Functional Topology of Networks

Sabri-E-Zaman

A thesis submitted to the University of London in partial fulfilment of
the requirements for the degree of

Doctor of Philosophy

School of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

August 2016

Dedicated to my parents and my wife

Statement of Originality

I, Sabri-E-Zaman, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature:

Sabri-E-Zaman

Date: 23rd August 2016

Abstract

In order to utilise network resources efficiently, we need a strong knowledge of how the resources are shared and provisioned. However, this information is often unavailable due to the complexity of modern networks, the restrictive access to information describing their configurations and accuracy/reliability issues regarding information provisioning methods. Here, we propose the concept of *functional topologies* to deduce how resources are shared between different traffic flows. A functional topology describes the dependencies between traffic flows as a graph of interactions; this is in contrast to typical network graphs that model the physical connections between network components (routers and hosts). Unlike other work relying on in-network data, this topology is constructed solely at end hosts by measuring interdependencies of traffic flows via cross-correlation analysis. In order to measure the complete sets of interdependencies of traffic flows, different time intervals are used for sampling time series data. It is shown that these time intervals are related to maximum delays of traffic flows in network. The results of cross-correlation analysis are validated using well-known inverse participation ratio (IPR). As a part of the validation process, the results are analysed and compared with dominant/important flows of the network obtained by a new technique that uses eigendecomposition and spanning tree algorithm. The methodology of measuring interdependencies of traffic flows is validated and evaluated using real world data from a sensor network, as well as detailed simulation modelling different network topologies e.g. local area network. All the dependency measurements of traffic flow results are fed into a novel algorithm to construct functional topology of the network. Result shows that the algorithm constructs accurate functional topology of the network. Functional topology simplifies network topology by considering only nodes that create dependencies among traffic flows. With the help of this topology, end hosts can gain insight into resource provisioning of a network without requiring ISP assistance.

Acknowledgements

Firstly, I want to thank the Almighty for helping me get this far in my life against all odds. Thank You for giving me the determination, courage and wisdom to embark upon and complete this journey.

I want to express my deepest gratitude towards my supervisor Dr. Raul Mondragon who has been exceptionally generous, patient and supportive throughout the course of my PhD. I am really grateful to him for giving me the opportunity to undertake this research and for all the assistance and invaluable advice he has been providing me. It would have been impossible for me to accomplish this work without his guidance. I am indebted to the rest of my research panel, Dr. Chris Phillips and Dr. Felix Cuadrado for their useful suggestions and insights on this research work. A Special thanks to my head of research group Prof. Steve Uhlig for his continuous guidance and fruitful suggestions during my research.

I am grateful to Dr. Gareth Tyson for all the help throughout the course of this PhD, proof-reading of this thesis and providing extremely useful inputs on my research.

Most of all, I am extremely grateful to my parents, M.M.Asaduzzaman and Nilufar Zaman for all the sacrifices they have made for me. I will remain indebted to them for the unconditional support and inspiration they have always given me, with all my decisions throughout life. I would like to thank my brother Navid E Zaman for all his help and support. Thanks to my wife Amna Abdul Wahid for her selfless cooperation during my studies and being the source of my strength.

I would like to thank my colleagues and friends at Queen Mary University of London, Marjan Falahrastegar, Kishan Patel, Jie Deng, Shan Huang, Timm Bottger, Hamed Saljooghinejad, Dr. Ammar Lilamwala, Dr. Ling Xu, Dr. Bo Zhong, Dr. Manik Gupta, Dr. Manmohan Sharma, Dr. Oleksandr Sushko, Dr. Khaleda Ali, Dr. Iftekhar Mobin and many others for always being so nice and helpful towards me. I am also thankful to Queen Mary IEEE student branch and EECS Systems Support for their assistance and support during my time at Queen Mary University of London.

Contents

List of Figures

List of Tables

List of Abbreviations

ASSOC	Associate Sensor Node
CBR	Constant Bit Rate
CO	Carbon Monoxide
EM	Expectation-Maximisation Algorithm
FIFO	First In First Out
ICMP	Internet Control Message Protocol
IPR	Inverse Participation Ratio
ISP	Internet Service Provider
LAN	Local Area Network
MST	Minimum Spanning Tree Algorithm
NS2	Network Simulator 2
OD	Origin-destination Matrix
OPNET	Optimized Network Evaluation Tool
P2P	Peer to Peer
PMFG	Planar Maximally Filtered Graph
QoS	Quality of Service
REP	Representative Sensor Node
RMT	Random Matrix Theory
SN	Sensor Nodes
SW	Sampling Windows
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WAN	Wide Area Network
WSN	Wireless Sensor Network

List of Publications

- Sabri -E-Zaman and Raul J Mondragon. *Functional Topology of Networks*. (To be submitted).
- Sabri -E-Zaman, Manik Gupta, Raul J Mondragon and Eliane Bodanese. *An Eigen Decomposition Based Adaptive Sampling Technique for Wireless Sensor Networks.* In Proc. 39th IEEE Local Computer Networks Conference, Edmonton,Canada (2014)
- Sabri -E-Zaman and Raul J Mondragon. *Measuring Interdependencies and Transitivity of End to End Traffic Flows by Traffic Correlation.* In Proc. PhD Forum of 21st IEEE International Conference on Network Protocols (ICNP2013 PhD Forum , Gottingen, Germany (2013)
- Sabri -E-Zaman and Raul J Mondragon. *Inferring Logical Network Topology by Traffic Correlations.* In Proc.14th Annual Postgraduate Symposiumon the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK (2013)

Chapter 1

Introduction

1.1 Motivation

Efficient utilisation of network resources requires a strong knowledge of how the resources are shared within the network. This information, however, is rarely available due to the complexity of modern networks and the restrictive access to the information describing their configurations (e.g. due to security threats). Unavailability of this information makes it difficult for third party stakeholders (such as content providers, peer to peer networks etc.) to understand and predict network performance. Hence, these third party stakeholders can often struggle to meet quality of service (QoS) requirements as they depend on the performance of internet service providers (ISP) [1, 2, 3, 4]. Providing insight about network behavior could be vital for applications to adapt and optimise their activity.

In order to get information about how resources are shared within an ISP network, we propose the construction of *functional topologies*. A functional topology describes the dependencies between traffic flows as a graph of interactions; this is in contrast to typical network graphs that model the physical connections between network components (routers and hosts). This topology is constructed by measuring the interdependencies of traffic flows via cross-correlation analysis of data collected at the end points of the network. Through this black-box approach, end hosts can gain insight into the resource provisioning of a network without requiring ISP assistance. This can be helpful for stakeholders to analyse perfor-

mance and avoid possible bottlenecks of the network. Of course, this could also offer benefits to ISPs, who could potentially see their networks used by third party services more efficiently (without having to directly expose information about themselves).

Importantly, the collection, analysis and sharing of network data is not limited to the above applications. There are many other situations where the principles could be applied. For instance, in Wireless Sensor Networks, spatially nearby nodes will often collect very similar data. The ability to detect such redundant data collection within a network could offer many benefits (e.g. reduction on data consumption and energy savings). The above suggests that the development of an integrated methodology for collecting, analysing and sharing various types of network data could be highly beneficial to a number of domains. In this thesis, a generic methodology is designed that allows data to be collected and correlated for various key purposes.

1.2 Overview of the work

1.2.1 Research Problem

The process of delivery content to end users (customers) in modern communication network has been changed. Nowadays, third party stakeholders (such as different content providers) are also responsible for delivering content alongside ISPs. Benefits of having third party stakeholder to deliver traffic can be achieved when ISP-third party stakeholders collaborate with each other. With the collaboration of ISP-third party stakeholder, optimum resources utilisation and meeting QoS requirements are possible to achieve. Third party stakeholders require topology information for efficient traffic delivery and resource utilisation. But this information is often unavailable from ISPs due to many reasons such as security threats. Due to unavailability of this information third party stakeholders struggles to meet QoS requirements. In this research, we look to infer maximum topology information without any help from an ISP in such a way that third party stakeholder can use this information to optimise their process and ISPs are comfortable with sharing this information as it would not cause any security threats.

Thus, we propose the construction of functional topology of the network. It provides topology information by describing dependencies of traffic flows of the network, which can be useful to third party stakeholder. On the other hand, it does not represent any actual node related information, so, we can

avoid security threats.

1.2.2 Workflow

We construct a functional topology of the network by measuring the interdependencies of traffic flows using a cross-correlation method. Previous studies [5, 6, 7, 8, 9, 10] show that this method can measure interdependencies of traffic flows in communication networks. In this method, traffic is measured at the end points (destination) of the network. There are two kinds of outputs produced by the method: negative and positive correlations. A negative correlation occurs when two traffic flows compete to pass through a common network resource (such as the buffer of a router) in the network. On the other hand, a positive correlation occurs as a result of competing for resources among two pairs of traffic flows. For example, flows a and b compete for one resource and flows b and c compete for another common resource in the network. As both flows a and c share a common flow b, positive correlation occurs between flows a and c. We call this relationship as *transitivity*. The results obtained from cross-correlation method represent how the resources are shared by traffic flows in the network.

We find that cross-correlation between two flows becomes strong if they pass through a shared congested node. Therefore we assume that we can measure dependencies of traffic flows accurately if there is a certain level of congestion in network. In this thesis, we use 70% of utilization of buffer of nodes in network as threshold of congested network. We find that delay time starts increasing exponentially if we set buffer utilization on 70% onwards in NS2 simulator. We also obtain some information from communication network industry that 70% utilization on link level is the threshold for upgrade. On the basis of this information, we decide to set 70% utilization of buffer as threshold of congested network. While measuring a congested network, we find that traffic flows with small delays exhibit stronger cross-correlations (as we count packets at the destinations) than flows with large delays when using small sampling window. This observation leads us to understand that capturing interdependencies of traffic flows using cross-correlation is related to delays in the network. Thus, to overcome the effect of delay and get accurate cross-correlation among all flows (ranging from small to large delays), we introduce the use of a range of sampling windows in the time series of packet counts at the end points. We show that we get peak (maximum or minimum) cross-correlation coefficients if we take sampling windows according to maximum delays of traffic flows. Thus, we consider a range of sampling windows

according to the maximum delays of flows in the network. Using different sampling windows in the time series data of traffic flows allow us to measure complete set of interdependencies among traffic flows.

To validate these cross-correlation results, we use inverse participation ratio (IPR). Inverse participation ratio (IPR) is introduced in localization theory [5, 6]. IPR provides the number of dominant flows in a network. We match the number of highly correlated traffic flows from cross-correlation matrix with IPR value. We find that the IPR value is bigger than this number of highly correlated pairs, which points to the presence of other flows that contribute to the IPR value. Since we have already taken into consideration the highly correlated flows, this suggests that weakly correlated flows in cross-correlation matrix can also be dominant flows in the network. In order to identify all the dominant flows of the network, we develop a technique using eigendecomposition and spanning tree. In our technique, we use projection of eigenvalues and eigenvectors obtained from cross-correlation matrix. Creating projection considering all eigenvalues and eigenvectors is computationally expensive. On the other hand, not all the eigenvalues contain useful information. Thus, we consider the largest eigenvalue and associated eigenvector for creating projection matrix as they contain most of the information about the system. Afterwards, we apply spanning tree algorithm to the projection matrix to sort the dominant flows according to their contribution. In the sorted list of flows, we find weakly as well as strongly correlated flows as the dominant flows in the network. We learn that the weakly cross-correlated dominant flows pass through multiple congested nodes and as a result, exhibit weak cross-correlation due to their large delays. The results show that our technique can identify all the dominant flows in a network. We use these results of dependency measurements to construct functional topology of a network. We summarise all the challenges and assumptions that we make in order to measure interdependencies of traffic flows is showing in Figure 1.1

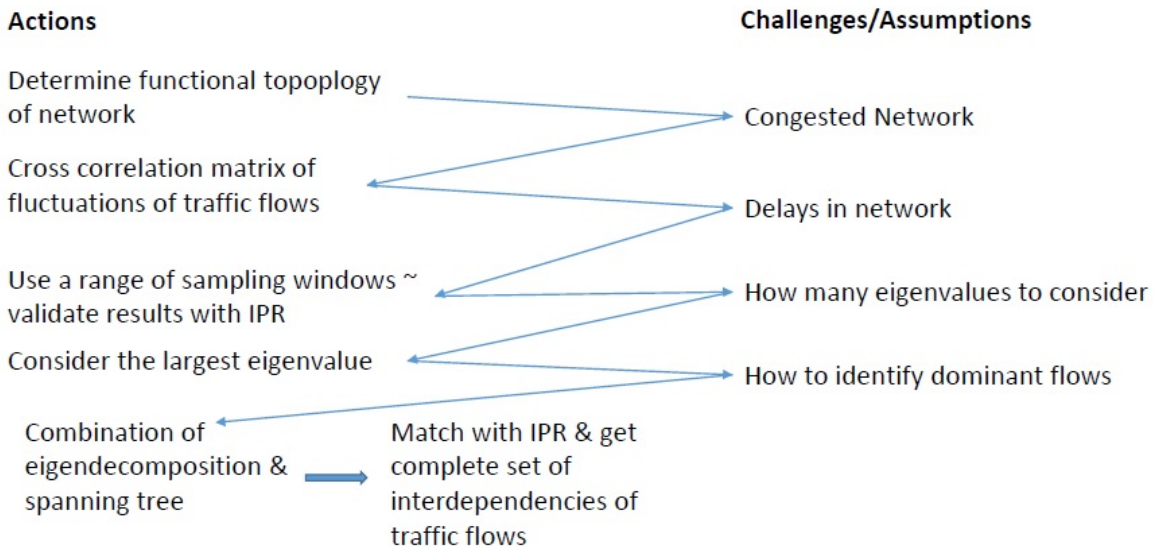


Figure 1.1: Summary of challenges and assumption to measure interdependencies of traffic flows

Based on dependency measurement of traffic flows we develop a novel algorithm to construct functional topology of the network. While constructing the topology, our algorithm considers negative correlation to occur when there is a shared resources between traffic flows. Positive correlations are used for validation purposes. After constructing a topology model, we use binary logic to validate the functional topology. Initially, we find that the newly constructed functional topology represents more resources than physical topology. This occurs because multiple flows share one common resource (in physical topology) which is represented by multiple resources (instead of one resource) in functional topology. This representation of sharing resources by traffic flows happens to be like this as functional topology describes interdependencies of each flow individually. Later, we modify our algorithm with an extended version to resolve this problem. We show that one common resource shared between flows is represented by single resource in functional topology by our refined algorithm. The advantage of this algorithm is that it simplifies the visualization of network connectivity. It only considers the congested nodes (all uncongested nodes are ignored) that create dependencies among traffic flows and presented as shared resources. The shared resources represented in functional topology do not reveal actual node related information. With this benefit, security threats can be easily avoided. This resultant functional topology provides knowledge to edge network operators (e.g. content providers) about the topology of the network and dependencies of flows without any ISPs assistance. Our algorithm uses dependencies among traffic flows in order to construct functional topology of a network. While creating functional topology,

our algorithm deals with many possibilities of representing traffic interactions as a graph by satisfying flows constraints. If it cannot satisfy the flow constraints, then the algorithm cannot represent traffic interaction in an order.Then, it needs to permute different lists (order list and resource list) for the correct resource order that can satisfy all flow constraints. If it cannot find correct resource order, then, it cannot construct functional topology of the network.

Cross-correlation method to create functional topology of the network treats the network as black box meaning that it does not have or require any prior knowledge of physical topology. The method depends on collecting data at the end points (users) and measure dependencies among traffic flows of the network. Then, using the measurement of traffic flows dependencies, it creates functional topology of network. If we consider all the end points of the network and collect the data from all end points, then, noise level would be minimum in our measurement. If we consider a subset of end users of a network, then, our method treats other traffic (that is not destine to our end points) as noise and measure interdependencies of traffic flows based on the data collected from end points that we consider.

We use the NS-2 simulator for generating a range of topologies including local area networks (LAN) to evaluate our algorithm. First, we validate the simulator using several experiments and analyse results such as analysis of traffic generation and queue performance. We run an experiment for 20 days of runtime in NS2 Simulator. It provides sufficient amount of time to construct functional topology of the network. We understand that traffic or infrastructure may change during or after the time duration we consider. Previous studies already show that they were able to detect anomalies of network by measuring interdependencies of traffic flows [6, 11]. Our method is able to detect the anomalies by measuring interdependencies of traffic flows. The dependencies among traffic flows in the network would change if traffic or network infrastructure changes. As the dependency of traffic flows changes, it will reflect on functional topology. We can observe the changes in network in our functional topology of the network.

After validating our simulators, we validate this cross-correlation based approach as a practical method using real data obtained from a wireless sensor network. We apply our algorithm to small network topologies for validation purposes and we successfully construct functional topology of large networks.

1.3 Contributions of the thesis

Cross-correlation method to show connectivity of network has been reported in [12, 13, 14, 15]. Most of the work have been applied in different fields such as stock exchange. To the best of our knowledge, nobody has reported constructing functional topology from end point measurement on any communication network. In this thesis, we propose a novel algorithm to construct the functional topology of the network based on our cross-correlation method. The method has been used to measure interdependencies of traffic flows in communication networks in several previous studies [5, 6, 7, 8]. These studies show that a single snapshot of interdependencies of traffic flows can be captured. In this thesis, we use the method such that it can capture complete set of interdependencies of traffic flows in a network. We use this dependency measurement result as input to our algorithm for constructing functional topology. The key contributions of this thesis are given below.

- A novel algorithm to construct the functional topology of a network from end point measurement has been implemented. The results show that our algorithm can accurately construct functional topology of different networks. In this thesis, we show that a functional topology simplifies visualization of connectivity of network flows by considering only congested nodes, which are represented as shared resources. This simplification of network visualization is beneficial for network operators.

- In order to get a complete set of interdependencies of traffic flows in a network, we use a range of sampling windows for sampling a time series of packet counts according to the maximum delays of traffic flows in our measurements. We show that dependencies of flows with small delays can be captured by small sampling windows and dependencies of flows with large delays can be captured by large sampling windows. Use of different sampling windows according to maximum delays of flows helps us to avoid the effect of delay in network and measure accurate cross-correlation between traffic flows.

- In order to find dominant flows in the network, we develop a technique using eigendecomposition and spanning tree. We find that our technique identifies dominant flows that represent all strongly correlated flows alongside some weakly correlated flows. It suggests that these weakly correlated flows exhibit weak correlation due to large delays. Our technique identifies all the dominant flows

that match with IPR value and provides validation for our cross-correlation analysis.

- We wanted to apply our method to other domain such as WSN. We develop an algorithm called *The Adaptive spatial sampling technique* in wireless sensor network (WSN). We use our method to measure interdependencies of traffic flows as a part of this algorithm. Here, our method is applied on real data obtained from sensor networks to identify dominant sensor nodes of the clusters. Our method can identify important/dominant network elements (traffic flows or sensor nodes) in both wired and sensor networks. Thus, this application of our method in a WSN provides validation for our method to measure interdependencies of traffic flows. The algorithm samples only a few important sensor nodes within clusters for a certain period of time and set the rest of the sensors of the clusters into sleep mode to save energy. We show that our algorithm achieves significant results in data reduction whilst retaining high accuracy in measurement.

1.4 Outline and organization

The aim of the thesis is to describe how to construct the functional topology of a network. In order to construct functional topology, first we need to measure interdependencies of traffic flows accurately. In Chapter 2, we discuss the theory behind measuring interdependencies of traffic flows. We also discuss traditional *Network Tomography* that has been used to discover the topology of a network. We discuss about the advantages, strengths and usefulness of cross-correlation methods for end point measurement. Chapter 3 describes the methodology to measure interdependencies using cross-correlation of traffic flows in small networks. Here, we explore challenges and assumptions that are to be taken into consideration during measurement. In chapter 4, we apply our method on large topologies (including LANs). Here, we show that our method can capture a complete set of interdependencies of traffic flows of the network. We use our method of traffic flow dependency measurement into development of an algorithm in wireless sensor networks. Details of the algorithm are given in Chapter 5. The algorithm produces promising results in terms of data reduction and accuracy. It also provides validation of our method to measure interdependencies of traffic flows. In Chapter 6, we describe our algorithm to construct the functional topology of a network. We evaluate our algorithm on different topologies and discuss advantages of the functional topology. We modify our algorithm at the end of this chapter to simplify the network visualization and compare functional topology with physical topology. In chapter 7, we discuss

conclusions and future work of our thesis. All the publication list can be found in list of publication chapter. Figure 1.2 shows how the publications are focused on different chapters.

Publications	Focus on Chapter
Sabri -E-Zaman and Raul J Mondragon. Functional Topology of Networks. (To be submitted).	Chapter 6: Construction of Functional Topology of Networks
Sabri -E-Zaman, Manik Gupta, Raul J Mondragon and Eliane Bodanese. An Eigen Decomposition Based Adaptive Sampling Technique for Wireless Sensor Networks. In Proc. 39th IEEE Local Computer Networks Conference, Edmonton,Canada (2014)	Chapter 5: Practical Validation of the Methodology usingWireless Sensor Networks
Sabri -E-Zaman and Raul J Mondragon. Measuring Interdependencies and Transitivity of End to End Traffic Flows by Traffic Correlation. In Proc. PhD Forum of 21st IEEE International Conference on Network Protocols (ICNP2013 PhD Forum , Gottingen, Germany (2013)	Chapter 3: Measuring Traffic Flow Interdependencies in Networks
Sabri -E-Zaman and Raul J Mondragon. Inferring Logical Network Topology by Traffic Correlations. In Proc.14th Annual Postgraduate Symposiumon the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, UK (2013)	Chapter 3: Measuring Traffic Flow Interdependencies in Networks

Figure 1.2: List of publications and chapters that papers are focused on

Chapter 2

Background Theory

2.1 Introduction

Topology information is important for efficient network provisioning. This information is rarely available to avoid security risk. To gain knowledge in this respect, we propose to construct a functional network topology. As mentioned in the previous chapter, the term 'functional topology' describes traffic flow dependencies as a graph of interactions. In order to construct this functional topology, we rely on data collected at the end points (destinations), due to unavailability of information elsewhere. We need to consider a method that can measure traffic flow dependencies, and these measurement results will then be used to construct a functional topology – we consider a cross-correlation method in this regard, as it only considers traffic count data obtained at the end points of the network. The method is well-studied and has been applied in many different fields of science to extract meaningful observations. It also has practical importance in biology, social networks and particularly finance, as it can disclose non-trivial relationships among stock portfolios [13, 12].

Two main approaches are used in cross-correlation to extract information. The first approach is known as 'cross-correlation based clustering', and it allows one to obtain clusters that are correlated among themselves. The second approach uses an eigen analysis or principle component analysis to analyse the cross-correlation matrix and investigate its properties. Both approaches are useful for measuring inter-

dependencies of traffic flows. In this chapter, we first discuss the cross-correlation method in relation to measuring traffic flow dependencies in communication networks and other fields of science. At the end of the chapter, we discuss another method, network tomography, which has also been used to discover network topologies. We discuss some accuracy and reliability issues associated with this method, which mean that it can only partially discover network topologies.

2.2 First approach: cross-correlation based clustering approach

Cross-correlation based clustering has been used to understand network structures in financial data. This approach was introduced for the first time in [16]. Previous studies show that the approach is useful for modelling financial entities such as stock portfolios [13, 14, 15, 12]. First, they collect a set of time series data on financial market stocks, in order to generate a cross-correlation based network. This approach always generates a completely connected network, so if there are n numbers of time series data, then it constructs $n \times n$ connected networks. For example, if we have six time series of data, it constructs 6×6 fully connected networks. Figure 2.1 shows a fully connected network.

