters) and the number of messages exchanged (seven messages for five clusters).

The fourth question addressed in this thesis was: when does the failure occur in the substrate network? To check when the substrate component failed, we developed a prediction mechanism to forecast failure in more than one component in the VNE. The failure prediction method is based on time series and SVR models. We constructed lagged-variables from the TTF of physical links, physical nodes and virtual networks. The time series was modelled using multiple regression that was integrated with the SVR model for forecasting the future failure in these components. The results in Chapter 5 show that our prediction method achieved high
accuracy in forecasting the failure. The RMSE values are very low (0.16%, 3.13% and 1.83 for the virtual network–SVR, physical node–SVR and physical link–SVR models respectively), and therefore, the SVR model achieved very high accuracy in forecasting the failure in the VNE. In addition, the SVR models achieved high performance in forecasting the failure of substrate components compared with the MLP and Gaussian process. For example, the NRMSE value was 0.0008 for forecasting the failure in virtual network by the SVR model. While the MLP and Gaussian process models show, higher NRMSE values (0.0461 and 0.3355, respectively). Thus, this means that the SVR model achieved higher performance than the Gaussian process and MLP models for forecasting the failure in VNE components.

6.2. Limitations and Future Work

In spite of we have introduced various techniques to enhance the virtual infrastructures dependability but there are still some limitations and challenges that need to be addressed before these techniques can be deployed in real world scenario. For future work, we plan to pursue several extensions to this thesis as follows:

- We are considering assessing optimal reliability design for the virtual network allocation in physical network. To assure system reliability, the virtual network is mapped onto the physical network with sufficient backup for virtual nodes and links. While a backup mechanism increases system reliability, the use of the physical resources may be significantly reduced. Thus, we plan to extend our dependability model to guarantee optimal reliability for a virtual network with optimal physical resources allocation. These techniques can reduce the use of physical resources for virtual network while guaranteeing system reliability.
Conclusions and Future Directions

- We plan to use reliability importance to provide a numerical rank to determine which components are more important to system reliability or more critical to system failure. In addition, reliability importance will be used to analyse the system availability according to the most important components.

- We used continuous-time Markov chain to model the VNE to capture the dynamic behaviour of the system in the event of failure. We plan to use a different model, such as the stochastic Petri net model, to analyse system reliability by adopting different recovery strategies with several redundant topologies and considering different failure modes to further enhance VNE dependability.

- We used two approaches for mapping virtual network onto a physical network to guarantee reliability. The first approach is passive mapping that maps the virtual network onto two physical routers, and when the primary physical router fails, the stand-by physical router starts working. The second approach is active mapping that maps the virtual network onto a primary router and a backup router running simultaneously. While the two approaches keep redundancy for reliable operation, keeping redundancy idle in normal operation leads wasting the cost and the resources of operation. Therefore, we will study a different approach that shares the backup between different critical nodes and find intelligent mechanisms to increase the reliability of the VNE.

- In our detection of failure in VNE mechanism, we used a hierarchal topology to represent a VNE. In future, we plan to study different virtual network topologies such as mesh topology. In addition, we focused in our study about scalability, flexibility and autonomic features in detection a failure in virtual network in one domain, further work required when virtual networks mapped into more than one domain.
Conclusions and Future Directions

- In detection the failure in VNE, we used message-passing interface for probing connection between point-to-point nodes and a conservative time-synchronisation algorithm to determine out of order time-stamp messages in Network Simulator 3. In future, we will apply the same detection mechanism to an actual VNE and compare the results with different algorithms.

- The proposed prediction mechanism is based on TTF feature of VNE components. In future, we will extend the features that include CPU, bandwidth and memory to predict failure in a VNE.
References


References


References


[127] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you?," in FAST, 2007, pp. 1-16.