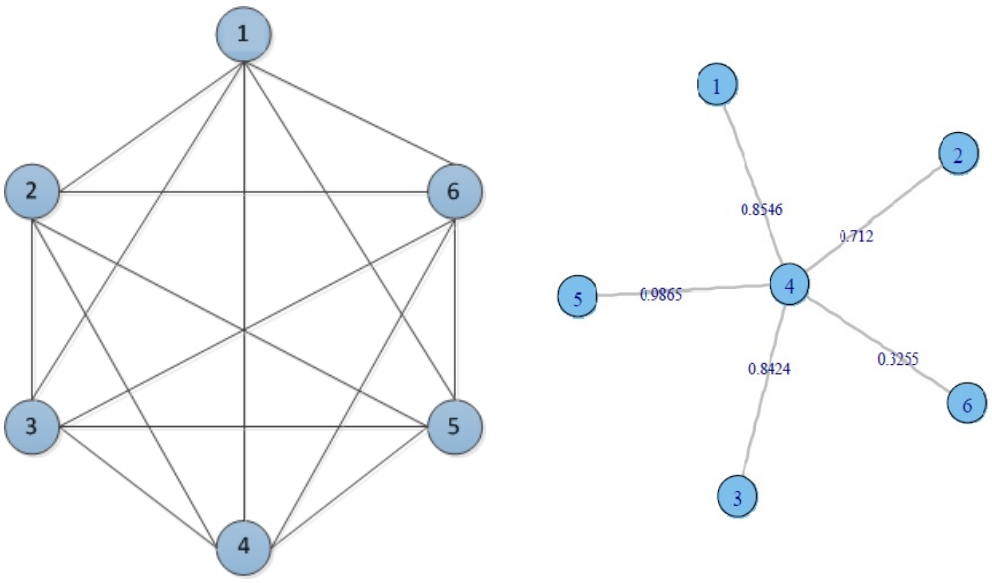


Figure 2.1: a) Fully connected mesh network for six sets of time series data b) Spanning tree for 6×6 networks

In the network graph, each pair of nodes is connected with weights determined by cross-correlation coefficients. The computation of a fully connected network is expensive, and often many node pairs,

such as weakly correlated links, will not provide useful information about the topology. With the help of different filtering techniques, though, these weakly connected links can be removed. A minimum spanning tree (MST) is one of these filter techniques and has been used on financial data [16, 12]. The MST considers only highly correlated links, in order to connect all network vertices but without forming a loop. In figure 2.1 b), we show the MST of a network with six time series datasets, which only considers the highly correlated link. A planar maximally filtered graph (PMFG) is another filtering technique employed to remove weakly correlated links in a network. The PMFG also produces networks, with most correlated links with constraint being represented on a plane without any edge-crossing [13, 14, 15].

The cross-correlation-based method, along with filtering techniques, produces a well-connected graph of a network. After producing the graph, researchers focus on the fact that network graphs change according to different time horizons or sampling windows of time series data. They observe that the cross-correlation coefficient decreases if the time horizon or sampling window of the time series data decreases due to the *Epps effect* [17, 13, 12]. The topological properties of an MST obtained using different time horizons were investigated for the first time in [12]. Bonanno et al. show that the hierarchical organisation of a set of stocks is detected when the time horizon changes – a result that helps to improve the modelling of stock portfolios. Later, the effect of using different time horizons was shown in [13, 14, 15], using PMFG. Their results indicate that selected stocks progressively form a hierarchical system structure in line with the changes in the time horizon. These results have an important impact on modelling financial entities, as they demonstrate that the cross-correlation method is able to model a subset of stock portfolios using a single time horizon. With the help of a range of time horizons, the method is able to model financial entities completely.

Cross-correlation clustering is really useful in modelling financial entities. Previous studies show that the method can filter weakly correlated links using different sampling techniques. It is also able to detect the hierarchical organisation of a set of stocks using different sets of time horizons. Filtering weakly correlated links and using different time horizons in time series data are particularly useful methods for our current research purposes, namely measuring interdependencies of traffic flows. It means that using different time horizon in time series data will help us to measure complete sets of dependencies among traffic flows. Filtering weakly correlated links involves filtering out irrelevant data from datasets and increasing measurement accuracy. However, the approach does not describe whether or not weakly

correlated links contain any useful information. On the other hand, some factors that influence the calculation of cross-correlations may affect the accuracy of the measurement. For example, if there is a delay in the network, then it is possible that cross-correlation coefficients will become insignificant due to this delay. In this case, filtering out all weakly correlated links means that we may lose important information about the network topology. Thus, we need to analyse further weakly correlated links, before we remove them from our measurements.

2.3 Second approach: principle component analysis or eigen analysis of the cross-correlation matrix in a communication network

The second cross-correlation approach deals with the principle component analysis or eigen analysis of the cross-correlation matrix. This approach has been applied in communication networks. Previous studies show that it has a unique way of measuring interdependencies of traffic flows within a complex and noisy network environment [5, 6, 7]. For this reason, the approach has credibility and usefulness for measuring interdependencies among traffic flows found in modern and complex networks. It treats the infrastructure between the points from which the traffic is obtained as a black box [7]. The benefits of this black box approach are that it is application-free and there are less overheads in the measurements. In this section, we discuss previous work demonstrating how the approach separates system-specific traffic interactions from random interactions. The random matrix theory is an associate tool that has been used for many years to filter out noise. Here, we describe the work done so far in communication networks using this approach.

Berthelmy et al. [5] introduce a traffic correlation-based method. Their motivation is that the existence of collective behaviour, such as congestion, in a network suggests that there is some form of correlation between different network elements and their connections. Thus, it is important to measure and quantify the correlation, in order to understand the behaviour of the system. In their paper [5], they study a wide area network (WAN), referred to as *Renater*, which consists of 26 active routers and 650 connections. They use the random matrix theory to separate system-specific interactions from random interactions and subsequently find large-scale correlation in the system. From the experiment and analysis, they also show that the traffic correlation method reveals the existence of active centres that exchange information with a large number of routers, thereby inducing correlation between corresponding connections.

As mentioned earlier, Berthelmy et al. first applied the cross-correlation method to measure and quantify correlations among traffic flows in networks. The results show that high correlations exist in internet traffic due to the presence of network congestion, and they illustrate how to find the number of dominant flows in the network by using the inverse participation ratio (IPR). However, there is no clear indication as to which traffic flows are dominant. They use a single time horizon or sampling window ($\tau = 300 seconds$) in time series data to measure cross-correlation in the flows, but the use of a single sampling window can provide a single snapshot in this respect. We need to use different sampling windows to measure correlations, in order to provide a complete set of flow interdependencies. In spite of a lack of some information, the paper nevertheless gives direction for applying the traffic correlation-based method to find the correlations of a system.

Rojkova et al. [6, 7] investigate the traffic behaviour of inter-connected backbone routers and the virtual local area network (VLAN) of University of Louisville, by using the traffic cross-correlation method. They also use the random matrix theory to filter out noise from genuine correlations. In the paper [6], they accomplish three tasks in relation to traffic analysis: they identify uncongested traffic, detect system-specific correlated traffic and locate anomalies in network traffic interactions. Furthermore, they show the stability of correlations among flows, by using an overlap matrix which compares the eigenvectors of time period t and time period $t + \tau$.

The analysis only shows changes in eigenvectors obtained from cross-correlation matrix, not the reason why they actually change. In [6], they use projection of eigenvector components and growth of traffic to identify dominant flows in the network. They identify three types of time series as dominating factors, but they do not provide any indication as to whether or not these flows are related to each other.

Rojkova et al. [7, 9, 10] present a comparative analysis through which they identify the state of traffic. They generate simulated congested and uncongested traffic by using an OPNET simulator, and then they compare these simulations with real traffic by applying the traffic correlation-based method. They also propose an important phenomenon that describes the IPR level of the spectrum within the boundaries of uncorrelated data and which can be used as a congestion level indicator. They show that IPR values within the bounds of an RMT deviate from the control line (IPR values of uncorrelated data), while IPR values can identify the number of dominant flows in low congestion network states. In a later section, they justify this phenomenon by finding dominant flows in the network, and they also show that the

cross-correlation-based approach is able to detect anomalies in a network. In other papers [7, 9, 10, 11], they extract the features of network traffic by applying the traffic correlation approach and the random matrix theory, in order to detect any changes in stable traffic dynamics.

All of these papers [7, 9, 10, 11] use a single time horizon or sampling window in time series data to measure correlations between flows. Single time horizons or sampling windows provide good results for the feature extraction of network traffic flows, but they also only provide a subset of interdependencies of traffic flows in the network. Thus, it is not enough to use a single sampling window in this respect, and so we need to use different sampling windows according to delays in traffic flows, to be able to acquire a complete set of interdependencies of traffic flows.

The correlation between flows changes with time in a communication network. This phenomena is described by Jian Yuan et al. [18], who investigate the spatial-temporal characteristics of traffic in a large network. They show that correlations in traffic flows change at different times in different domains of the network. Their experimental results suggest that the cross-correlation method provides a possibility to observe shifts in congestion in a network. The paper by [18] shows that their proposed method is really useful for monitoring network-wide traffic in time and space. However, they do not consider delays and their effects on measurements. In their methodology they use traffic flow data count as a means of calculating cross-correlation between traffic flows. Conversely, we use traffic flow fluctuations (relative traffic growth) in this regard, which removes any data volume bias in our measurement.

Delayed correlation matrix analysis can extract meaningful information from systems. A study of the delayed correlation matrix, by using time lagged correlation between time series, describes the evolution of correlation patterns and extract features in a system [19, 20], and the analysis has been applied in electro-psychological time series in relation to brain response data [21], the relocation of earthquake [22] and financial portfolios [23, 19]. Victoria Rojkova et al. [8] apply the delayed correlation matrix to observe the evolution of network traffic correlation. The analysis is useful in identifying the congestion level of network traffic systems and understanding the correlation of patterns between traffic flows and their evolution over time. The resulting matrices show that the largest eigenvalue and the corresponding IPR values oscillate between two time periods. This finding fits well into the long-range dependence properties of network traffic. In their paper, [8] they argue that the analysis of the eigenvalue spectrum can be an indicators of long-range dependency and self-similarity properties.

The paper shows that the delayed correlation matrix has a complex eigenvalue spectrum, but the general properties of this matrix are unknown so far. For measuring interdependencies of traffic flows, we cross-correlation matrix that has real eigenvalue spectrum. As a result, we do not consider the delayed correlation matrix when measuring interdependencies of traffic flows.

The traffic correlation-based method is useful for monitoring spatial-temporal traffic patterns in a network. Monitoring traffic patterns in space and time is important for improving the performance of a network, and it also provides useful information about network management and control. Client-server-based traffic and the P2P traffic pattern have been analysed using the traffic correlation method [24, 25, 26]. The analysis shows that the traffic correlation-based approach can capture traffic patterns at a macroscopic level [24]. The authors use principle component analysis and the random matrix theory to extract meaningful interactions from noise. Furthermore, Jia et al. [25, 26] show that they can extract key information from traffic patterns using 10% of nodes as observation points. These observations are important to our approach, as we also analyse subsets of whole networks and treat other background traffic as noise. It shows that the cross-correlation method is useful for traffic monitoring purposes, although the method does not infer topological information from the measurements.

We have used different theories from different field of science. We highlight the theories and domain of science that we borrow for our methods are given in figure 2.2.

Theory	**Domain**	**Attributes**
Cross-correlation based clustering	Finance	1. Provides idea of using different sampling windows 2. Use of Spanning tree for filtering 3. Observe the 'Epps effect'
Principle component analysis or eigen analysis of the cross-correlation matrix	Finance/ Communication Networks/ Biology/Physics	1. Analysis of cross correlation matrix 2. Use of Random Matrix theory as filter 3. Use of IPR
Inverse Participation Ratio	Localization theory	1. Determine the number of dominant factors of the system

Figure 2.2: Highlighting theories borrow from different field of science for our method

Figure 2.2 shows that the list of theories, the domains of science that we borrow the theories and some attributes of those theories. Cross-correlation based clustering is heavily used in finance especially for creating stock portfolios. It observes the epps effect while calculating cross-correlation among different

stocks. It uses spanning tree to filter our weak cross-correlation links among stocks. It also initiates the idea of using different sampling windows in the measurements. This theory is really important for method as we get the motivation of using different sampling windows in our measurements. By using different sampling windows, we can actually avoid the effect of delay from our measurements. The second theory provides us background on how to analyse a cross-correlation matrix. It originates from physics and heavily used in finance, biology and also in communication networks. The theory is useful for us as it provides guideline on cross-correlation matrix analysis and filtering redundant data. Inverse Participation ratio was introduced in localization theory. It determine the number of dominant component of the system. It derives from analysing eigenvector computed from cross-correlation matrix. In our case, it explains how many dominant flows are there in the network. It provides us validation for result obtained from cross-correlation matrix.

2.4 Cross-correlation method applied in different fields of research

The cross-correlation method has a wide range of applications. Due to its unique way of extracting meaningful information from a system, it has been used in fields such as communication networks, finance, biology, criminology, etc. This method is cost-effective, meaning that it does not need to employ external resources (for example, sending active probes for network measurements) for measurement purposes. The method is based on system data and extracting information from this dataset. It uses the random matrix theory to remove noise from the dataset, which in turn increases the accuracy of the measurement. The random matrix theory, which is used as a tool to separate noise and extract meaningful information, was developed by Wigner in 1955 [27] in order to study the complex energy levels of heavy nuclei [6]. The method provides a significant amount of information about the interactions and energy levels of nuclei [28], and it is useful for separating system-specific interactions from random interactions, as detailed in [29, 30, 31, 32].

The approach uses a unique methodology to separate random interactions from non-random interactions. Thus, it has been applied in criminology. Jameson L. Toole et al. produced some significant results by analysing the criminal records of the city of Philadelphia, in order to identify patterns in criminal aggregation that could help to predict and prevent crime in the future [33]. The approach has also been applied in in the field of Genetics. For recent genome technology, it is possible to generate a huge amount

of gene expression data for the entire genome, but it is really hard to correlate patterns in these data – the cross-correlation method is used in this instance [34]. Cross-correlation also reveals correlations in trade markets in the modern economy [35, 36, 37, 38], by helping to predict market conditions and dynamics, which it achieves by revealing the collective behaviour of stock markets. The approach has also successfully revealed correlations among stocks [39, 40]; Parameswaran Gopikrishnan et al., for instance, analysed stock markets at two different times to check how correlations among these stocks evolve over time [41, 42].

The cross-correlation-based method has a strong background and history. A it can extract specific information from random interactions (noise), it is useful for understanding the collective behaviour of a system. In different disciplines, including networks, researchers investigate correlations in fluctuation factors (for example, prices, criminal data or traffic flows). In networks, the method is successfully able to show the correlation between two paths and helps to identify congested (localised points) areas. In economics and finance, the approach has been employed to calculate successfully the correlation between different stock markets at the microscopic level. It can therefore be said that the acceptability of this approach is becoming universal. In all of the papers mentioned above, it has been shown that the cross-correlation method can extract meaningful information about a system, but they do not discuss whether or not the amount of information they discard from the measurement is actually useful. On the other hand, they always use a single time horizon in their time series data, which, according to our understanding, provides a single snapshot of the correlation state for system elements. These correlation states change if the time horizon changes in time series data. Consequently, we need to consider all of these aspects while measuring interdependencies of traffic flows in the network. There are also other methods that can provide useful information about network topology. In the case of communication networks, for example, network tomography is a well-known method that deals with discovering a network topology, and we will therefore discuss it in the next section.

2.5 Network tomography

Network tomography was introduced by Y Vardi [43]. The original discussion centered on estimating an origin-destination traffic matrix. Network tomography deals with network topology and its properties. There are two methods in network tomography – passive and active. Passive tomography is

node-oriented and collects packets and network information passively, by monitoring agents located in different devices (routers, switches, etc.) in a network [44], whereas active tomography is a path-oriented method that collects information on different connections and latency by sending active probe packets [44].

In passive tomography, the aim is to collect every bit of information about the distribution of traffic flows, from origin to destination. To characterise the pattern of the information obtained from the origin-destination (OD) matrix, one needs to take measurements at different time scales, as different sources have different variations in sending data. To collect data, different agents are located in different devices. Due to the huge volume of data, the number of packets is considered to be counted only. A routing matrix is used in this approach to infer information on different network nodes, but it changes over time [45] and therefore makes it a difficult inverse problem, although Soule et al. [46], who investigate routing matrix changes in their paper, actually address this problem. The modification of traffic matrix estimation incorporated with temporal consideration and the aim is to observe the temporal evolution with time. They propose a state-space model which identifies the spatial dependence of OD flows and captures traffic system noise [47]. Later, Liang et al. [48] modify the model to allow the occasional direct measurement of selected OD flows that aid in calibrating parameters within the model. Vardi [43], in the meantime, deals with a traffic matrix that is derived from origin-destination traffic counts. He considers general topology and adopts a simple Poisson model for origin-destination (OD) traffic byte counts. Later, Cao et al. [49] consider a Gaussian model for OD traffic counts. These simple assumptions do not hold for a real network. The *Tomogravity* model has been proposed to address this problem [50, 51]. The model assumes independence between source and destination [44]. Traffic matrix estimation is really useful for network operators, especially for network planning, but the nature of the problem requires imposing many assumptions for modelling in many cases [44, 52], which has a negative impact on the accuracy of results [52].

As mentioned earlier, active tomography deals with estimating the quality of service parameters (loss, delay etc.) relating to links in a network. This information is helpful for characterising the performance of a network [44]. In active tomography, a probe, which can be unicast or multicast, is used to estimate different links' parameters. Unicast probing cannot estimate all of the parameters at the link level, so in this case multicast probing is useful for overcoming the problem. In some networks, the option to use multicast probing turns off due to security reasons. In this situation, Tsang et al. [53] propose back-

to-back unicast probing, where unicast probes are sent within nanoseconds. In traditional multicast probing, probes are sent to all receivers in the network, in order to infer the logical multicast topology of the network [54, 55, 56]. Multicast probes can successfully infer the parameters of different links, but the disadvantage of this scheme is its inflexibility and the fact that it can overload the network. To overcome this problem, Xi et al. [57] and Lawrence et al. [58] propose a flexible class of multicast probing. In this scheme, they send probe packets to a different region of the network at different times and at varying intensities. Inferring loss rate and delay are the key factors in active tomography. Methods for inferring loss rates have also been studied [59, 60], and these are based on a simple assumption that was addressed later by the expectation-maximisation (EM) algorithm [57, 61, 60]. In respect to delay distribution, multicast probing was initially used [58, 62], but it can only be applied to small networks. A different approach has therefore been proposed for larger networks [48, 58]. Inferring parameters such as delay and loss of network links can be done under strong simplifying assumptions and in a static network. However, instead of doing that, [63] suggests a new algorithm that infers the probability of each link being congested. They consider Boolean inference and check the accuracy of the Boolean inference algorithm.

Traceroute is one of the most popular tools used in network tomography. It has inaccuracy and reliability issues to infer the topology [64]. It depends on ICMP echo responses where always there is a possibility of generating errors. It is possible that it might infer false links of the network. For a false inference it could treat per flow load balancing to successive distinct flows. Matthew Luckie et al. analysed this issue and showed that 2.71% of router links and 0.76% of traceroute graphs were wrong in their experiment, which consisted of 365k destinations [65]. A *third party address* is one element behind inaccuracies in traceroute, and so there is ongoing work in relation to identifying them accordingly [66]. On the other hand, traceroute can work properly by assuming the network is functioning properly and that different network elements are prepared to cooperate [60]. Due to malicious attacks (for example a denial attack), many network elements opt out the cooperation option [60, 44], and so in this case traceroute cannot discover the complete topology. We acknowledge that tomography can measure some quality of service parameters, such as delay and loss for particular links, but there are many accuracy and reliability issues that have to be considered.

2.6 Our approach for measuring interdependencies of traffic flows in a network

Network tomography does not provide any information about traffic flow dependencies in networks. In addition, it is difficult to provision network using a method that can only partially infer network topology. Thus, we consider the cross-correlation method for measuring interdependencies of traffic flows and constructing a functional topology. Before using the cross-correlation method for our measurement purposes, we also consider both of the approaches that use this method. Cross-correlation clustering works on the idea of using different time horizons in time series data and extracting meaningful traffic flow dependencies. We do not use any filtering technique (MST or PMFG) to construct networks from a cross-correlation matrix, as they only consider the strongest correlated links to connect vertices. However, MST is used for identifying dominant traffic flows in the network that we discuss later. The filtering techniques used by this approach give us an idea about removing irrelevant data from the measurement process. We use cut-off cross-correlation (threshold), verified against a null hypothesis, to remove weak links from our network while constructing a functional topology. Thereafter, we use a second approach for validation purposes, namely an eigen-analysis of the cross-correlation matrix and the inverse participation ratio (IPR). The IPR value simply describes the number of dominant flows in a network [5, 6] – it does not describe which are the most dominant. We use eigendecomposition to project the largest eigen values and eigenvectors, in order to identify dominant flows. Later, we use a spanning tree to rank all of the dominant flows according to their contribution. We understand that both approaches are highly important for our measurement process, so we adopt parts from both approaches and create our own methods to measure interdependencies of traffic flows and to construct a functional topology.

2.7 Summary

In this chapter, we discuss and compare different theories involved in measuring interdependencies of traffic flows in a network. We discuss correlation clustering and the eigen-analysis-based approach that uses cross-correlation and provides system-specific information. We observe that both approaches are really useful for measuring interdependencies of traffic flows in a network, and so we adopt parts from both approaches and modify them according to our requirements. We also discuss network tomography, which is used to discover network topology. We know that tomography actually deals with network node-related information, which is rarely available nowadays, due to access issues. We learn that net-

work tomography provides us limited information about network topology because of the accuracy and reliability issues in the technique. It is difficult to provision a network efficiently with this limited information about network topology. Thus, we propose to construct functional topology that can describe dependency of traffic flows and help to provision network efficiently.

In the next chapter, we describe our method to measure interdependencies of traffic flows in detail. We also discuss a number of challenges and assumptions that we make during our measurement, based on some experimental results.

Chapter 3

Measuring Traffic Flow Interdependencies in Networks

3.1 Introduction

In this chapter, we discuss the method, challenges and assumptions involved in measuring the interdependencies of end-to-end traffic flows in networks. We create network with small topologies and apply our method on these topologies with different scenarios. Then, based on the results obtained from this empirical approach, we learn how to overcome challenges to measure interdependencies of traffic flows and make some assumptions for further analysis and experiments. This chapter is the keystone of the thesis as we build up our method using different experiments using empirical approach. We apple our method on larger topology to check scalability in the next chapter. So, next chapter can be considered as extension of this chapter.

In this chapter, first, we start by collecting data at the end points (destinations), creating time series data using time intervals, calculating cross-correlation coefficients and constructing a cross-correlation matrix. While calculating the cross-correlation between network traffic flows, we observe that a few things such as delays in traffic flows an the congestion state of the network can affect our measurement. Thus, we discuss this issue and find that cross-correlation reaches to its peak if the traffic flows pass through a congested node buffer. We create different levels of congestion in the network and measure cross-correlation among flows to show how congestion can change our interdependency measurements.

We then make an assumption, based on this result of some experiments on our simulator and information from communication network industry, that we need a certain level of congestion in a network while measuring the interdependencies of traffic flows. In this thesis, we assume that 70% of utilization for of buffer of node as threshold congested network. While measuring the cross-correlation between two pairs of traffic flows (one has a larger delay than the other), we discover that their coefficients are different, even though they pass through the same congested level. To determine peak cross-correlation coefficients (minimum or maximum), we use different sampling windows in time series data. We find minimum cross-correlation coefficients for two pairs of traffic flows, using two different sampling windows. Then we check the end-to-end delay distributions of the traffic flow to establish a relationship between the delays and the sampling windows. We establish that we acquire peak cross-correlation coefficients of traffic flows if we take sampling windows according to maximum flow delays. As there are many flows with many different delays in the network, we find the need to use different sampling windows to capture a complete set of dependencies. Thereafter, we compare our results against a null hypothesis, to distinguish actual traffic interaction and random interaction. From this comparison, we determine a cut-off cross-correlation coefficient that filters out noise from our measurement. Then, we validate our results with an inverse participation ratio (IPR) that gives the number of dominant or important flows in the network. Details of the IPR are provided later in the chapter. While matching the cross-correlation coefficients and IPR value, we feel the need to identify dominant traffic flows in the network, for which we propose a technique. At this point, we can see that our method can calculate minimum or maximum cross-correlation coefficients, using different sampling windows, and we are able to identify dominant traffic flows to validate our results with IPR values. We use two network topologies for our experiments, in order to evaluate our method. In the next section, we discuss our method, starting with how to collect data, create a time series, construct a cross-correlation matrix, etc. We also discuss how time intervals or sampling windows of time series data uncover peak correlation (minimum or maximum) coefficients of traffic flows suggested by their delays.

3.2 Cross-correlation matrix

In all of our experiments, we collect data at the destination (end points) of each traffic flow. We count the packets from each route to the destination f_{ij}(connection from i to j) with equal time intervals(sampling windows), and afterwards, we measure fluctuations in the traffic, termed the 'growth rate' g_{ij}, which is

the logarithmic ratio of two successive time series counts [5, 6, 7, 9]. Growth is expressed as

$$g_{ij} = \ln[\frac{f_{ij}(t+\tau)}{f_{ij}(t)}] \tag{3.1}$$

The reason behind transforming time series data for a number of packet counts to relative increments of successive counts is to make our measurement independent of the data volume [5, 6, 9]. Next, we calculate the cross-correlation coefficients of two traffic flows, the equation for which is

$$C_{ij} = \frac{avg(g_{ij}g_{kl}) - avg(g_{ij}) * avg(g_{kl})}{\sigma_{ij}\sigma_{kl}} \tag{3.2}$$

We know that there are two types of cross-correlation coefficient – positive and negative. Negative cross-correlation occurs when two flows share one common resource. If two flows pass through the same node buffer (such as a router), one has to wait while the other tries to pass through the node. This in turn creates negative cross-correlation coefficients between the flows. Positive cross-correlation means transitivity of traffic flows whereby two pairs of traffic flows share one common flow, which creates transitivity among these three traffic flows.

In order to get accurate cross-correlations, we need to check autocorrelations of time series data. Autocorrelated time series can introduce inaccuracy while calculating cross-correlations. In our case, we need to check autocorrelation of traffic growth, g_{ij} to determine accuracy of cross-correlations. In appendix C, we mathematically show, if autocorrelation coefficients of lag 1 of a time series is around -0.5, then we consider the data as non-autocorrelated. We check autocorrelations of traffic growth, g_{ij} of traffic flows from our experiments. We find the autocorrelation coefficients of lag 1 of traffic growth is -0.5 in appendix C.

Delays in traffic flows have an impact on dependency measurements. To understand this issue in relation to calculating cross-correlations, we use different sampling windows in the time series data. In this section, we show that we need to use sampling windows according to maximum delays of traffic flows, in order to produce peak cross-correlation coefficients. To collect data and calculate cross-correlation coefficients, we create a simple network that consists of four traffic flows in NS2. We run the experiments for 20 days simulator run time. We run the same experiments 30 times to check consistency of

experimental data.

In this experiment, we set the utilisation ratio for all queues at around 80%, to capture correlations among the flows. In the experiments we use Poisson distributed traffic. We use other traffic such as Pareto distributed traffic in the experiments of chapter 4. The network topology for this experiment is given in Figure 3.1.

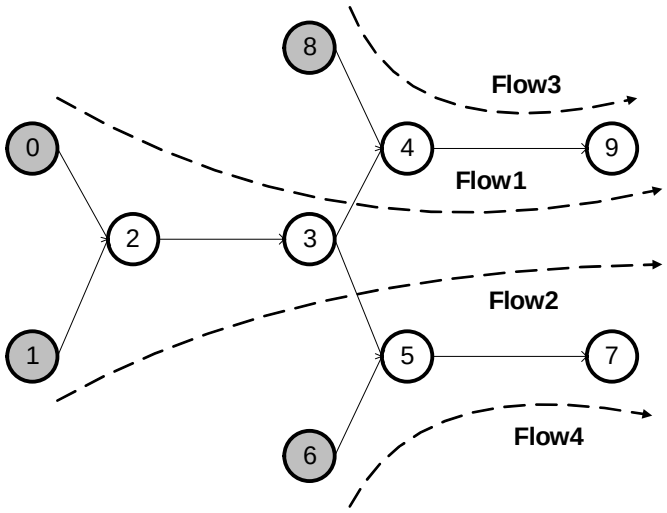


Figure 3.1: Network Topology

This experiment runs for about 20 days (simulator time). The central limit theorem is applied, in order to determine experiment time. We set a buffer utilisation ratio to more than 80% of nodes 2, 4 and 5, by generating Poisson distributed traffic with a high rate from sources. We collect the flow data (packet counts) at the end points and form a time series using time intervals (sampling windows) of 2 seconds. Then, we calculate cross-correlation coefficients between different traffic flows. Two pairs of cross-correlation coefficients are given below.

Cross-correlation between traffic flows	cross-correlation coefficients
Traffic flows 1 & 2	-0.0125
Traffic flows 1 & 3	-0.8634

Table 3.1: Cross-correlation coefficients of two traffic flow pairs

From the two pairs of cross-correlation coefficients shown in Table 3.1, we can see significant differences for these flows. As both of traffic flow pairs pass through a heavily congested buffer, we expect to

see minimum correlation coefficients (less than - 0.8) between traffic flows. To find out the root cause, we check the delays.

Traffic flows	Average delay	Maximum delay
Traffic flows 1	10 sec	120 sec
Traffic flows 2	12 sec	125 sec
Traffic flows 3	4 sec	10 sec

Table 3.2: Delays of traffic flows 1,2 and 3

We can see delays for traffic flows 1, 2 and 3 in Table 3.2. The maximum delays for traffic flows 1 and 2 are around 120 seconds, whereas traffic flow 3 has a maximum delay of 10 seconds. To investigate further the minimum cross-correlation coefficients of traffic flows 1 and 2, we change the time intervals of our time series data. We set time intervals to 100 seconds, create a time series and calculate the cross-correlation coefficients.

Cross-correlation between traffic flows	cross-correlation coefficients
Traffic flows 1 & 2	-0.3109
Traffic flows 1 & 3	-0.7034

Table 3.3: Cross-correlation coefficients of two traffic flow pairs with time intervals of 100 seconds

We find that the cross-correlation coefficients for traffic flows 1 and 2 decrease to their minimum, while in the case of traffic flows 1 and 3 they increase towards zero, as shown in Table 3.3. To understand the effect of delays, we use different time intervals (sampling windows) in the time series data and calculate the cross-correlation coefficients. We also consider and map their average and maximum delays during this measurement.

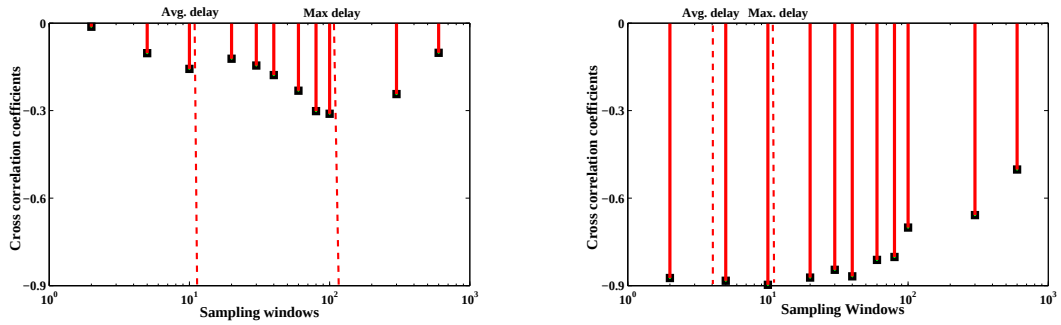


Figure 3.2: Comparison of the cross-correlation coefficients of two pairs of traffic flows, with sampling windows ranging from 2 to 600 seconds

Traffic flows 1 and 2 share common resources (node 2 and 3) among themselves, which means that one flow has to wait while another flow passes through the buffer. For this reason the correlation coefficients should be negative. Flows 1 and 2 pass through two different buffers before reaching their destination. These buffers are congested and responsible for changing the delay distribution of these flows. From Figure 3.2 we can see that the minimum correlation coefficient is around -0.3 with sampling windows of 100 seconds. So, if we take the sampling windows at around 100 seconds, we can then find the minimum correlation coefficient, which we are expecting between flows 1 and 2.

Flows 1 and 3 are directly connected to each other. Flow 3 follows the shorter path in the network. The queue for node 4 is heavily congested, due to the high traffic rate of flows 1 and 3 passing through node 4. From Figure 3.2, we can see that the minimum correlation coefficient is -0.87, if we take the sampling window at around 10 seconds.

We expect minimum cross-correlation coefficients, as traffic flows share resources among themselves. We produce minimum cross-correlation coefficients for two pairs of traffic flows (1-2 and 1-3), using different sampling windows. To understand why we have minimums in this case, we check their end-to-end delays. We observe that traffic flows 1 and 2 have greater delays than traffic flow 3, and we also find that the minimum cross-correlation coefficients for traffic flows 1 and 2 are around 100 seconds, while for traffic flows 1 and 3 they are around 10 seconds, as shown in Figure 3.2. Table 3.2 suggests that the maximum delays in traffic flows 1 and 2 sit at around 120 seconds, and traffic flow 3 has a maximum delay of around 10 seconds. From this experiment we observe that if we use sampling windows around the maximum delay limit, then we find maximum or minimum cross-correlation coefficients. If we calculate the cross-correlation coefficients of two flows, where one has a significant delay and the other has a small delay, then we take the sampling window according to small flow delays. In this case, if we consider a sampling window according to the large delays, then it normalises the fluctuation of traffic growth and we do not get a peak cross-correlation coefficient. In networks, different flows have different delays, so we need to use different sampling windows according to maximum traffic flow delays, to calculate accurate cross-correlation coefficients and to discover a complete set of interdependencies among the traffic flows. We check our observations about sampling windows and maximum delays in later experiments in this chapter.

After calculating the cross-correlation coefficients of the time series, we construct a cross-correlation

matrix **C**. If the network has n traffic flows, then we construct a $n \times n$ cross-correlation matrix **C**. The cross-correlation coefficient between two flows describes the strength of dependency. In a network, some traffic flows exhibit strong correlations or dependency, and some exhibit weak correlations, the latter of which often occurs due to the presence of noise or random interactions. In our measurement, we need to separate noise from the actual correlation, to improve the accuracy of our measurement. As a result, we need a cross-correlation coefficient threshold that works as a filter to remove noise from our measurement. In the next section, we discuss this threshold and deduce how to determine it. We call this threshold the 'cut-off cross-correlation coefficient'.

3.2.1 Cut-off cross-correlation coefficient

So far, we have discussed our methods for calculating cross-correlation coefficients and constructing a cross-correlation matrix. While calculating cross-correlation coefficients, we observe different coefficient values, ranging from -1 to 1, but it is possible that noise may find its way into our measurement in this instance. Here, noise means the random interaction of traffic flows. To measure accurately the dependency of traffic flows, we need to filter out random interaction while constructing a cross-correlation matrix. To do so, first we randomize our data and create a spectrum of correlation coefficients relating to random interactions. Consequently, if any correlation coefficient falls out of this range, we can consider it as an actual traffic interaction; otherwise, we consider it as random. We call the limit of this spectrum the 'cut-off cross-correlation coefficient'.

To create the spectrum, first we take time series data (experiment mentioned in the previous section) and transform them into traffic growth g_{ij} for a certain sampling window. Then, we randomize each traffic growth g_{ij} vector, using the standard MATLAB library, and calculate the cross-correlation coefficients. We iterate the same process 30 times and calculate the cross-correlation coefficients for sampling windows ranging from 2-600 seconds for all four traffic flows, following which we collect the results and plot them with cross-correlation coefficients deduced from the cross-correlation matrices for all sampling windows.

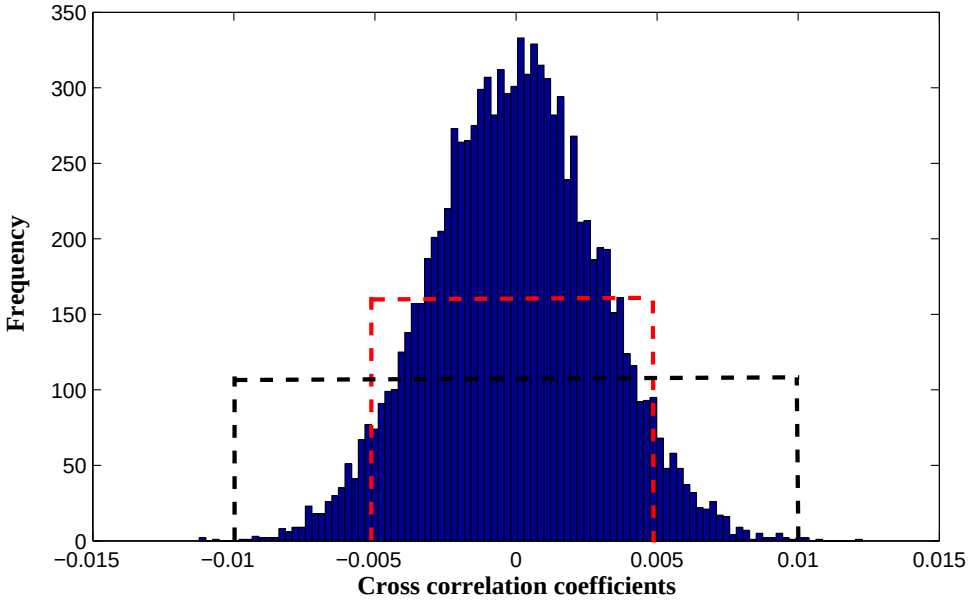


Figure 3.3: Cross-correlation coefficients of flows 1 and 2, with different sampling windows

The standard deviation for randomised correlation coefficients is $\alpha = 0.005$. We consider our cut-off correlation coefficient is $2\alpha = 0.01$. In our experiment, we use high network traffic rates, which create correlated traffic flows. Thus, most of the correlation coefficients fall out of spectrum for this experiment.

So far, we have discussed our method for calculating cross-correlation coefficients between traffic flows, how sampling windows suggested by the maximum delay can provide minimum or maximum cross-correlation coefficients and how to determine cut-off cross-correlation coefficients to filter out our random interactions. In the next section, we introduce the inverse participation ratio (IPR), in order to validate our results. The IPR provides us with the number of dominant flows in the network. It is a good candidate for validating our results, as we can match a number of strongly correlated pairs with the IPR number.

3.3 Inverse participation ratio (IPR)

An inverse participation ratio (IPR) determines the number of dominant flows (important flows) in a network. To calculate the IPR, we deduce eigenvalues and eigenvectors for the cross-correlation matrix **C**. Then we calculate the IPR for each eigenvector, using the equation below:

$$I^k = \sum_{l=1}^{n} [u_l^k]^4 \tag{3.3}$$

where u_l^k is the kth eigenvector, with l number of components [5, 6, 7, 9].

Usually, the largest eigenvalue contains most of the information about the system. Thus, we calculate the IPR using the largest eigenvalue of the cross-correlation matrix. The IPR describes only the number of dominant or important flows in the network – it does not provide any information about "which" are the dominant flows, so, in order to identify these, we develop a technique that uses eigen-decomposition and a spanning tree. The technique is detailed in the next section.

3.4 Identifying dominant traffic flows in a network

Cross-correlation is a bivariate analysis and measures the strength of dependency between traffic flows. We need to investigate this a bit further, to identify dominant network flows, using a cross-correlation matrix. First, we derive eigenvalues and associated eigenvectors from a cross-correlation matrix. Here, random matrix theory can be applied as a filter to remove eigenvalues that actually represent noise[5, 6]. After deriving the eigenvalues and eigenvectors, we create a projection for all of them. This process is called 'eigen-decomposition', and while evaluating it, a cross-correlation matrix, C, can be written as [67, 68]:

$$Cv = \lambda v \tag{3.4}$$

where λ is the eigenvalue and v is the corresponding eigenvector. If the correlation matrix **C** is a symmetric matrix, then it can be denoted as [67]:

$$C = \lambda v v^T \tag{3.5}$$

where v is an orthogonal matrix and v^T is the transpose of v. Using the equation, the cross-correlation matrix **C** can be represented using the projection of eigenvalues and eigenvectors. We can illustrate the previous equation for all eigenvalues as:

$$C = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + ... + \lambda_n v_n v_n^T \tag{3.6}$$

Here, $\lambda_1, \lambda_2, ..., \lambda_n$ are the eigenvalues and $v_1, v_2, ..v_n$ are the corresponding eigenvectors. We call this matrix a 'projection matrix'. Now, if we use random matrix theory, we use $n - r$ eigenvalues, where r is the number of eigenvalues representing noise. We know that the largest eigenvalue contains most of the information about the network, so we consider only the largest eigenvalue in this case. After creating the projection matrix, we derive a distance matrix from this projection matrix, which assigns weights to its elements so that the highest contributor gets the smallest weight (distance) and vice-versa,using the equation below:

$$d_{ij} = 1 - \mid c_{ij} \mid \tag{3.7}$$

Afterwards, we apply a minimum spanning tree to our distance matrix. We know that the minimum spanning tree creates a sub-graph connecting all of the vertices, using the minimum weights of edges. We exploit this property of the spanning tree into the distance matrix, to rank all of the traffic flows according to their contribution. This spanning tree usually has a reference node that organises the whole network. We consider the reference node as the most important or dominant node in the tree. Other important or dominant nodes are identified by their distance away from the reference node. If the distance of a node to the reference node is less, then it means the node is a strong contributor in the network. In our algorithm design, a prims algorithm is used for computing the minimum spanning tree.

In the above sections, we discuss calculating cross-correlations among traffic flows and using different sampling windows while constructing cross-correlation matrices. We also discuss validating our results using an IPR. We understand that the IPR only describes the number of dominant flows and does not identify them as such. Here, we describe our technique for identifying them in a network. This completes our methods for measuring the interdependencies of traffic flows in a network. In the next section, we apply our methods to different network scenarios, to measure network interdependencies.

3.5 Measuring traffic flow interdependencies, with different congestion states

In this section, we present two network topologies for measuring traffic flow interdependencies. We analyse four cases using a network topology and then three cases using a second network topology. We set different congestion states in each case, calculate the cross-correlation coefficients of traffic flows and use different sampling windows to achieve minimum or maximum correlation coefficients. Then, we construct a cross-correlation matrix. Here, we use cut-off cross-correlation coefficients to filter out any random interactions, following which we then validate our results with an IPR value. The first topology contains only four flows, and it is easy to identify dominant flows; thus, we do not apply our technique to find out the dominant flows in the network. The second network topology is not symmetric and has six traffic flows, and in this instance we do use our technique for finding the dominant flows. In these experiments, we observe how the sampling windows suggested by maximum delays can produce peak correlation coefficients.

3.5.1 Network topology 1

Network topology 1 has four traffic flows. The network has a simple symmetric topology and it is easy to visualise the congested parts that create dependencies. We need at least three flows, to show a transitive relationship among the flows. We can show two pairs of transitive relationships using this topology. Due to the small number of flows, it is easy identify dominant examples that match with the IPR value. Network topology 1 is shown in Figure 3.4.

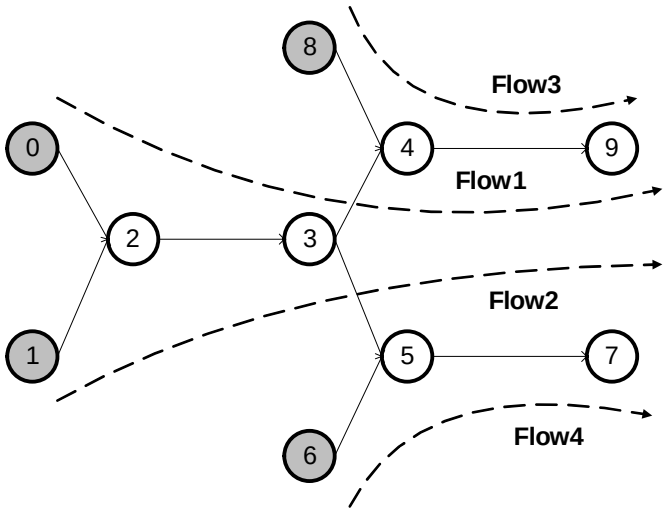


Figure 3.4: Network Topology 1

We create different congestion states in these four cases and measure traffic flow interdependencies. We change the congestion state by changing the traffic rats of different flows in the network. We quantify the congestion state by measuring the utilisation ratio (load) of the queue or node buffer. We describe how to quantify the utilisation ratio in Appendix A. In these cases, we use different colours to represent the utilisation ratio or queue load.

3.5.1.1 Case 1 for network topology 1

In the first case, the network is heavily congested. The load on the queues of nodes 2, 4 and 5 is set to 0.99 (see Figure 3.5).

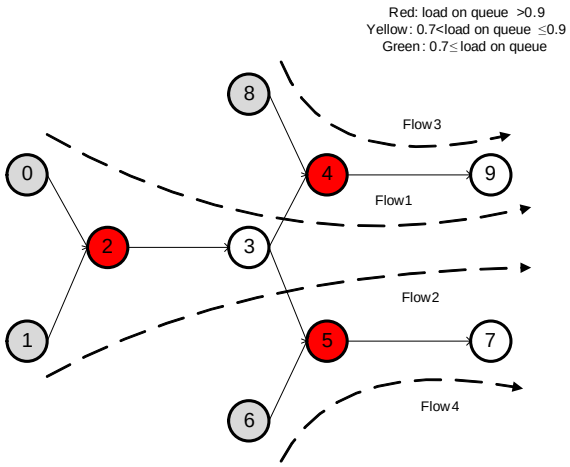


Figure 3.5: Case 1: heavily congested network

Traffic flows	Maximum delay
Traffic flows 1	120 sec
Traffic flows 2	122 sec
Traffic flows 3	50 sec
Traffic flows 4	55 sec

Table 3.4: Delays of traffic flows 1,2,3 and 4

All of the traffic flows pass through heavily congested queues in the network. Flows 1 and 2 have the longest path, as their flows pass through two different queues. The heavily congested queues will spread the delay distribution of flows, depending on the load. As flows 1 and 2 pass through two heavily congested queues, we expect that their end-to-end packet delays will be widely distributed. From Table 3.4, we can see that the delay is spread up to 120 seconds. We take the sampling window of around 100∼200 seconds and find the minimum negative cross-correlation coefficients between traffic flows 1 and 2. Flows 3 and 4 share the same path as flows 1 and 2, respectively. These flows (3 and 4) have the shortest path but are heavily congested in the queues of nodes 4 and 5, respectively. As these flows pass through only one heavy queue, we expect them to have suppressed delay distribution in comparison to flows 1 and 2. From Table 3.4 we can see that the delay spreads up to 50 seconds. We get minimum negative correlation coefficients at around 30∼60 seconds. Cross-correlation coefficients among all traffic flows are greater than the cut-off correlation coefficients, so we expect that all of the flows can be reflected as dominant flows in the IPR.

We consider the largest eigenvalue and associate eigenvector, to identify dominant traffic flows in the network. We calculate the IPR value from the largest eigenvector, which is 4 for all sampling windows. This means that we can identify all of the traffic flows in the congested networks.

3.5.1.2 Case 2 for network topology 1

In the second case, flows 1 and 2 are heavily congested (buffer utilisation is 99% of node 2). Nodes 4 and 5 have 80% buffer utilisation (see figure 3.6).

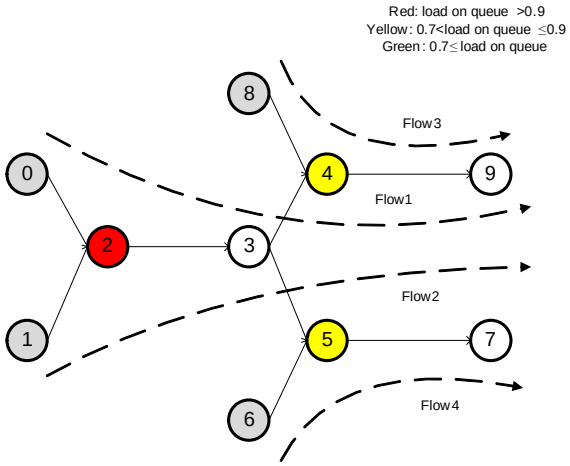


Figure 3.6: Case 2: partial congested network

Traffic flows	Maximum delay
Traffic flows 1	82 sec
Traffic flows 2	80 sec
Traffic flows 3	10 sec
Traffic flows 4	12 sec

Table 3.5: Delays of traffic flows 1,2,3 and 4

Flows 1 and 2 pass through two different congested queues. From Table 3.5, we can see the delays for flows 1 and 2 are distributed up to 80 seconds. We get the minimum correlation coefficients with a sampling window of around 60∼100 seconds. The coefficient starts to increase towards zero thereafter, and as a result, the suitable sampling windows for flows 1 and 2 are around 60∼100 seconds. Flows 3 and 4 are congested with flows 1 and 2, but not as much as in case 1. From Table 3.5, we can see that the delays for flows 3 and 4 are spread up to 10 seconds, so we acquire minimum correlation coefficients,

by using sampling windows of 5∼10 seconds.

The IPR value starts from 4 for sampling windows of 2, 5 and 10 seconds, but it drops down to 2 for sampling windows of 20 seconds upwards. As the level of congestion is less than seen in case 1, it can identify all of the traffic flows with smaller sampling windows. However, traffic flow dependencies passing through less congested path disappear when face with larger sampling windows.

3.5.1.3 Case 3 for network topology 1

In the third case, the load on node 2's queue is set to 0.99, and the load on the queues of nodes 4 and 5 is set to 0.6 (see Figure 3.7).

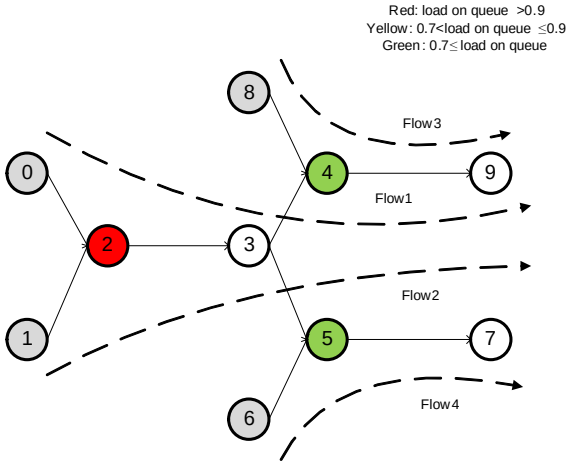


Figure 3.7: Case 3: node 4 and 5 are in uncongested state

Traffic flows	Maximum delay
Traffic flows 1	60 sec
Traffic flows 2	62 sec
Traffic flows 3	4 sec
Traffic flows 4	3 sec

Table 3.6: Delays of traffic flows 1,2,3 and 4

From Table 3.6, we can see that the delays in flows 1 and 2 are distributed up to 60 seconds as they pass through the congested queue of node 2. These flows are the most dominant in the network. The other two flows (3 and 4) have suppressed delay distribution, due to the congestion states. The minimum correlation coefficient is found with a sampling window of around 40∼60 seconds, which makes them

suitable sampling windows for flows 1 and 2. The suitable windows for flows 3 and 4 are around 2∼3 seconds, due to the congestion state.

The IPR value for all of the sampling windows in this network is 2. It always captures flows 1 and 2 as the dominant flows. We notice that due to changes in the congestion state of the network, the dependencies of the two other flows have disappeared.

3.5.1.4 Case 4 for network topology 1

We change the symmetry (in terms of congestion states) of the network in case 4. Flows 1 and 2 are heavily congested (load 0.99 on the queue of node 2), while flows 2 and 4 are also heavily congested (load 0.99 on the queue of node 5). Flows 1 and 3 are in an uncongested state (load 0.6 on the queue of node 4, see Figure 3.8).

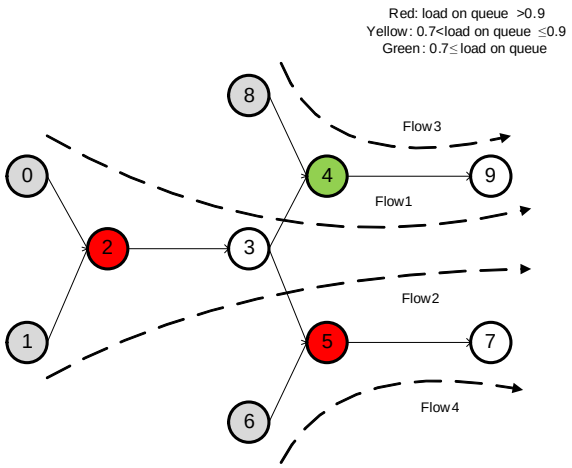


Figure 3.8: Case 4: only node 4 is in uncongested state

Traffic flows	Maximum delay
Traffic flows 1	60 sec
Traffic flows 2	95 sec
Traffic flows 3	4 sec
Traffic flows 4	65 sec

Table 3.7: Delays of traffic flows 1,2,3 and 4

From Table 3.7, we can see that the delay in flow 2 is widely distributed due to congestion. The suitable sampling window is around 100, where we can find the minimum cross-correlation coefficients. Delays

in flows 1 and 4 are distributed up to 60 seconds, so the suitable sampling windows is 60 seconds. For flow 3, the suitable sampling window is around 2 seconds.

The IPR value derived from the largest eigenvector starts from 2 for sampling windows 2-40 seconds. It can identify traffic flows 2 and 4 as being dominant, and it can also capture traffic flow 1 as one of the dominant flows with sampling windows of 40 seconds upwards. Thus, the IPR value jumps to 3 with sampling windows of 40 seconds and remains the same for the rest of the sampling windows. Due to congestion, no significance is evident for flow 3 in the IPR value.

From the results of the above experiments, we found that congestion has a strong influence on dependency measurement. If a traffic flow passes through an uncongested state (buffer utilisation of less than 70%), then its dependency does not reflect in the IPR value. Here, we also observe that minimum cross-correlation coefficients can use sampling windows suggested by maximum delays of traffic flows. With the topology of these experiments, it is easy to detect dominant flows that match with the IPR value, but with a larger network, we need our technique to identify explicitly the most dominant or important flows. In the next section, we present a network topology of six traffic flows. We also apply our technique to identify dominant flows and discuss how minimum or maximum cross-correlation coefficients can be deduced by using the sampling windows suggested by maximum traffic flow delays.

3.5.2 Network topology 2

In this section, we consider a larger network topology than employed in the previous case. The network has six traffic sources and two destinations, as shown in Figure 3.9. Here, we change the symmetry of the topology; for example, flow 1 passes through one extra node (node 4) than flow 2. The presence of this extra intermediate node changes the correlation coefficients of different flows and their measuring interdependencies. With this topology, it is not easy to identify dominant traffic flows that match with the IPR value, so we use our previously introduced technique and validate our result with the IPR value. We observe how cross-correlation coefficients change with the use different sampling windows, and at the same time, we also observe that dominant flows and IPR values change in the same regard. We analyse all of these observations and explain how different sampling windows (suggested by maximum delays of traffic flows) change the correlation coefficients that are responsible for changes in dominant flows and IPR values. The sections below describe three cases using the topology shown in Figure 3.9. We change

the load (utilisation ratio) on the buffers of different traffic flows to change congestion states, following which we calculate cross-correlation coefficients, measure the IPR values and identify dominant flows to match these IPR values.

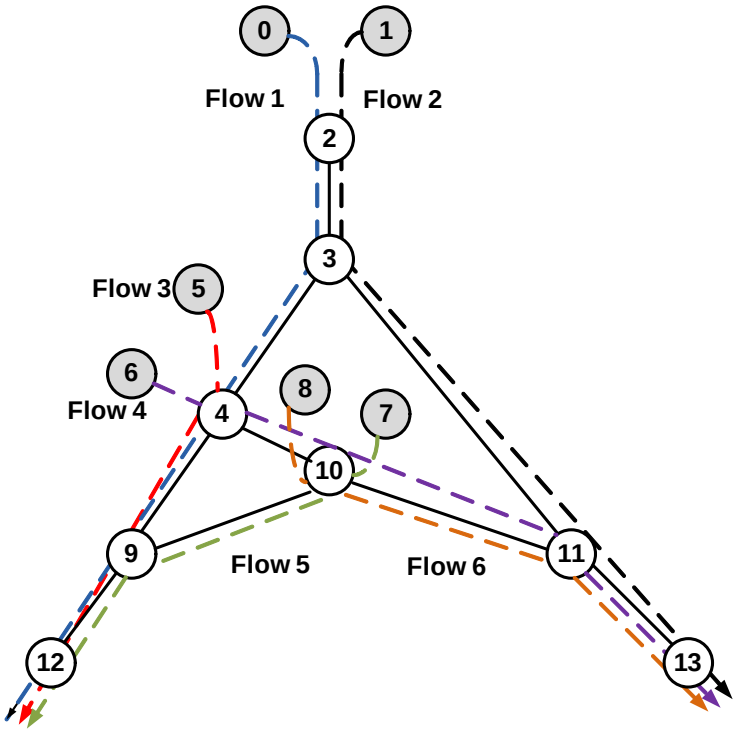


Figure 3.9: Network topology 2

3.5.2.1 Case 1 for network topology 2

In this scenario, we analyse a congested network. All of the node buffers have a more than 90% utilisation ratio (see Figure 3.10), while delay distributions are shown in Table 3.8. Here, we initially check for suitable sampling windows, to acquire the minimum or maximum cross-correlation coefficients of traffic flows suggested by their maximum delays. Then, we construct cross-correlation matrices for different sampling windows. We calculate IPR values and apply our technique to identify dominant flows for each of these cross-correlation matrices. We observe that the IPR value changes while using different sampling windows to construct the cross-correlation matrices. We explain the change in IPR values and dominant flows as being the results of the change in sampling windows in terms of cross-correlation coefficients while analysing the result. Here, we consider only the largest eigenvalue and its associate eigenvector when calculating the IPR value.

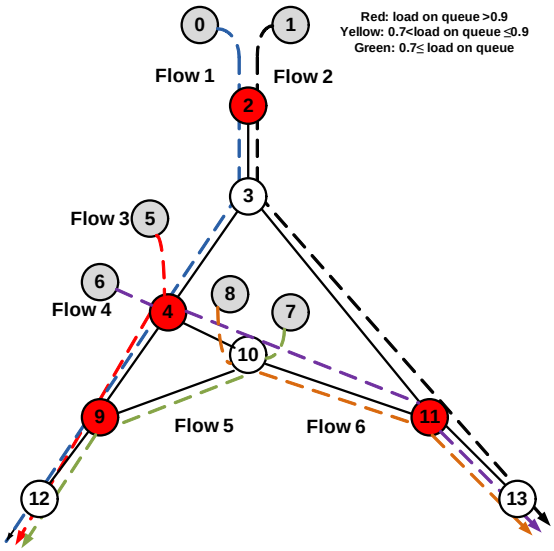


Figure 3.10: Case 1: heavily congested network

Traffic flows	Maximum delay
Traffic flows 1	110 sec
Traffic flows 2	108 sec
Traffic flows 3	25 sec
Traffic flows 4	10 sec
Traffic flows 5	11 sec
Traffic flows 6	12 sec

Table 3.8: Delays in traffic flows 1 to 6

From Table 3.8, we can see that flows 1 and 2 have a maximum delay of around 110 seconds. Consequently, we expect minimum cross-correlation coefficients of around 100∼200 seconds. From the cross-correlation we find the minimum cross-correlation coefficient is -0.31 between flows 1 and 2, with sampling windows of 100 seconds. Flow 3 has a maximum delay of around 25 seconds, so we expect minimum cross-correlation coefficients of around 15∼25 seconds. We get minimum cross-correlation coefficients of -0.54 between flows 1 and 3, with sampling windows of 20 seconds. Flow 6 has a maximum delay of around 12 seconds, and in this case we expect a minimum cross-correlation of around 5∼15 seconds. We get minimum cross-correlation coefficients of -0.40 between flows 4 and 6, with sampling windows of 5 and 10 seconds. Now we can look at the IPR values with different sampling windows, following which we introduce our technique to identify dominant flows in the network. Thereafter, we match these dominant flows with highly correlated flows, to check whether the latter (dependent) flows

are dominant or not.

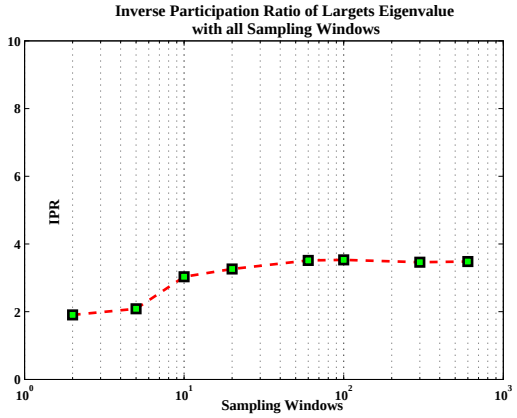


Figure 3.11: IPR values of largest eigenvalues with different sampling windows

From Figure 3.11, we can see that the IPR values start at 2 (for sampling windows of 2 and 5 seconds) and reach up to 3 for this network. For sampling windows of 2 and 5 seconds, the IPR value is 2, meaning there are two dominant flows in the network. We check the cross-correlation matrix for sampling windows of 2 and 5 seconds and find two pairs (flows 1 and 3 and flows 4 and 6) of highly correlated flows. To identify which pair is represented in the IPR value, we introduce a technique to identify dominant flows, which includes eigen-decomposition and a spanning tree.

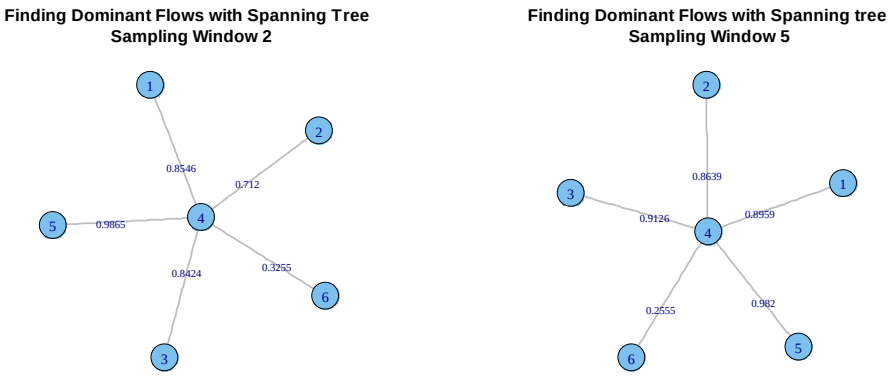


Figure 3.12: Spanning tree to find dominant flows, with sampling windows of 2 and 5 seconds

Traffic flows 4 and 6 have the shortest distance (0.3255 and 0.2555) between them, as we can see in Figure 3.12. Thus, traffic flows 4 and 6 are the dominant flows for sampling windows of 2 and 5 seconds,

and they share the same node buffers, namely 10 and 11, and the same destination node 13. We also get minimum cross-correlation coefficients (-0.40) with sampling windows of 5 and 10 seconds. Their cross-correlation coefficients start to increase towards zero in the presence of larger sampling windows, while. IPR values jump from 2 to 3 (see Figure 3.11) for sampling windows of 10 seconds, which means that there are three dominant flows in the network. The IPR value remains at 3 for the rest of the sampling windows.

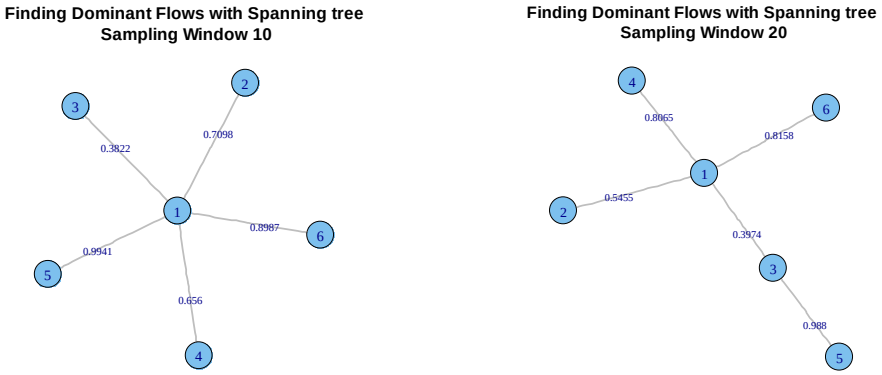


Figure 3.13: Spanning tree to find dominant flows, with sampling windows of 10 and 20 seconds

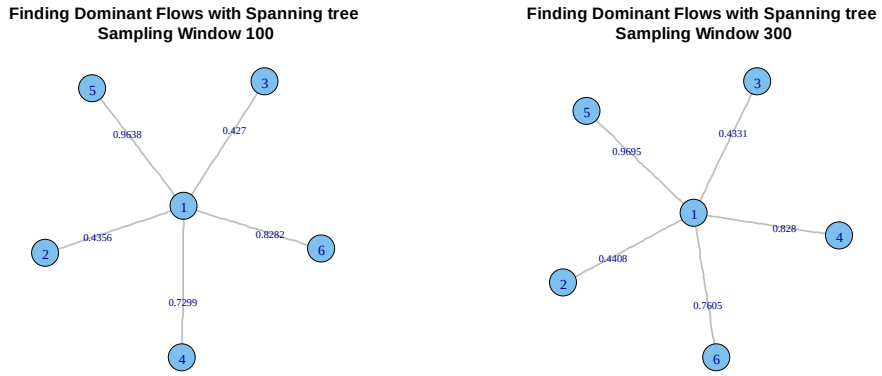


Figure 3.14: Spanning tree to find dominant flows, with sampling windows of 100 and 300 seconds

Our technique identifies that flows 1, 3 and 4 are dominant in relation to sampling windows of 10 seconds, as they have the shortest distance with reference to flow 1, as shown in Figure 3.13. This is a transition period for changing dominant flows, where flows 4 and 6 are replaced by flows 1 and 3.

We identify flows 1, 2 and 3 as dominant for sampling windows of 20 seconds. Here, we can see that flow 4 is replaced by flow 2. We mentioned earlier that minimum cross-correlation coefficients between flows 1 and 3 are found with sampling windows of 20 seconds. While identifying dominant flows in the network, we also found that the distance between flows 1 and 3 is the minimum for sampling windows of 20 seconds. Our technique identifies flows 1, 2 and 3 as dominant for rest of the sampling windows. We acquire minimum cross-correlation coefficients between flows 1 and 2, with sampling windows of 100 seconds. We also find the minimum distance between flows 1 and 2 with sampling windows of 100 seconds, as shown in Figure 3.14. Cross-correlation coefficients between flows 1 and 2 start to increase when we employ sampling windows of more than 100 seconds. We can see the distance between flows 1 and 2 also starts to increase in the presence of a spanning tree of 300 seconds (see Figure 3.14).

3.5.2.2 Case 2 for network topology 2

In this section, we investigate how congestion can affect the measurement of traffic flow interdependencies. The traffic rate of flow 3 is reduced, which changes the load of the node 4 queue to 0.7(see Figure 3.15). All other queues in the network are in heavily congested states (load 0.9). The end-to-end delay distributions of the traffic flows are given in Table 3.9.

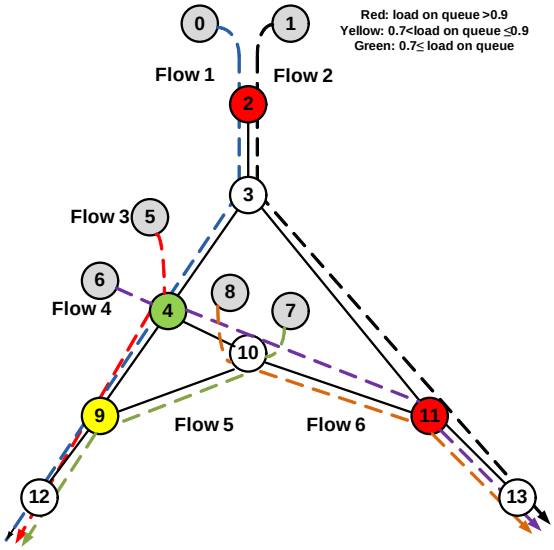


Figure 3.15: Case 2: partly congested network

Traffic flows	Maximum delay
Traffic flows 1	82 sec
Traffic flows 2	80 sec
Traffic flows 3	11 sec
Traffic flows 4	11 sec
Traffic flows 5	10 sec
Traffic flows 6	12 sec

Table 3.9: Delays of traffic flows 1 to 6

The queues for nodes 2 and 4 are heavily congested by traffic flows 1, 2, 4 and 6. Traffic flows 1 and 2 pass through one extra congested node (node 2) than flows 4 and 6, before reaching their destination. For this reason, the delay distributions of traffic flows 1 and 2 are spread more than the other flows, as we can see from Table 3.9. The maximum delay for traffic flows 1 and 2 is around 80 seconds, so we expect minimum cross-correlation coefficients for these traffic flows, using sampling windows of 60$\sim$ 100 seconds. We check the cross-correlation matrices and find minimum cross-correlation coefficients (-0.52) between flows 1 and 2, with sampling windows of 60 and 100 seconds. Flows 3 and 5 have low traffic rates, and so their dependencies with flow 1 disappear in this experiment. Flows 4 and 6 have maximum delays of around 12 seconds, and their minimum cross-correlation coefficients (-0.53) are found with sampling windows of 5 and 10 seconds. Now, we check the IPR values of the cross-correlation matrices with different sampling windows, following which we apply our technique to identify dominant flows. Thereafter, we match the number of highly correlated flows with the dominant flows.

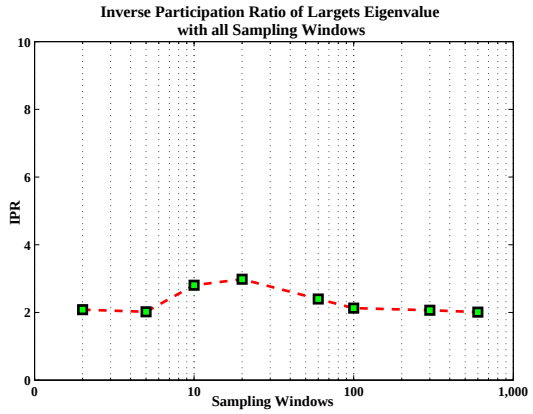


Figure 3.16: IPR values with different sampling windows of six partially congested network traffic flows

Here, we consider the largest eigenvalue and associate eigenvector, to calculate an IPR with different sampling windows. The result shows that the IPR value starts at 2, but then it jumps to 3 for sampling windows 10 and 20 seconds and then falls back to 2 for the rest of the sampling windows, as shown in Figure 3.16. We also apply our technique to identify dominant traffic flows in the network.

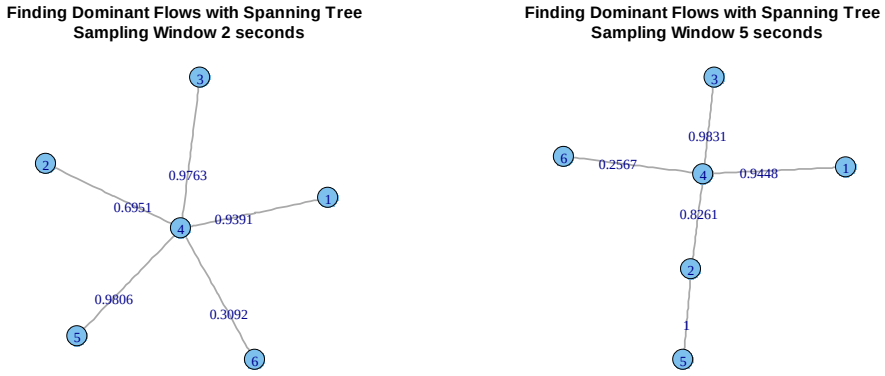


Figure 3.17: Spanning tree to find dominant flows with Sampling Window 2 and 5 seconds for scenario 2: case 2

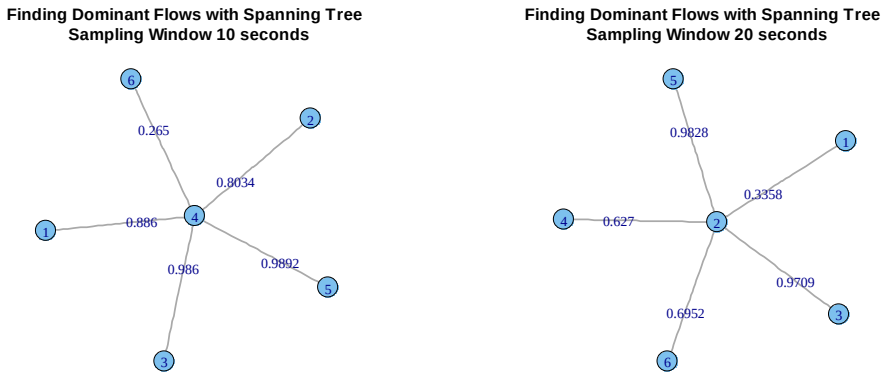


Figure 3.18: Spanning tree to find dominant flows, with sampling windows of 10 and 20 seconds for scenario 2, case 2

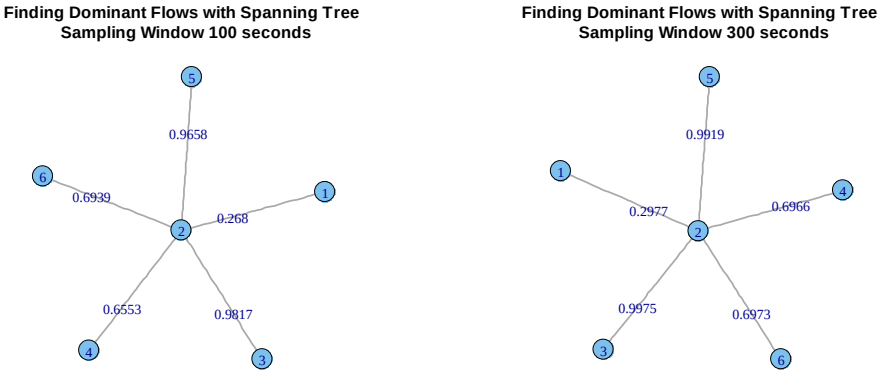


Figure 3.19: Spanning tree to find dominant flows, with sampling windows of 100 and 300 seconds for scenario 2, case 2

Figure 3.17 shows that traffic flows 4 and 6 have the shortest distance (0.3092 and 0.2567) for sampling windows of 2 and 5 seconds, so they are dominant for sampling windows of 2 and 5 seconds. We also discover that flows 4 and 6 have minimum correlations with sampling windows of 5 and 10 seconds. The IPR value jumps from 2 to 3 for sampling windows of 10 seconds, as shown in Figure 3.16, while Figure 3.18 suggests that traffic flows 2, 4 and 6 are dominant, with sampling windows of 10 seconds. As we know, when cross-correlation coefficients between flows 4 and 6 start to increase towards zero, their dependencies also start to disappear, with sampling windows of more than 10 seconds. With sampling windows of 20 seconds in Figure 3.18, we can see flow 2 as a reference node of the spanning tree. It is closer to flow 1 than the other flows, and flow 4 is slightly further away than flow 6. Thus, flows 1, 2 and 4 become dominant flows in the network. It is a transition period whereby dominant flows 4 and 6 for the sampling windows of smaller sampling windows (2$\sim$10 seconds) are replaced by flows 1 and 2. The IPR value falls down from 3 to 2 and remains at 2 for the rest of the sampling windows. Our technique identifies flows 1 and 2 as dominant flows for the rest of the sampling windows. These flows have minimum correlation coefficients with sampling windows of 60 and 100 seconds. In the spanning tree, they also have the shortest distance (0.268) with a sampling window of 100 seconds. As their correlation coefficients starts to increase, we can see their distance (0.297) also starts to increase in line with sampling windows larger than 100 seconds, as seen in Figure 3.19.

3.5.2.3 Case 3 for network topology 2

In this case, we measure the interdependencies among flows in a partially congested network. We reduce the traffic rates of flows 3 and 6, which apply a load of 0.7 onto the queues of nodes 4 and 11, as shown in Figure 3.20. In this scenario, we can see only flows 1 and 2 passing through a congested node 2, which in turn creates dependency between these two flows. As a result, we expect these two flows only to be dominant.

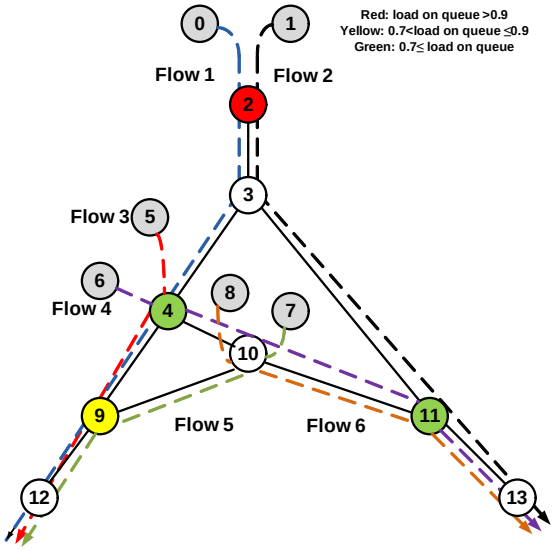


Figure 3.20: Case 3: nodes 4 and 11 are in an uncongested state

Traffic flows	Maximum delay
Traffic flows 1	22 sec
Traffic flows 2	20 sec
Traffic flows 3	4 sec
Traffic flows 4	3 sec
Traffic flows 5	5 sec
Traffic flows 6	4 sec

Table 3.10: Delays in traffic flows 1 to 6

As the network is less congested (only flows 1 and 2 pass through the congested node), maximum delays for flows 1 and 2 are around 22 seconds. Consequently, we expect a minimum cross-correlation coefficient of around 20$\sim$ 30 seconds. We check cross-correlation matrices and find that the minimum correlation coefficient between these two flows is -0.79, which occurs while using sampling windows of

20 seconds. Now, we can calculate the IPR values of the six traffic flows, the results for which are given in Figure 3.16.

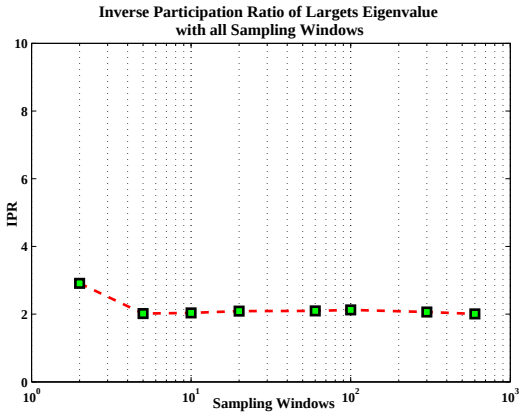


Figure 3.21: IPR values with different sampling windows for six traffic flows in a partially congested network

Figure 3.21 shows that the IPR value is initially close to 3. It drops from 3 to 2 with a sampling window of 5 second upwards. Our technique finds dominant flows for each sampling window.

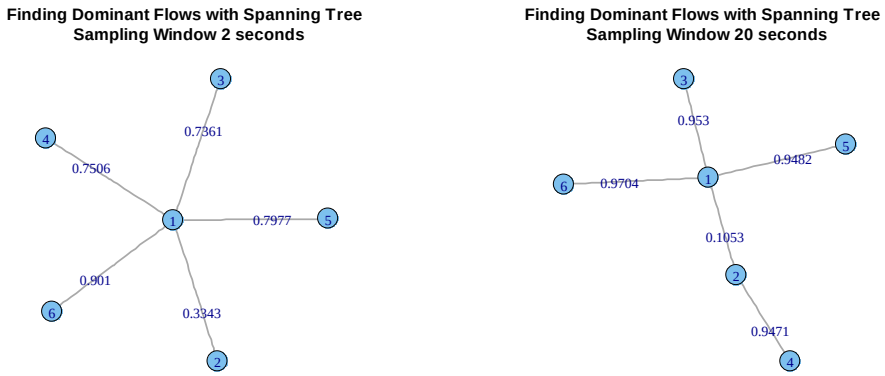


Figure 3.22: Spanning tree to find dominant flows, with sampling windows of 2 and 20 seconds for network scenario 2, case 3

Figure 3.22 shows the spanning tree for sampling windows of 2 and 20 seconds. In both cases, we can see that flow 1 is the reference node in the spanning tree, and it is closer to flow 2. We can see flow 3 is close to flow 1 in the sampling window of 2 seconds, but the distance between flows 1 and 3 (0.7361) is quite significant in relation to the distance between flows 1 and 2 (0.3343), which shows that flow 3

is weakly dependent on flow 1 with a sampling window of 2 seconds. It also reflects in the IPR value. With a sampling window of 5 seconds and upwards, this dependency between flow 1 and 3 disappears. The rest of the spanning tree for the other sampling windows shows that flows 1 and 2 are the dominant flows. The distance between flows 1 and 2 in the spanning tree becomes the shortest with sampling windows of 20 seconds. We also find minimum cross-correlation coefficients between these two flows, with sampling windows of 20 seconds.

From the experiments above, we observe that the congestion state of a network can affect how we calculate cross-correlation coefficients. If a traffic flow passes through an uncongested node buffer, then it does not create any dependency with the other flows; instead, it exhibits weak correlation coefficients with the other flows and its significance does not reflect in the IPR value. We also find that traffic flows with weak dependency are far away from the reference node in the spanning tree. From these experiments, we understand that we need a certain level of congestion (buffer utilisation more than 70%) in a network to calculate cross-correlation among traffic flows. We also observe that we calculate peak (minimum or maximum) cross-correlation coefficients if we choose sampling windows around their maximum delays. Then, we check the spanning tree of the dominant flows for that particular sampling window that provides peak correlation coefficients. We find that the distance between flows becomes shortest for that sampling window. Thus, we can say the sampling window from which we acquire peak correlation coefficients between traffic flows also exhibits the shortest distance in the spanning tree of the dominant flows.

3.6 Summary

In this chapter we have discussed our methodology for measuring interdependencies in traffic flows. We find that congestion can affect our measurements, and so we need a certain level of congestion for the purposes of accuracy. During our calculation, we find that if we take sampling windows of around the maximum delays of the traffic flows, we produce peak (maximum or minimum) cross-correlation coefficients. As different traffic flows have different delays, we use different sampling windows to acquire their peak correlation coefficients. With the use of different sampling windows, we can measure complete sets of dependencies, and to validate our results, we use an IPR value derived from the eigen analysis of cross-correlation matrices, which gives us the number of dominant flows in the network.

Usually, highly correlated flows are considered as dominant. To justify this argument, we introduce our technique to identify such flows, and we then match the number of highly correlated flows with dominant flows and the IPR value, and we discover that highly correlated flows are dominant for the two network topologies presented in this chapter. We also find that a sampling window that provides a peak correlation between traffic flows also exhibits the shortest distance in the spanning tree while identifying dominant flows.

In the first example of this chapter, we observe that if flows pass through multiple congested node buffer, the result is weaker cross-correlation coefficients (using data collected at the destinations) than flows passing through one congested node buffer. Here, we present two small network topologies in which a flow passes through a maximum of three congested node buffers. With a certain congestion state of these networks, we apply our methodology and measure traffic flow interdependencies accurately. In the bigger topology, a flow expects to pass through many congested intermediate node buffers before reaching its destination. As it passes through multiple nodes, it is expected to have greater delay distribution. In these cases, we expect weaker cross-correlation coefficients, due to delays in the traffic flows.

We wish to investigate how our methodology measures traffic flow interdependencies in this condition, and so we need to answer the following questions: a) can we measure the peak cross-correlation coefficients of traffic flows that pass through many congested nodes, b) does a sampling window have any influence over the peak correlation coefficients of traffic flows that pass through many congested nodes and c) can our technique identify flows as dominant, even though they may exhibit weak cross-correlation coefficients, due to delays caused by multiple congested nodes. To answer these questions, we to create large network topologies and apply our method for measuring traffic flow interdependencies in the next chapter.

Chapter 4

Measuring Traffic Flow Interdependencies for Large Topologies

---

4.1 Introduction

Measuring traffic flow interdependencies is the building block of constructing a functional topology. Thus, it is really important to measure complete sets of traffic flow interdependencies accurately. In the previous chapter, we discussed a methodology for measuring these traffic flow interdependencies, and we learnt how congestion can affect our dependency measurement. From that observation, we understood that we need a certain level of congestion (buffer utilisation more than 70%) to obtain accurate traffic flow interdependencies. We also observed that sampling windows suggested by maximum delays in traffic flows can be a means of acquiring peak cross-correlation coefficients between traffic flows. We also measured dependency strength between traffic flows, according to cross-correlation coefficients. To validate our results, we matched a number of highly correlated traffic flows with the IPR value. In the previous chapter, we also found that when two flows pass through two congested node buffers, we see weaker cross-correlation than when flows pass through single congested nodes. In large networks, it is expected that flows pass through multiple congested nodes, and so we wanted to learn how our methodology measures traffic flow interdependencies in this case.

We discuss two large network topologies (including one local area network) in this chapter. In order to measure traffic flow interdependencies, we calculate cross-correlation coefficients among traffic flows and construct a cross-correlation matrix. To obtain a complete set of interdependencies, we use different sampling windows, suggested by maximum delays in traffic flows. We expect to get peak cross-correlation coefficients, using sampling windows suggested by the flow's maximum delay. Then, we identify dominant network flows, using our technique described in this previous chapter, and thereafter, we match dominant network flows with IPR values to validate our results in relation to measuring traffic flow interdependencies. Here, we focus on traffic flows passing through multiple congested nodes in a network, as we need to ascertain whether we could produce peak cross-correlation coefficients of traffic flows suggested by their maximum delays. We are also interested in discovering whether our method identifies flows that exhibit weak correlation coefficients, due to their large delays as a result of being dominant network flows.

4.2 Network topology with 17 traffic flows

In this section, we measure the traffic flow interdependencies of a network that has 17 routes (source-destination pairs), as illustrated in Figure 4.1.

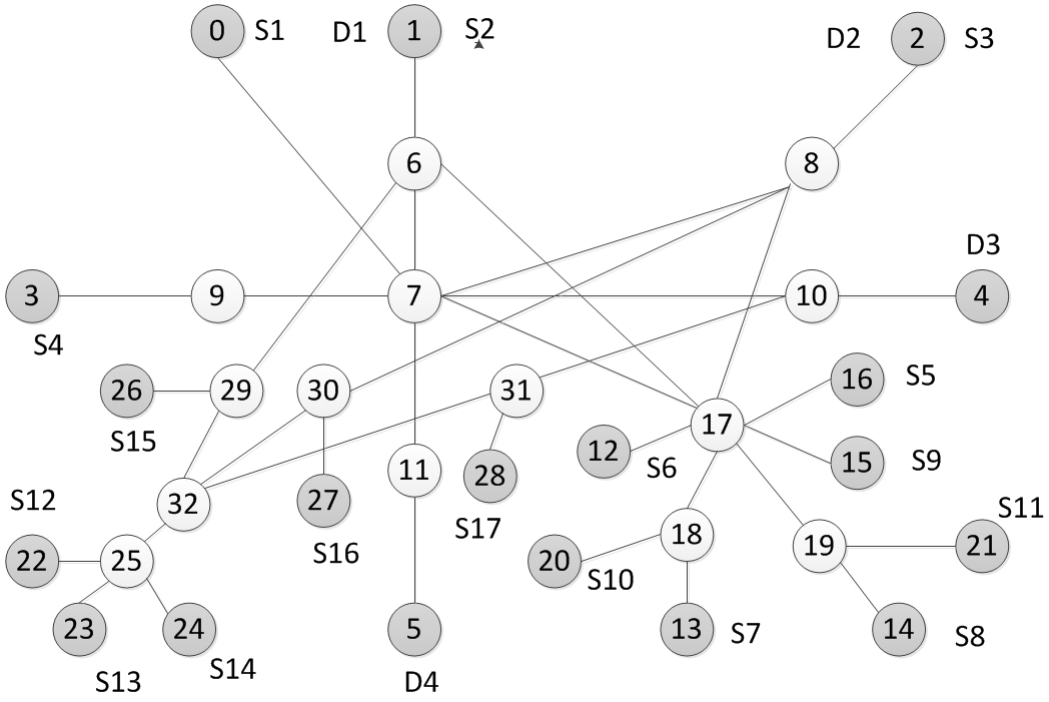


Figure 4.1: Network topology of a 17-route network

In this network, we have 17 traffic sources and four destinations. We use Pareto distributed traffic (long-range-dependent traffic) and Poisson distributed traffic in this experiment. The details (how packets traverse through the path) of each source-destination are given in Table 4.1.

S1	0	7	10	4	D3		
S2	1	6	7	10	4	D3	
S3	2	8	7	11	5	D4	
S4	3	9	7	11	5	D4	
S5	16	17	7	11	5	D4	
S6	12	17	6	1	D1		
S7	13	18	17	6	1	D1	
S8	14	19	17	8	2	D2	
S9	15	17	8	2	D2		
S10	20	18	17	8	2	D2	
S11	21	19	17	6	1	D1	
S12	22	25	32	29	6	1	D1
S13	23	25	32	30	8	2	D2
S14	24	25	32	31	10	4	D3
S15	26	29	6	1	D1		
S16	27	30	8	2	D2		
S17	28	31	10	4	D3		

Table 4.1: Source-destination pair with intermediate nodes

In this network, we set a high rate for all traffic flows, except for traffic flows 9, 12 and 15. Consequently, most of the node buffers have a utilisation of more than 80%. We over-provision nodes 7 and 11. Buffer utilisation for these nodes is around 60%. We collect the data from end points and construct a cross-correlation matrix, for which we then calculate the eigenvalues and eigenvectors. In order to measure traffic flow interdependencies, we need to analyse the eigenvalues and associated eigenvectors that contain most information of the system. Here, we choose first the three largest eigenvalues derived from the cross-correlation matrix, as these are significantly larger than the rest of the eigenvalues. In Figure 4.2, we compare the strength of the first-, second- and third-largest eigenvalues of the cross-correlation matrices for different sampling windows. Eigenvalues are sorted in descending order, and we can see that their strength becomes significant as the sampling windows increase. As the largest eigenvalue carries most of the information and is more significant in strength than the other two largest eigenvalues, we analyse only the largest eigenvalue in this section.

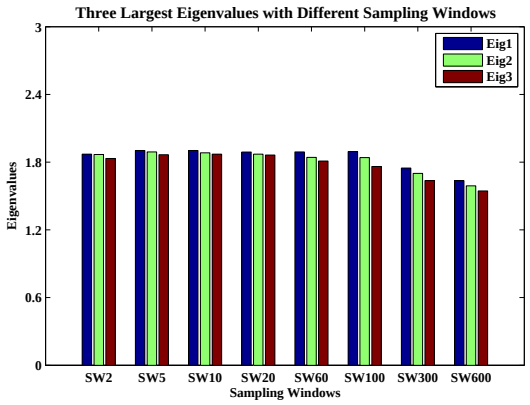


Figure 4.2: Three largest eigenvalues grouped with different sampling windows

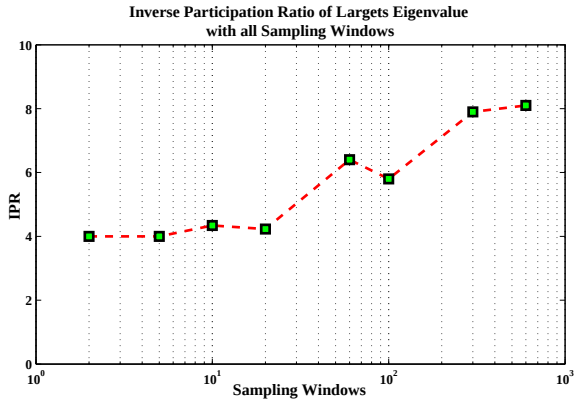


Figure 4.3: IPR values of the largest eigenvalue, with different sampling windows

In order to measure the traffic flow dependencies, we need to calculate the inverse participation ratio (IPR) of the largest eigenvalue. This number reflects the number of dominant flows in the network. Sometimes, two flows passing through multiple congested nodes exhibit weak cross-correlation coefficients due to their large delays. In this type of cases, the IPR can be really helpful and identify the traffic flow dependencies. Figure 4.3 shows that IPR values increase gradually in line with sampling windows, which means that we can explore more traffic flow dependencies by using different sampling windows. We apply our technique, which uses eigendecomposition and spanning tree (discussed in the previous chapter) algorithm, to identify the dominant flows and analyse why they keep changing in line with the sampling windows.

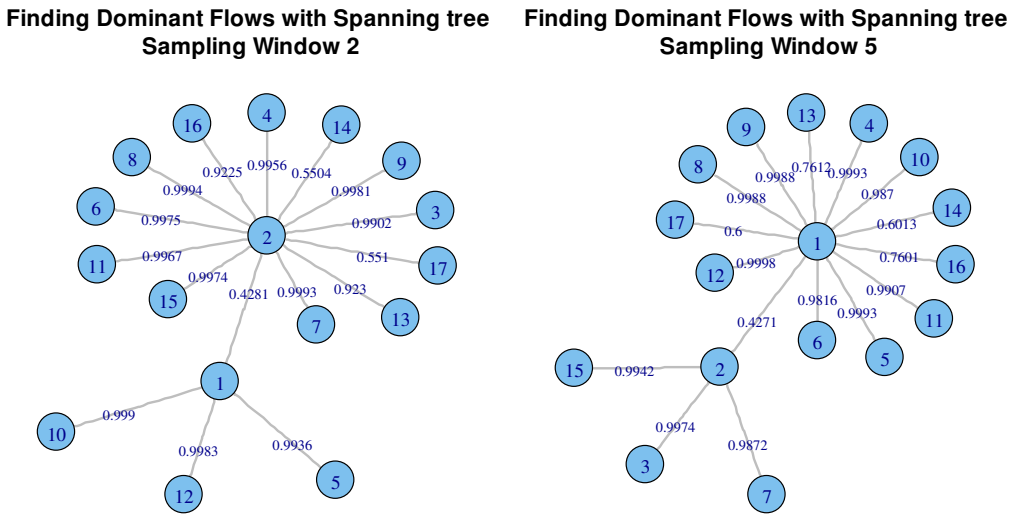


Figure 4.4: Spanning tree for finding dominant flows, with sampling windows of 2 and 5 seconds

The IPR value is 4 for sampling windows of 2 seconds, as shown in Figure 4.3, so four dominant flows are identified as captured by a sampling window of 2 seconds. To find out which flows are dominant, we apply the eigendecomposition and a spanning tree-based technique described in the previous chapter. The technique considers the largest eigenvalue, eigenvector and spanning tree to rank all of the flows according to their contribution. The spanning tree shows that flows 1, 2, 14 and 17 are dominant in Figure 4.4. Here, flow 2 is the reference node (central node), and flows 1, 14 and 17 are closer to flow 2 in terms of distance.

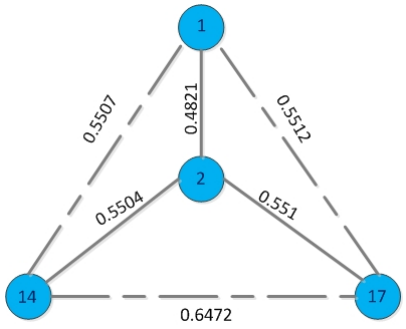


Figure 4.5: Distance among flows 1, 2, 14 and 17 in the spanning tree

From the network topology shown in Figure 4.1 and Table 4.1, we can see that flows 1, 2, 14 and 17 have the same destination, namely D3. All of these four flows share the same nodes 10 and 4 resources along their path. The queue for node 10 becomes congested and results in the negative correlation of these

four flows. Furthermore, node 4 is over-provisioned and does not affect the creation of a correlation in traffic flows. The end-to-end delay to flows 1 and 2 is smaller than for flows 14 and 17, so smaller sampling windows can capture the stronger dependencies between flows 1 and 2 amongst four flows. To analyse this scenario further, we check the distance matrix values of these four flows. Figure 4.5 shows the minimum values that are considered by the spanning tree in solid lines, while the other values are in dashed lines. The differences between these two sets (solid lines and dashed lines) of values are very minimal. The spanning tree only considers the lowest value, which is why it considers flows 1 and 2 as reference nodes of the spanning tree and flows 14 and 17 as other dominant flows in the network.

The IPR value is 4 for a sampling window of 5 seconds, and we can see the same set of dominant traffic flows (flows 1, 2, 14 and 17) in the network capture. In the spanning tree, we can see that there are two main reference nodes – flows 1 and 2. We also find that flows 1 and 2 have a minimum cross-correlation coefficient of -0.8928, which is captured using sampling windows of 5 seconds. Figure 4.4 shows that the distance becomes the shortest (0.4271) between flows 1 and 2, by using a sampling window of 5 seconds in the spanning tree.

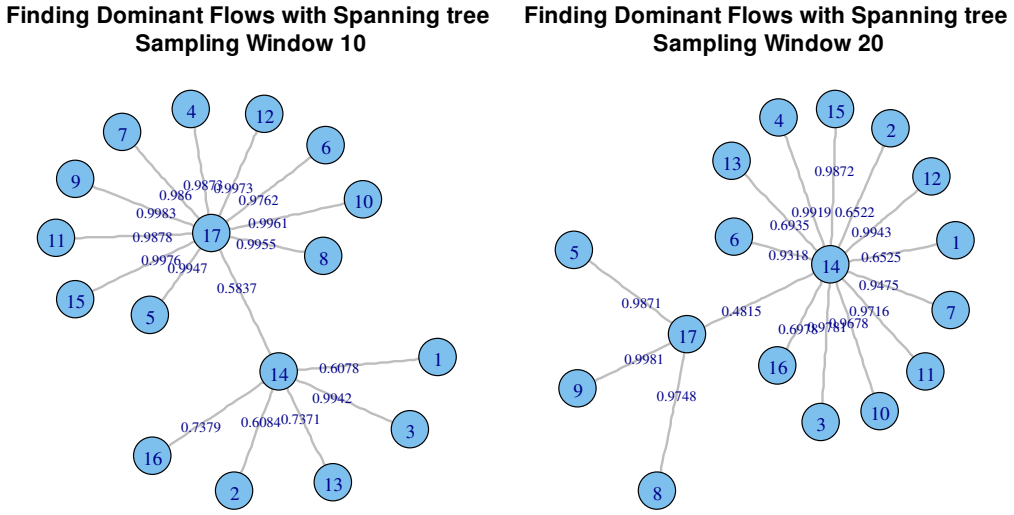


Figure 4.6: Spanning tree for finding dominant flows, with sampling windows of 10 and 20 seconds

The IPR value is 4 for sampling windows of 10 and 20 seconds, as shown in Figure 4.3. The dominant traffic flows for these two sampling windows are flows 1, 2, 13, 14 and 17, as they are closest to the reference nodes in the spanning tree. Here, the reference nodes of the spanning tree have been changed to flows 14 and 17, with sampling windows of 10 and 20 seconds, as we can see from Figure 4.6. We observe that cross-correlation coefficients between flows 1 and 2 start to increase when flows 14 and

17 start to decrease. We found that the minimum cross-correlation coefficient between flows 14 and 17 is -0.8933, which is captured using sampling windows of 20 seconds. From figure 4.6, we can see that distance between nodes 14 and 17 is at its shortest (0.4815) when using sampling windows of 20 seconds in the spanning tree.

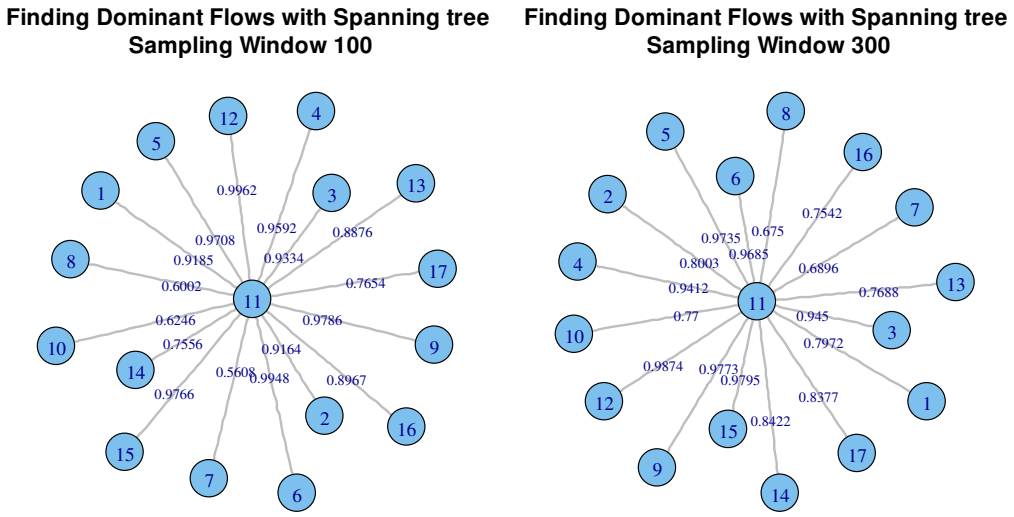


Figure 4.7: Spanning tree for finding dominant flows, with sampling windows of 100 and 300 seconds

The IPR value jumps from 4 to 6 for sampling windows of 60 and 100 seconds, as shown in Figure 4.3. The spanning tree for sampling windows of 60 seconds is similar to that used for sampling windows of 100 seconds, and both have the same dominant flows and reference flows. Thus, we discuss only the spanning tree for 100 seconds. We can see that traffic flow 11 is now the reference node of the spanning tree for sampling windows of 100 and 300 seconds, as shown in Figure 4.7, and we can identify that traffic flows 7, 8, 10, 11, 14 and 17 are the dominant, with sampling windows of 60 and 100 seconds. Reference flow 11 shares the same node 17 with flows 7, 8 and 10, as shown in Figure 4.1. These flows show strong dependencies on flow 11. Flows 8 and 11 also share a congested node buffer, 19, before they reach node 17. As they pass through multiple congested nodes before reaching their destination, they exhibit a weak cross-correlation coefficient of -0.0078, which is captured by sampling windows of 100 seconds. These two flows have the shortest distance at 0.6002, with sampling windows of 100 seconds. Our technique identifies flow 8 as the dominant flow, though cross-correlation coefficient between flows 8 and 11 is really weak. Flows 7 and 10 have peak cross-correlation coefficients with flow 11, with sampling windows of 100 seconds, and they have the shortest distance (distances from flow 11 to flows 7 and 10 are 0.5608 and 0.6246, respectively) in the spanning tree with the same sampling window. It

also captures two other dominant traffic flows, 14 and 17, as being dominant. Traffic flow 2 has strong dependencies with traffic flows 14 and 17, as we have shown before, but it also shares the same node buffer 6 with traffic flow 11. Thus, sampling windows of 60 and 100 seconds capture the dependency (transitive relationship) of traffic flows 14 and 17 on traffic flow 11. Flows 14 and 17 have a larger distance than other stronger dominant flows (flows 7, 8 and 10 with reference flow 11).

Eight dominant traffic flows are captured by sampling windows of 300 seconds. According to the spanning tree, traffic flows 1, 2, 7, 8, 10, 11, 13 and 16 have sampling windows of 300 seconds. Traffic flows 7, 8, 10 share the same resources with traffic flow 11, whilst traffic flow 1 has a transitive relationship with traffic flow 11, as both share the same resource with traffic flow 2. As a result, we can treat traffic flows 1 and 2 as being dominant, with sampling windows of 300 seconds. Traffic flows 8 and 10 share the same node buffer 17 with traffic flow 11, while they also share the same buffer with traffic flows 13 and 16 at node buffer 8, as shown in Figure 4.1. Due to this transitive relationship, we can treat traffic flows 13 and 16 as being dominant in line with sampling windows of 300 seconds.

In this section, we discover that our technique can identify dominant flows in a network. We are also able to capture a complete set of interdependencies measurements, by using different sampling windows. Maximum delays in traffic flows 1, 2, 14 and 17 are fewer than other dominant flows in the network. Thus, their dependency is captured using small sampling windows (up to 20 seconds). We are able to obtain minimum cross-correlation coefficients for these traffic flows, using sampling windows of 10$\sim$ and 20 seconds. We also observe that they have the shortest distances in the spanning tree, using the same sampling windows. Traffic flows 3, 4 and 5 pass through some uncongested node buffers (7 and 11), so we do not consider them dominant traffic flows in the network. Traffic flows 7, 8, 10 and 11 have greater delays than flows 1, 2, 14 and 17, so we acquire their minimum cross-correlation coefficients by using larger sampling windows. Using the same sampling windows, our technique identifies them as dominant, and we also find that they have the shortest distance in the spanning tree, with same sampling windows of 100 seconds. In this experiment, we find that our technique can capture the dependency of traffic flows passing through many congested nodes and exhibiting weak cross-correlation coefficients due to their large delays. For example, we find traffic flow 8 is dominant, even though it exhibits weak cross-correlation with traffic flow 11. We will consider this dependency between flows 8 and 11 while we construct our functional topology. In the next section, we measure the traffic flow interdependencies of a local area network (LAN), for which we expect to find more flows passing through multiple nodes.

We will check whether our methodology can measure their dependencies.

4.3 Measuring the interdependencies of end-to-end traffic flows in a local area network

In this section, we discuss how our methodology measures the inter-traffic flow dependencies in a large network such as a LAN. First, we collect data at traffic flow destinations, use different sampling windows in time series data and construct cross-correlation matrices for each sampling window. Then, we apply our technique, based on eigendecomposition and a spanning tree, to identify dominant flows in the network and match these with IPR values. In the previous chapter and last section, we observed that the use of different sampling windows suggested by maximum delays in traffic flows is very useful for obtaining a complete set of traffic flow interdependencies – we expect to see the same results for our LAN topology. From our previous experiments, we observed that traffic flows passing through multiple congested nodes exhibit weak cross-correlation coefficients, but our method can still identify them as dominant network flows. We use small topologies from previous experiments and find flows passing through one or two congested nodes. In this LAN topology, we have many flows passing through multiple congested nodes and exhibiting weak correlation. The aim of this section is to find out whether our method can measure the traffic flow interdependencies of a LAN and identify all dominant network flows, especially those passing through multiple congested nodes and which have weaker cross-correlations among themselves.

Figure 4.8 shows the network topology which is modelled on a departmental internal LAN topology of an university. This network has 147 traffic flows. Here, traffic generator sources are considered as users of the university, and intermediate nodes are considered as switches. We consider a congested network (buffer utilisation at more than 70%) to apply our method, and we use Pareto distributed and Poisson traffic sources. Furthermore, we create congestion in the network by changing different traffic generation rates at the sources. Consequently, switch buffers are congested and correlate traffic flows passing through.

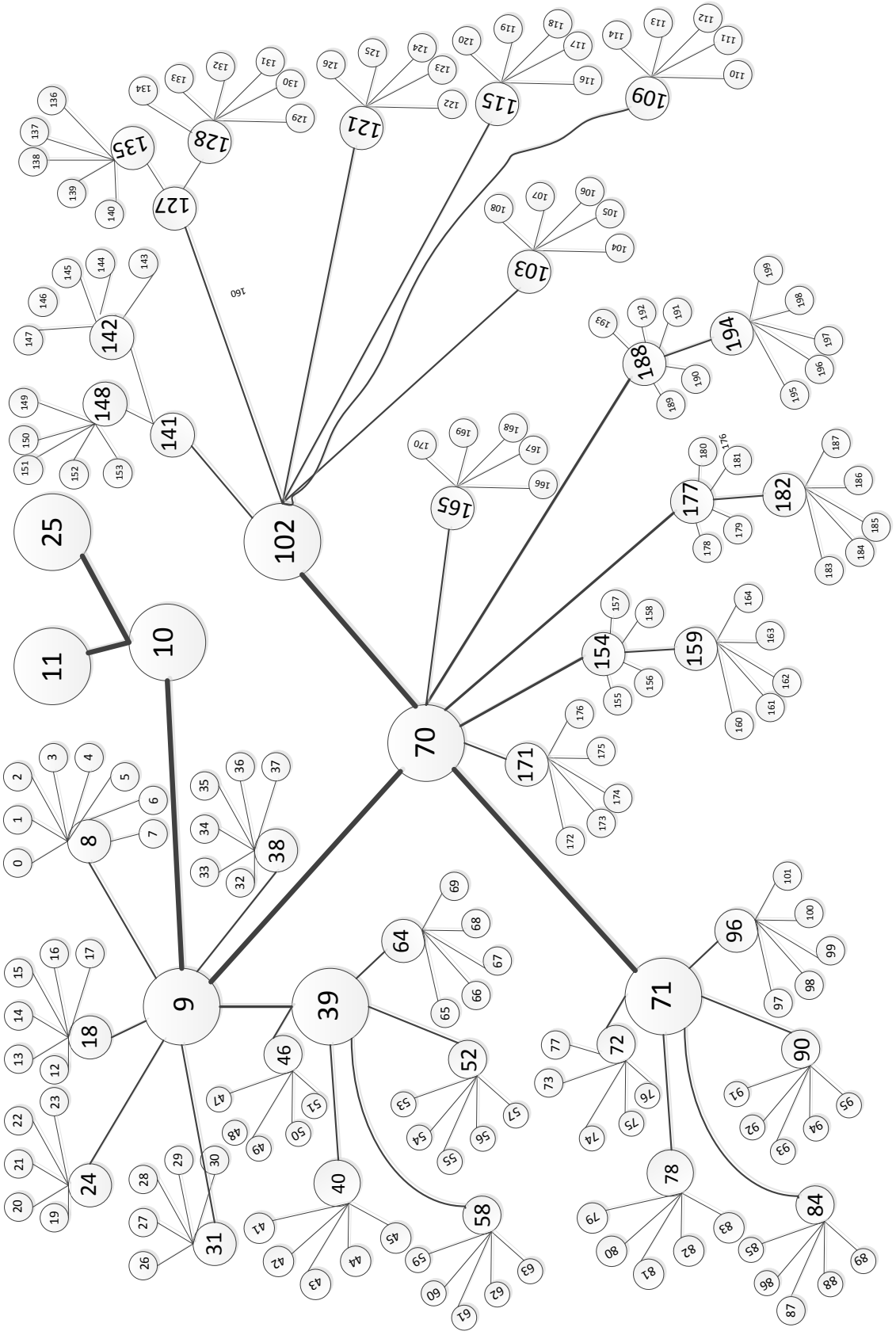


Figure 4.8: Network topology of a 147-route network

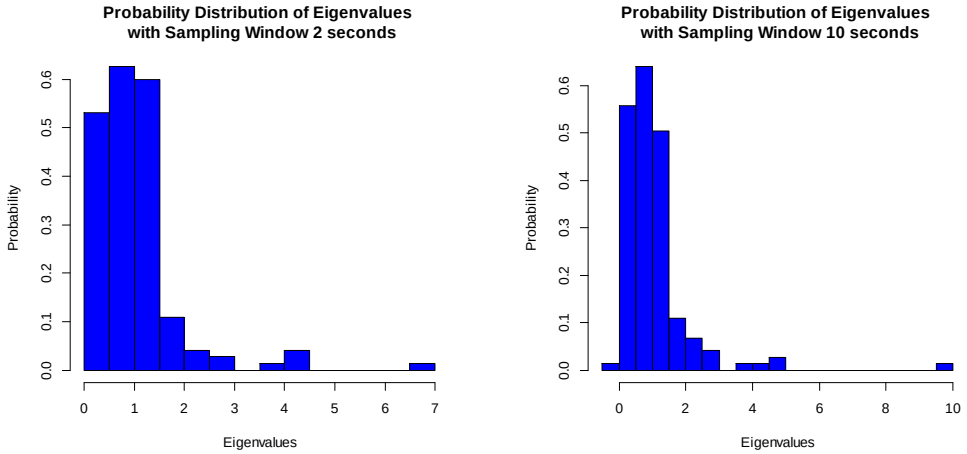


Figure 4.9: Histogram of eigenvalue distribution: sampling windows of 2 and 5 seconds

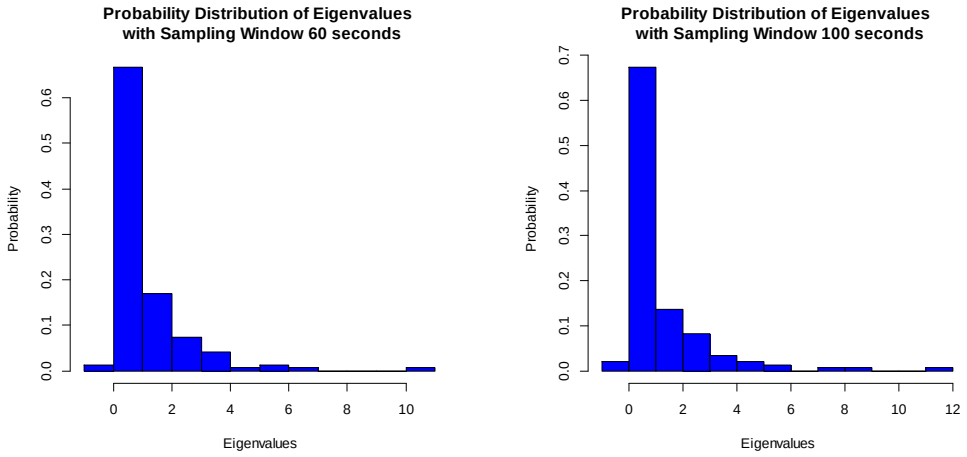


Figure 4.10: Histogram of eigenvalue distribution: sampling windows of 60 and 100 seconds

We apply our methodology to measure the traffic flow interdependencies on a LAN. First, we construct a cross-correlation matrix after collecting data from the network's end points. Thereafter, we calculate the eigenvalues and eigenvectors of cross-correlation matrix. Figure 4.9 and Figure 4.10 show the eigenvalue distribution of the correlation matrix, with sampling windows of 2, 10, 60 and 100 seconds. We can see that there is a distinct gap between the largest eigenvalue and the others, which contains most of the information about the system. For this reason, we will use only this largest eigenvalue to determine traffic flow interdependencies in this section. We now use our eigendecomposition-based technique, described in the last chapter, to identify dominant network flows for each sampling window.

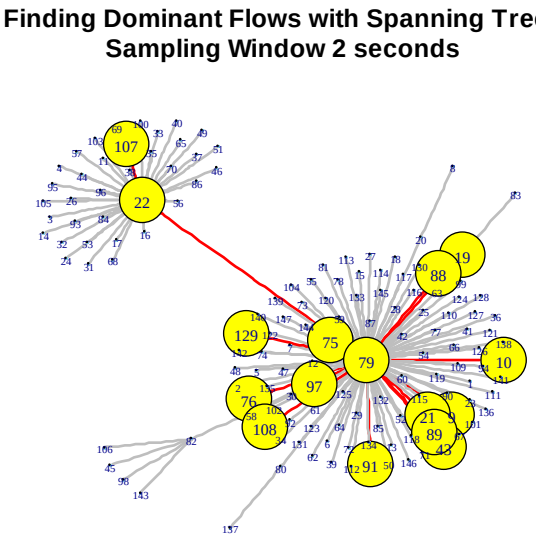


Figure 4.11: Spanning tree to find dominant flows, with a sampling window of 2 seconds

We calculate the IPR value from the largest eigenvector deduced from our cross-correlation matrix, using sampling windows of 2 seconds. The IPR value for this sampling window is 16, meaning that there are 16 dominant flows in the network. Figure 4.11 shows that there are two reference nodes – 22 and 79 – in the spanning tree. All the dominant nodes, along with their paths, are highlighted in the spanning tree. From the Figure 4.11, we can see that flow 107 is dependent on traffic flow 22. The rest of the 13 flows are related to reference node 79. Now, we shall explore how these traffic flows are related to each other.

We highlight all the dominant traffic flow end-to-end paths in Figure 4.12, to find out the relationships of dominant flows with reference flows. Here, the source is coloured yellow, while the destination is coloured blue. Nodes which are both source and destination are coloured green. Table 4.2 shows the source node id, flow passes through the intermediate nodes and the destination node id of the dominant traffic flows, with sampling windows of 2 seconds. In Table 4.2, we show the reference traffic flow's information at the top.

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
79	112	102,70,9	30
9	12	9,10	11
10	13	9,10	25
19	27	9,10	11
21	29	9,70,71	77
22	83	71,70,9	30
43	57	39,9,70,71	89
75	107	102,70,9	25
76	108	102,70,9	7
88	124	102,70,71	101
89	125	102,70,71	101
91	129	102,70,9	11
97	136	102,70	157
107	151	102,70,9	7
108	152	102,70	199
129	178	70,9	11

Table 4.2: Source destination pairs of dominant traffic network flows, with sampling windows of 2 seconds

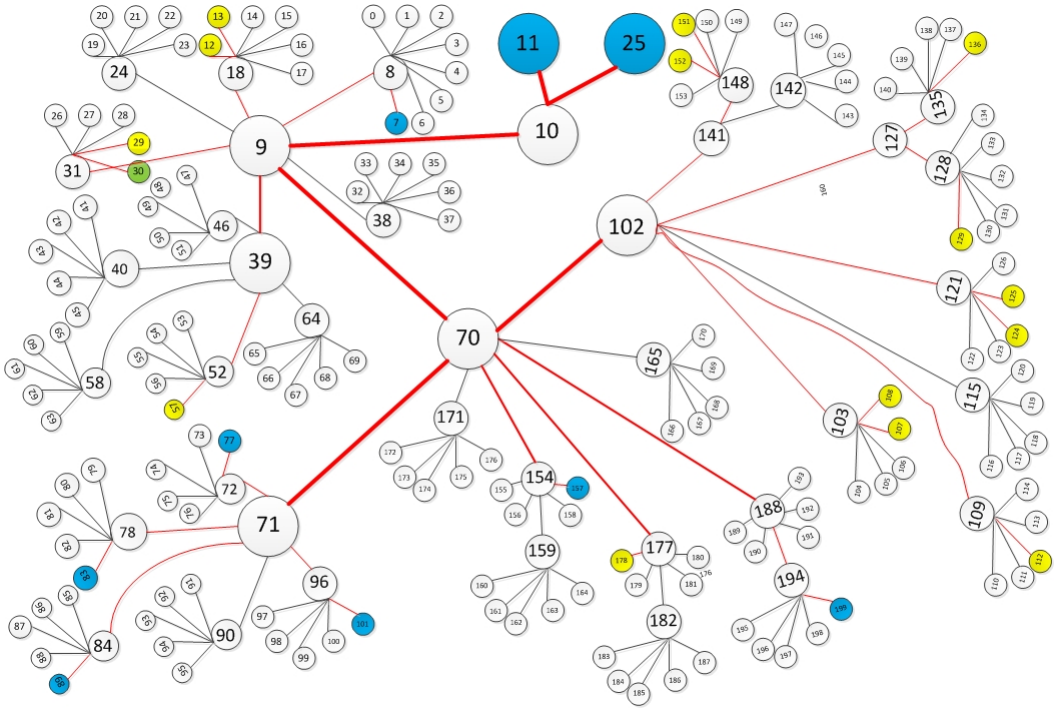


Figure 4.12: Mapping of dominant traffic flows in the network, with sampling windows of 2 seconds

In this experiment, we set the traffic generation rate of traffic flows 9 and 10 higher than any other traffic

source, thus creating congestion on nodes 18, 9 and 10 in Figure 4.12. Our reference traffic flow 79 shares the same node buffer 9, which creates anti-correlation between traffic flow 79 and 9 (-0.17408), flow 79 and 10 (-0.18157). As a result, we can see that traffic flows 9 and 10 become dominant, due to sharing the same buffer in node 9 with the reference flow 79. We find minimum cross-correlation coefficients among traffic flows 9, 10 and 79, with sampling windows of 2 seconds.

We can see from Table 4.2 that all dependent traffic flows have either the same destination as reference flow 79 (22) or they share the same node buffer 9 (flows 19, 21, 43, 75, 76, 91 and 129). Among these flows, 43, 91 and 129 have small delay distributions. Thus, their dependencies are captured by sampling windows of 2 seconds. Flows 75 and 76 have larger delay distributions than flow 79, and they share the common node buffer 9. Consequently, their dependencies are also captured by sampling windows of 2 seconds. We can also see that sampling windows of 2 seconds can capture the dependencies of some traffic flows,namely 88, 89, 97 and 108, that share common intermediate node buffers (70 and 102). They also have larger delays than reference flow 79, but they have minimum cross-correlation coefficients in relation to flow 79, due to sharing the common buffer 70. Thus, their dependencies are also captured by sampling windows of 2 seconds. Another reference node, 22, shares the same node buffers 70 and 9 as flow 107, which itself has small delays. As a result, flows 22 and 107 have a minimum cross-correlation coefficient (-0.22), with sampling windows of 2 seconds, and we can see that flow 107 exhibits the shortest distance with flow 22 in the spanning tree. Here, we can see that traffic flows with small delays become dominant within the network. We also find that some flows have larger delays than the reference flow 79 as dominant flows. In addition, they have minimum cross-correlation coefficients with flow 79, which is why we can capture their dependencies with sampling windows of 2 seconds.

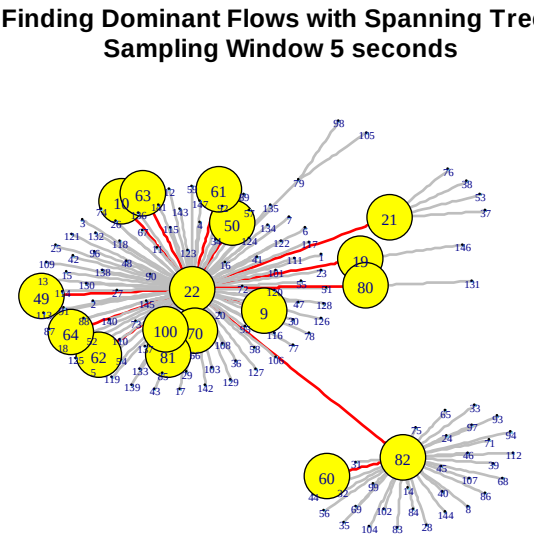


Figure 4.13: Spanning tree for finding dominant flows, with sampling windows of 5 seconds

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
22	83	71,70,9	30
9	12	9,10	11
10	13	9,10	25
19	27	9,10	11
21	29	9,70,71	77
49	65	39,9,70,102	108
50	66	39,9,70,102	114
60	80	71,70,9	25
61	85	71,70,9	11
62	86	71,70,9	25
63	87	71,70,9	30
64	89	71,70,9	7
70	99	71,70,102	120
80	113	102,70,71	77
81	114	102,70,9,39	63
82	116	102,70,9	30
100	139	102,70,9	25

Table 4.3: Source destination pairs of dominant traffic network flows, with sampling window 5 seconds

Traffic flows 22 and 82 are the reference flows of all the dominant flows in the network, with sampling windows of 5 seconds, as we can see from Figure 4.13. It also shows that traffic flow 60 is dependent on traffic flow 82. Now, we shall explore how these dominant flows are dependent on the reference traffic flows 22 and 82.

We know that traffic flows 9 and 10 have higher traffic rates and share the same buffer at node 9 with reference traffic flow 22. Thus, they have a negative correlation with traffic flow 22 and are identified as dominant flows, with sampling windows of 5 seconds. Traffic flows 19 and 21 also share the same buffer at nodes 9 and 70 with reference flow 22, so their fluctuations have been captured by sampling windows of 5 seconds and identified as dominant traffic flows. Traffic flows 49, 50, 61, 62, 63 and 64 share common node buffers 70 and 9 with reference node 22. All of these flows in turn share common node buffer 9 with flow 22, as they have a delay distribution of around 5$\sim$8 seconds. We found their peak cross-correlation coefficients with flow 22 using sampling windows of 5 seconds, so we can deduce that they are closer to reference flow 22 in the spanning tree.

Traffic flow 70 shares the same buffer with flow 22 at nodes 71 and 70. Similarly, traffic flows 80, 81, 82 and 100 share buffers at 70 and 9 with reference traffic flow 22. These traffic flows exhibit minimum negative cross-correlations with traffic flow 22 and become dominant in the network, due to sharing the same resources with the reference traffic flows.

Figure 4.13 shows that traffic flow 60 is dependent on flow 82. It shares the node buffer 70 and 9 with traffic flow 82 and has minimum cross-correlation (-0.20).

In this section, we note that most of the traffic flows that depend on reference node 22 share the most common node buffer 9, which means that traffic flows which share the same resources closer to the reference node's destination (node 30) become dominant in the network, with smaller sampling windows of 5 seconds.

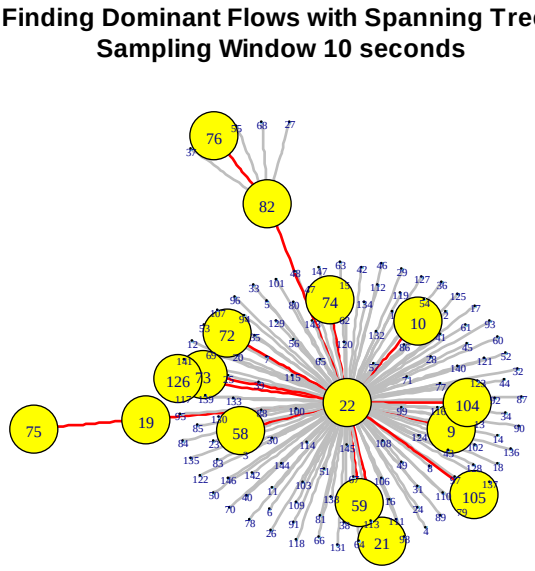


Figure 4.14: Spanning tree to find dominant flows with sampling windows of 10 seconds

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
22	83	71,70,9	30
9	12	9,10	11
10	13	9,10	25
19	27	9,10	11
21	29	9,70,71	77
58	77	71,70,9	30
59	79	71,70,9	11
72	104	102,70,9	11
73	105	102,70,9	25
74	106	102,70,9	11
75	107	102,70,9	25
76	108	102,70,9	7
82	116	102,70,9	30
104	146	102,70,71	77
105	147	102,70,9	25
126	174	70,102	120

Table 4.4: Source destination pairs of dominant traffic network flows with sampling windows of 10 seconds

We have identified 15 dominant traffic flows by sampling windows of 10 seconds. Here, the reference flow is 22, while traffic flows 75 and 76 depend on traffic flows 19 and 82 accordingly. Traffic flows 9, 10 and 19 are dominant, as they share the same node buffer 9 with traffic flow 22. We find that they have minimum cross-correlation coefficients with reference flow 22, using sampling windows of 10 seconds.

Traffic flows 58, 59, 72, 73, 74 and 126 have delays of up to 15 seconds, and we find that they have minimum cross-correlations with flow 22, using sampling windows of 10 seconds. They also are closer to reference flow 22 in the spanning tree. Traffic flows 104 and 105 have delays of around 22 seconds, so we expect to find minimum cross-correlation coefficients, using sampling windows of 20 seconds.

Dominant traffic flows 21, 58 and 59 share buffers of node 71, 70 and 9 with traffic flow 22. Traffic flows 72, 73 and 74 share buffers of 70 and 9 with traffic flow 22. All of these dominant flows commonly share buffers of node 9 and 70 with traffic flow 22. On the other hand, dominant flows 104 and 105 share nodes 71 and 70 with traffic flow 22, while flow 126 shares only node buffer 70 with flow 22. Here, we can see that a sampling window of 10 seconds (larger than the 5-seconds sampling window) starts to capture traffic flow dependencies which share intermediate nodes with the path of traffic flow 22.

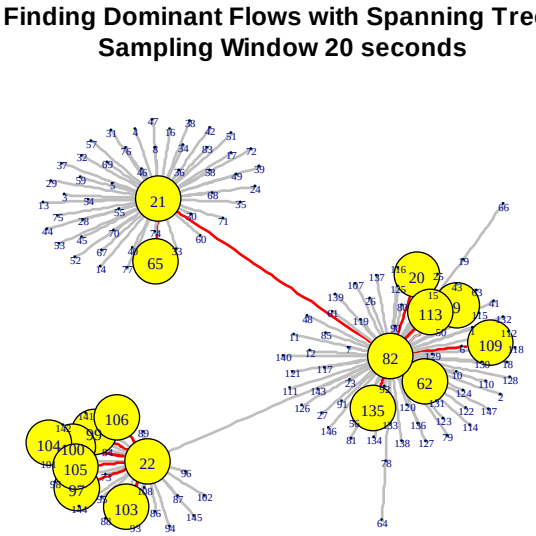


Figure 4.15: Spanning tree for finding dominant flows, with sampling windows of 20 seconds

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
82	116	102,70,9	30
9	12	9,10	11
20	28	9,10	25
21	29	9,70,71	77
22	83	71,70,9	30
62	86	71,70,9	25
65	91	71,70,9	108
97	136	102,70	157
99	138	102,70	199
100	139	102,70,9	25
103	145	102,70	199
104	146	102,70,71	77
105	149	102,70,71	77
106	150	102,70	199
109	153	102,70,9	30
113	158	70,9	7
135	185	70,9	30

Table 4.5: Source destination pairs of dominant traffic network flows, with sampling windows of 20 seconds

It is evident that there are 15 dominant flows with three reference traffic flows, namely 82, 21 and 22, in Figure 4.15. Flow 82 has a large delay distribution, so we expect to capture its dependencies with large sampling windows (sampling windows of 20 seconds and upwards). Here, reference traffic flow 82 shares common node buffers 70 and 9 with flows 9, 20, 21, 109, 113 and 135. All of these dependent flows share node 9 with flow 82, which is closer to the destination (node 30) of flow 82. Reference flow 21 shares the buffers of 70 and 71 with dependent flow 65.

Reference flow 22 shares common node buffers 70 and 71 with dependent flows 97, 103, 104, 105 and 106. They have delays around 20$\sim$ 25 seconds. We find their minimum cross-correlation coefficients with flow 22, using sampling windows of 20 seconds. We also find that they are closest to reference flow 22 in the spanning tree. Flows 99 and 100 have larger delays. We expect to see more dependencies of these flows with flow 22, using larger sampling windows. As a result, as we can see, a sampling window of 20 seconds reveals the dependencies of traffic flow 22, where dependent flows share buffers of intermediate nodes along the path of traffic flow 22.

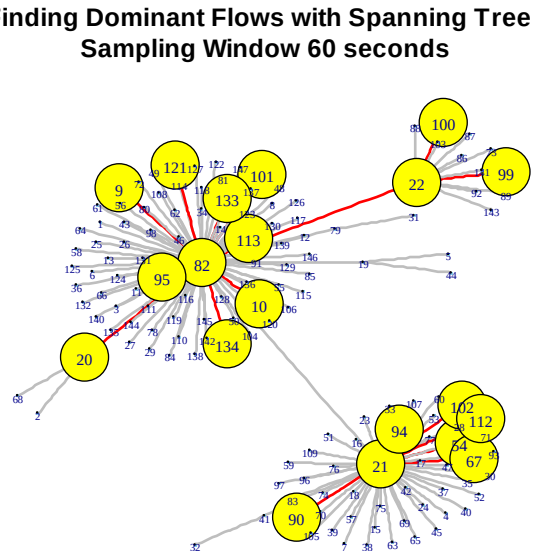


Figure 4.16: Spanning tree for finding dominant flows with sampling windows of 60 seconds

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
82	116	102,70,9	30
9	12	9,10	11
10	13	9,10	25
20	28	9,10	25
21	29	9,70,71	77
22	83	71,70,9	30
54	73	71,70,9	11
67	94	71,70,9	11
90	126	102,70,9	25
94	132	102,70,71	77
95	133	102	153
99	138	102,70	199
100	139	102,70,9	25
101	143	102,70,9	30
102	144	102,70	199
112	157	70,9	7
113	158	70,9	7
121	168	70,102	153
133	183	70,9	7
134	184	70,9	7

Table 4.6: Source destination pairs of dominant traffic network flows, with sampling windows of 60 seconds

Figure 4.16 shows that there are 18 dominant nodes organised by three reference traffic flows (flows

82,21 and 22). Reference traffic flow 82 shares a common buffer with traffic flows 9, 10 and 20 at node 9. Other dependent traffic flows 101, 113, 121, 133 and 134 share common node buffers 70 and 9 with flow 82. These flows have delays of around 80 seconds. Thus, we expect to start capturing their dependencies, by using sampling windows of 60 seconds. We also observe their dependencies, using sampling windows of 100 seconds. Here, we note that all of these dominant flows share node buffer 9 with flow 82. The remaining two dominant flows, 95 and 121, which depend on flow 82, share the common node buffers 102 and 70. We can see that our sampling window of 60 seconds captures the traffic flow dependencies on flow 82 that share the same resources closer to the destination (node 30) and also the intermediate nodes (node 102 and 70).

Traffic flows 54, 67, 90, 94, 95, 102 and 112 have delays of around 65 seconds. As a result, our technique finds their minimum cross-correlation coefficients with flow 21, by using sampling windows of 60 seconds. They also are closer to reference flow 21 in the spanning tree. Reference traffic flow 21 shares buffers of all three intermediate nodes of 9, 70 and 71 with traffic flows 54 and 67. It shares buffers of node 9 and 70 with traffic flows 90 and 112. Other two dominant flows 94 and 102 share common node buffer 70 with flow 21. Here, we can see that sampling window 60 captures the traffic flow dependencies that share commonly node buffer 70 and 71 with flow 21.

There are two dependencies (traffic flows 99 and 100) of reference flow 22. We find that flows 22 and 100 have peak cross-correlation coefficients, using sampling windows of 60 seconds. Flow 99 has delays of around 75 seconds, and we expect to get peak cross-correlation coefficients, with sampling windows of 100 seconds. These flows share the common node buffer 70 with flow 22, which is an intermediate node in the path of reference flow 22.

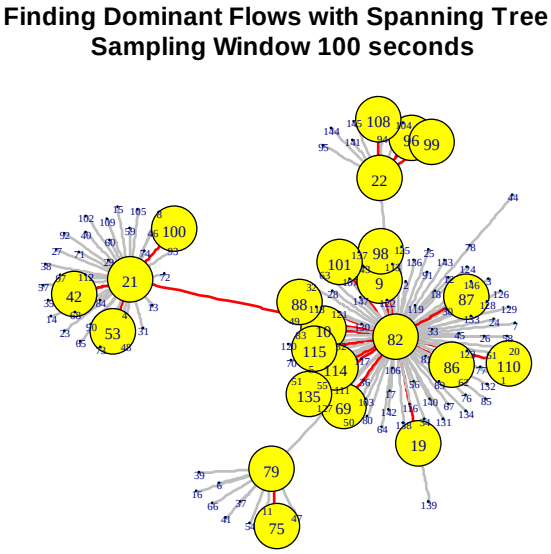


Figure 4.17: Spanning tree for finding dominant flows with sampling windows of 100 seconds

Traffic Flow (Number)	Source(Node)	Intermediate Nodes	Destination (Node)
82	116	102,70,9	30
9	12	9,10	11
10	13	9,10	25
19	27	9,10	11
21	29	9,70,71	77
22	83	71,70,9	30
42	56	39,9,70,71	83
53	69	39,9,70,102	108
69	98	71,70,102	126
75	107	102,70,9	25
79	112	102,70,9	30
86	122	102,70,71	77
87	123	102,70,71	101
88	124	102,70,71	101
96	134	102,70	157
98	137	102,70	157
99	138	102,70	199
100	139	102,70,9	25
101	143	102,70,9	30
108	152	102,70	199
110	155	70,9	11
114	160	70,9	30
115	161	70,9	30
135	185	70,9	30

Table 4.7: Source destination pairs of dominant traffic network flows, with sampling windows of 100 seconds

The number of dependencies that we calculate from the IPR is 21. Figure 4.17 shows that 21 dominant traffic flows are organised by four reference traffic flows, i.e. 21, 22, 79 and 82. Traffic flows 82 shares only node buffer 9 with traffic flows 9, 10, 19 and 21. Flows 69, 86, 87, 88, 98, 101, 110, 114, 115 and 135 have delays of around 105 seconds. Due to significant delays, they exhibit weak cross-correlation coefficients with flow 82, but our technique can still identify them as dominant in the network. We also find that they are closer to reference flow 82 in the spanning tree. In terms of the sharing buffer, flow 82 shares the common buffer of nodes 70 and 102 with traffic flows 69, 86, 87, 88, 98, 101 and 110. It also shares buffers of nodes 9 and 70 with traffic flows 114, 115 and 135. Here, we note that sampling windows of 100 seconds capture both traffic flows that share the same resources closer to its destination (node 30) and also share only intermediate nodes (node 70 and 102).

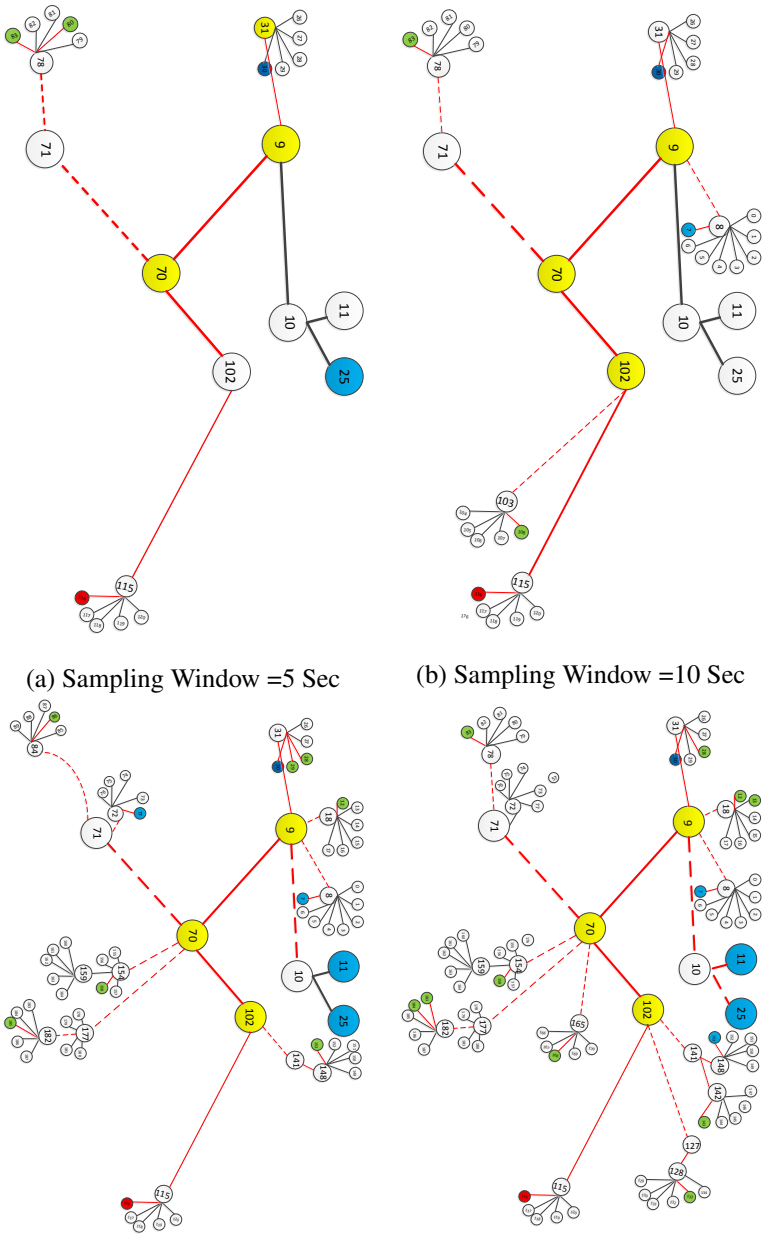
Reference traffic flows 21 shares node buffers 9 and 70 with dependent traffic flows 42, 53 and 100. Nodes 9 and 70 are the intermediate nodes for traffic flow 21, which has its source at node 29 and destination at node 77. Another reference traffic flow, 22, has three dependent traffic flows – 96, 99 and 100. Flow 22 has minimum cross-correlation coefficients with flow 99. All of these flows share a common node buffer 70 (intermediate node for flow 22) with traffic flow 22. Flow 79 has only one dependent flow 75, and they share all intermediate nodes (9,70,102) along their path.

In this section, we observe that our method can measure the traffic flow interdependencies of a LAN topology, and we also observe the significance of using different sampling windows in time series data, as it helps us to reveal more traffic flow dependencies in a network. We also find that peak cross-correlation coefficients of flows that have small delays can be found by using small sampling windows suggested by their maximum delays. Traffic flows with large delays becomes dominant flows with large sampling windows (sampling window of 20 seconds onwards). Here, flows passing through multiple congested nodes exhibit weak cross-correlation coefficients. Though they have weak cross-correlation coefficients among themselves, our technique is still able to identify them as dominant flows. The cross-correlation coefficients of these sorts of traffic flows are so weak that sometimes it is hard to determine their peak. But we find that they are closer (in terms of distance) to reference flows in spanning tree with the sampling windows suggested by their delays, and they subsequently become dominant in the network. This result is really important for constructing a functional topology, as we use cross-correlation as a means of constructing the topology.

We have already mentioned the significance of using different sampling windows in our measurements. We observe that, usually, flow dependencies that have smaller delays are captured with small sampling windows, while larger windows capture flow dependencies that have larger delays which occur mainly if a flow passes through many congested intermediate nodes. In the next section we show how different sampling windows can identify dominant flows (with different delays), by using different sampling windows.

4.4 Evolution of measuring traffic flow interdependencies with different sampling windows

Sharing the same resources (e.g. buffer of nodes) is one of the main ways of creating correlations between traffic flows. The probability of getting highly correlated traffic flows is high if the network is congested. Here, we should emphasise that each time a traffic flow passes through a congested node, its delay distribution spreads. Traffic flows that share the same one or two congested nodes have small delay distribution and their dependencies can be measured by small sampling windows. We need to use large sampling windows in order to measure the traffic flow dependencies that share multiple intermediate nodes along their path and have large delay distributions. We observe this requirement when we measure the traffic flow interdependencies of local area network. In order to show the significance of using different sampling windows in our measurement, we consider traffic flow 82 as an example and show its evolution of revealing dependencies with different sampling windows.



(a) Sampling Window =5 Sec

(b) Sampling Window =10 Sec

(c) Sampling Window =20 Sec

(d) Sampling Window =60 Sec

Figure 4.18: Spanning tree to find dominant flows with sampling windows of 10 and 20 seconds

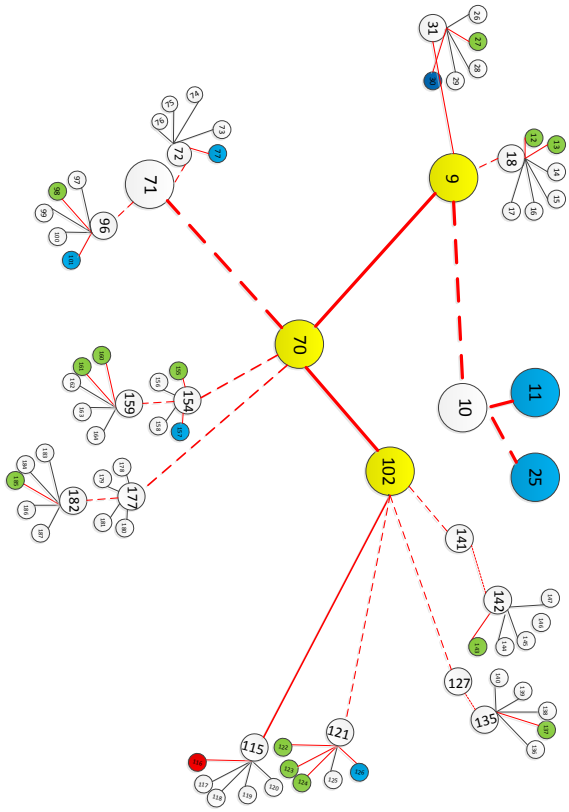


Figure 4.19: Spanning tree to find dominant flows, with sampling windows of 300 seconds

Traffic flow 82 is a long traffic flow that passes three main congested intermediate nodes, namely 102, 70 and 9. The destination of this traffic flow is node 30, and the nearest intermediate node to its destination is node 9. Traffic flow 82 has a maximum delay of around 120 seconds.

We do not see traffic flow 82 as the dominant node with a sampling window of 2 seconds, but it does become dominant while using sampling windows of 5, 10, 20, 60 and 100 seconds. We can see all the traffic flows that depend on traffic flow 82 either have the same destination or share the same node buffer 9 (which is closer to its destination) with sampling windows 5, 10 and 20 seconds, as shown in Figure 4.18a, 4.18b, 4.18c. These dependent dominant flows have small delays, and their peak cross-correlation coefficients are found with smaller sampling windows (up to 20 seconds). They are also closer to reference flow 82 in the spanning tree, with smaller sampling windows. A sampling window of 60 seconds not only reveals the dependencies of traffic flow 82 that pass thorough one or two congested nodes, but also the traffic flows that share the buffer of intermediate nodes along its path. We can see traffic flows 95 and 121 become dominant nodes, and they share the same buffer as intermediate nodes 70

and 102, as we can see from Figure 4.18d. Here, we can mention that traffic flow 95 only shares the same buffer as intermediate node 102 with traffic flow 82. The phenomenon continues as we use a sampling window of 100 seconds. It also reveals that traffic flows 69, 86, 87, 88 and 98 are dominant, which share only intermediate nodes 70 and 102 with the reference traffic flows 82, as shown in Figure 4.19. All of these flows have delays of around 100 seconds. They have weak cross-correlation coefficients with flow 82, but our technique can still identify them as dominant flows. These important results are fed into the algorithm to construct a functional topology.

4.5 Summary

The results of our interdependency measurement are really important for constructing a functional network topology. In order to obtain an accurate functional topology, we need to devise a complete set of traffic flow interdependencies in a network. We apply our method to measure traffic flow interdependencies in two large networks. We use different sampling windows suggested by maximum delays of flows, which produce peak cross-correlation coefficients, find dominant network flows and match with the IPR values, in order to measure traffic flow interdependencies. We had to overcome the challenge of whether our method could identify dominant flows from flows passing through multiple congested nodes and exhibiting weak cross-correlation coefficients. In this chapter, we show that our method can achieve this task. Along with these results, we also show that using different sampling windows in time series data can significantly reveal more traffic flow interdependencies in a network.

So far, we have shown that our technique can identify dominant flows and measure complete sets of traffic flow interdependencies. The results we have obtained using this technique are important in constructing a functional network topology. As this technique is really important for our measurement, we therefore want to check whether its application is limited to interdependency measurements in networks. To check its applicability, and also for validation purposes, we apply this technique on real data obtained from a sensor network deployed in India, to identify dominant nodes. We discuss the validation and application of our technique in wireless sensor networks in detail.

Chapter 5

Practical Validation of the Methodology using Wireless Sensor Networks

---

5.1 Introduction

In the previous two chapters, we show that our method can successfully measure interdependencies of traffic flows of different networks (including LAN). A few modifications in cross correlation method such as use different sampling windows according to maximum delays of flows for constructing correlation matrices, identify dominant flows of the network help us to measure accurate dependency of traffic flows in wired network. We know that application of our method has wider spectrum. It is not only applicable for measuring interdependencies of communication networks but also it can be useful in other networks such as wireless sensor network (WSN). Sensors are energy hungry in nature with limited battery life. So, it is really important to make balance between saving energy and obtain accurate data from sensors. In order to make the right balance between saving battery life of sensors and obtain data with high accuracy, we develop an algorithm. We call this algorithm as *Adaptive Spatial Sampling technique*. We use our method to measure interdependencies of traffic flows in wired network as a part of the algorithm. This algorithm samples only a few important sensor nodes for certain amount of time and set other sensors to sleep mode in order to reduce data consumption by sensors. Reduction in data consumption eventually leads to save energy for sensors. We are aware that data reduction increases error

in our measurement. Our algorithm shows that it can reduce significant amount of data and introduces little amount of error in measurement.

In this chapter, we find that our method to measure interdependencies of network entities (e.g. traffic flows) as a part of the algorithm becomes really useful while identifying dominant sensor nodes within clusters. Here, we show that our method is capable of not only measuring interdependencies of traffic flows in wired network but also sensor nodes in WSN. It also can identify dominant flows/sensors of both the networks. The application of our method as a part of the algorithm applied in WSN provides validation of the method to measure interdependencies of network entities.

Our method has been included in the algorithm to measure interdependencies of sensor nodes and identify dominant nodes of a cluster. It exploits spatial correlations of sensor nodes to construct cross correlation matrix of each clusters. Then, it applies eigen-analysis on cross correlation matrix and use eigendecomposition and spanning tree to identify dominant sensor nodes of the cluster. After identifying the dominant nodes of clusters using our method, the algorithm samples only those dominant nodes for a certain amount of time. It sets other nodes of the cluster into sleep mode, which save energy for those nodes. As our algorithm only samples most important or dominant nodes of the clusters, it can achieve significant data reduction while introducing little inaccuracy in our measurement. In this chapter we discuss the research problem that considers in WSN, background of the problem, our modified methodology for WSN and results achieved using our algorithm in WSN.

5.2 Research problem

Wireless sensor networks (WSN) have recently gained attention as they can be used to acquire fine grained data in environmental monitoring. One of the obstacles when deploying these sensor networks is how to manage and reduce the energy consumption as many of the sensors are battery operated. In our case, main application of interest is urban air pollution monitoring.Environmental scientists need to understand and develop better pollution dispersion models by means of fine grained measurements collected using multiple sensor nodes deployed at several locations. But one of the main challenges posed by the collection of such fine grained data is due to the energy hungry nature of the gas sensors used. Hence, it is essential to adopt data collection techniques that are not only energy efficient, but also provide accurate data.

Recently, adaptive sampling has emerged as a popular technique for sensor energy management [69] as only the most important events happening in the environment are sampled. This technique gives substantial energy savings by sampling only the most relevant events and also serves as an effective data reduction strategy. Previously, a temporal sampling technique was reported in [70]. Now this previous work needs to be extended to the spatial domain. Spatial correlation patterns have been exploited in the design of adaptive sampling techniques. In real deployments, the data collection is prone to noise and the data might not reflect the "***true***" correlations amongst the various nodes. Here, we developed a technique using our methodology (for measuring interdependencies of traffic flows) that uses Random matrix theory (RMT) to filter the noise in pollution time series and to extract meaningful correlation patterns amongst different data traces collected via the sensor nodes. Here, we also show that it is possible to use the eigendecomposition of the correlation patterns to identify the ***dominant nodes***, that is, the subset of nodes needed to provide reliable data.

To evaluate the proposed technique we use Carbon Monoxide (CO) datasets (sampled at 1Hz) collected during pollution trials carried out in India.

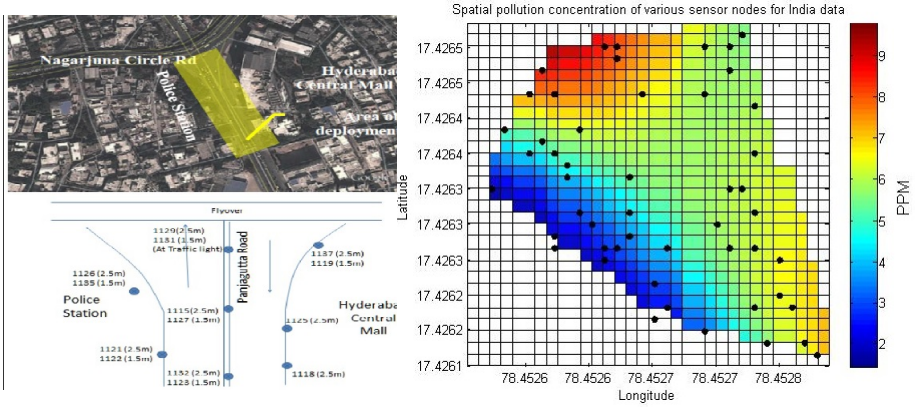


Figure 5.1: a) Map of deployed sensors b) data interpolation map

Figure 5.1 shows map of deployed sensors in India and interpolation data data map. More details about the trials can be found in [71, 72]. The pollution datasets have very strong spatial correlations and in fact, the nodes measuring similar data can be clustered together. From the cluster a dominant node can be selected as a unique data sampling node.

The main contributions of this chapter are:

1. We show how to apply our methodology derived adaptive spatial sampling technique to select the dominant sensor nodes in a cluster where the environment is noisy. The selection of the nodes is based on data correlations between different nodes
2. An empirical evaluation of the adaptive sampling technique using real, fine grained pollution datasets
3. Good performance results in terms of data reduction ($\sim 30\%$) and rms error across different parameters of the algorithm
4. It provides validation of our method to measure interdependencies of traffic flows in communication network that has discussed and applied in previous two chapters

5.3 Related work

Szczytowski et. al [73] introduced a technique named ASample which uses the partition of the space using Voronoi tessellation. To fulfil specified accuracy requirements, their technique removes unnecessary sensor nodes (SN) from over–sampling regions and generates additional new sampling locations in the under-sampling regions. Another algorithm that follows a similar approach to spatial adaptive sampling was introduced by Willet et. al and is called Backcasting [74]. It operates by having a small subset of the wireless sensors communicating their information to a fusion centre. Their theoretical predictions for the balance of data accuracy and energy consumption are supported by means of simulated experiments, however, they have not been tested in real datasets. Lin et al. [75] introduced a region sampling approach which begins by segmenting the network into several non-overlapping regions and performs sampling within each region. A *bottom-up partitioning* algorithm is used to partition the sensor network into regions. Each region computes the sample statistics, and sends the results back to the query node. The query node combines the partial aggregates from each region to approximate the query result. As each region offers different statistics and sensor costs, different sampling rates will be assigned to the regions. This algorithm considers only mean and median queries and does not employ temporal correlations. Also, the sampling schedule is created by a query node and is not decided in a real-time manner by the respective region heads. Lee et. al [76] developed a distributed adaptive sampling technique for sensor based autonomic systems called SILENCE. Main purpose is to reduce redundancy in raw data through selective representation, and without compromising on the accuracy of the reconstruction of the

phenomenon at the sink. *Similarity* and *correlation* in the sensed data is used on the fly to optimize the number of representative nodes reporting to the sink in a distributed manner, in both space and time domains. The proposed distributed solution enables each node to decide its state (or role) and sleeping schedule independently based on correlation and similarity of its own sampled data with that of the neighbouring nodes. This aggregated data is used to determine if a node plays the role of a *representative* (REP) and, consequently, to actively report data to the sink on behalf of a group of nodes; or to be an *associate* (ASSOC) to a REP and sleep. There is a high message overhead involved in this technique due to large number of control messages between the various nodes.

It is clear that the trend in spatial sampling methods is to use the correlations in the data, spatial and temporal, to improve the energy consumption of the network without being detriment in the data's quality. What we propose here is to use the cross–correlation between the data collected by a cluster of nodes to decide which node or nodes provide "good enough" data. This selection of dominant nodes can be used to save energy by, for example, send the other nodes in the cluster to sleep. The main highlight of the proposed methodology in this work is that it can be used to filter data-noise and choose the representative nodes in a cluster-based manner. We have further validated our methodology using real dataset.

5.4 Adaptive spatial sampling technique

This section gives a description of algorithm of adaptive spatial sampling technique. The technique we proposed here, works in a clustered environment where the hierarchical agglomerative clustering [77] based on data correlations is used to perform the node grouping and extract the spatial distribution patterns from a spatially interpolated pollution field. The detailed algorithm for the adaptive spatial sampling technique is as follows:

We modify our methodology of measuring interdependencies of traffic flows before we apply it in developing adaptive sampling technique. For measuring interdependencies of traffic flows, we use growth of traffic g_{ij} to calculate correlation coefficients and construct correlation matrix.The growth of traffic can be expressed as

$$g_{ij} = \ln[\frac{f_{ij}(t+\tau)}{f_{ij}(t)}] \tag{5.1}$$

Algorithm : Algorithmic description of the adaptive spatial sampling technique

1: define the duration of the time window (full time cycle), w
2: collect time data $d_i(t)$, $t \in (0,w)$ for the $i = 1,\ldots n$ nodes
3: evaluate the cross–correlation matrix $\mathbf{C}$ with elements $C_{ij} = (\langle d_i d_j \rangle - \langle d_i \rangle \langle d_j \rangle)/(\sigma_i \sigma_j)$. Where $\langle d_k \rangle$ denotes temporal average and σ_k the standard deviation for node k.
4: find the eigenvalues $\{\lambda_1,\ldots,\lambda_n\}$ and corresponding eigenvectors $\{e_1,\ldots,e_n\}$ of the correlation matrix $\mathbf{C}$.
5: discard all the eigenvectors where $\lambda_u \in [\lambda_-,\lambda_+]$ and $\lambda_\pm(Q) = 1 + \frac{1}{Q} \pm \frac{2}{\sqrt{Q}}$. Here, $Q \equiv \frac{L}{N} > 1$ and N is the number of time series of L elements of cross–correlation matrix, $\mathbf{C}$.
6: calculate IPR using $I^k = \sum_{l=1}^{n} [u_l^k]^4$. Where$u^k$is the k-th eigenvector and l= 1,.,k are the components of the eigenvector.
7: build the simplified cross–correlation matrix $\mathbf{D} = \sum_k \lambda_k e_k e_k^T$, where the sum is over all non-discarded eigenvectors, and e_j^T is the transpose of vector e_j.
8: evaluate the spanning tree of matrix $\mathbf{D}$.
9: rank all dominant nodes according to their contribution towards eigenvectors.

We have used traffic growth to calculate correlation coefficients to make our measurement independent of volume of traffic. Here, we do not consider growth of traffic as all the sensors have same capacity. So, we take time series data obtained from deployed sensors to calculate correlation coefficient and construct correlation matrix.

In order to find the most dominant nodes within each network cluster, first the cross correlations between the data from different cluster nodes are computed for a given sampling duration. This computation is carried out on the cluster node with the maximum remaining energy levels. The sampling duration during which all the nodes sample the data is termed as the ***full time cycle***. Then the eigenvalues $\{\lambda_1,\ldots,\lambda_n\}$ and eigenvectors $\{e_1,\ldots,e_n\}$ are calculated from the cross correlation matrix, $\mathbf{C}$. Afterwards, the range of eigenvalues of random interactions that are considered as noise is defined using random matrix theory [5, 6]. All the eigenvalues within the range given by RMT are discarded and eigenvalues that represents actual interactions are considered for further analysis. After discarding all the eigenvalues within the range of RMT boundaries, another method called ***Inverse Participation Ratio*** (IPR) is applied. IPR determines how many significant nodes are contributing to a particular eigenvector [78]. This provides the number of dominant nodes that can be expected from particular eigenvector. Next, a projection of cross correlation matrix $\mathbf{C}$ is taken and a simplified cross correlation matrix $\mathbf{D}$ using eigendecomposition is constructed. This matrix provides information about the contribution of each nodes towards eigenvectors. Finally, minimum spanning tree is used to rank all the components (nodes) of the eigenvectors according to the weight of their contribution [12]. By mapping the number

of IPR and order of spanning tree, the dominant nodes in cluster are determined. Once the dominant nodes are determined, these nodes serve as the ***sampler nodes*** and are chosen to sample during the consecutive sampling duration, while the ***non-sampler nodes*** are put to sleep to save energy and reduce the amount of data sampled. This sampling duration is termed as ***adaptive time cycle***. The data from the non-sampler nodes can be reconstructed using the data from the sampler nodes using appropriate reconstruction techniques like polynomial regression. The clustered architecture is reconstructed at the end of the adaptive time cycle and nodes revert back to the full time cycle in order to collect the data.

5.5 Experiments and performance analysis

Around 1800 simulations were performed using spatially interpolated datasets. The datasets contain carbon monoxide measurements collected in pollution trails in India [71, 72]. The sampled time is 8 hours. The simulation experiments consist of positioning the data–collecting nodes randomly in a given area, then use an agglomerative clustering technique to derive the network topology. The parameters considered in the simulation are the percentages of nodes sampled, the duration of the sampling window (SW). The percentage of nodes sampled varies from 10% to 50% and the duration of the sampling window varies from 10 mins to 20 mins. The correlation threshold, which determines if two nodes belong to the same cluster, is 0.5 or 0.9.

The metrics used to evaluate the performance of the algorithm are:

1. ***Percentage of Data Reduction***: This is calculated as a ratio between the number of total sampled points to the number of actual points in the real datasets. The data reduction is obtained by averaging across the different clusters in the network.

2. ***Root Mean Square Error***: The loss in data accuracy due to fewer sampled nodes. The '***polyfit***' function from MATLAB is used to approximate the behaviour of the data collected in the dominant node using a polynomial fit. Thereafter this polynomial is use to predict the behaviour of the data for the non–sampled nodes. The root mean square error: $RMSError = (\sum_{t=1}^{n} (\hat{y}_t - y_t)^2/n)^{(1/2)}$ where, $\hat{y}_t$ is the predicted value and y_t is the real value of data points of the non–sampled nodes. This error is used as an indicator of loss of accuracy in the data collection. In a cluster we average the rms for all the non–sampled nodes.

Each experiment is repeated 30 times. The results are presented graphically showing the mean and the error bar shows one standard deviation.

5.5.1 Percentage of data reduction

The clusters size depends on the correlation threshold. High correlation threshold (≥ 0.7) generate many small sized clusters and low correlation threshold (< 0.7) generates some large and small sized clusters. For the large sized clusters, during the adaptive time cycle, our technique samples few dominant nodes and keep the rest of the nodes in sleep mode. For a threshold of 0.5, there is a 44% of data reduction as shown in figure 5.2 a). For small sized clusters generated when the threshold is 0.9 there is data reduction of 12%-25% as shown in figure 5.2b).

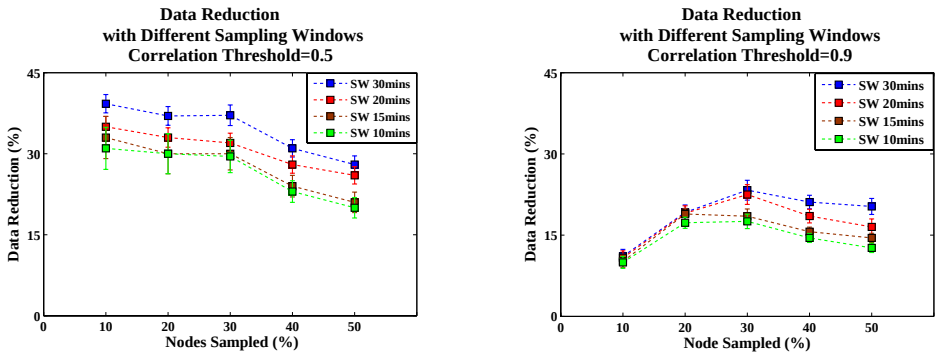


Figure 5.2: a) Percentage of nodes sample for the case of large cluster sizes b) Percentage of nodes sample for the case of small cluster sizes

Figure 5.2 shows the effect of the sampling window in the data-reduction. For short sampling windows (between 10 to 15 mins), in the adaptive-time cycle the algorithm selects different dominant nodes. The low probability that the same dominant nodes are selected every adaptive-time cycle, results in a low percentage of data reduction. With longer lasting sampling windows (20 and 30 minutes), the probability of getting the same dominant nodes is high and as a result a higher data reduction is achieved.

5.5.2 Average root mean square error

Previous section describes how percentages of data reduction can be achieved by sampling a few dominant nodes. Our technique introduces inaccuracies in the measured quantities as it does not sample all the nodes during the adaptive time cycle. This section analyse the results for root mean square error

obtained for different percentage of sampled nodes and sampling windows.

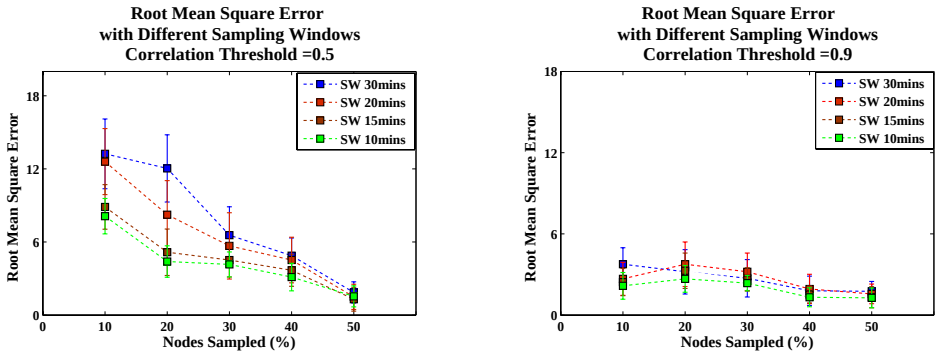


Figure 5.3: a) Average RMS error for large clusters b) average RMS error for small clusters

For large sized clusters, the data-reduction is of 30%-44% and the error is relatively high (see figure 5.3a)). Many nodes are sampled in small sized clusters and the error introduced in the data samples is lower than for large sized clusters (see figure 5.3b)). In both cases the duration of the sampling window plays an important role in the inaccuracies of the sampled data. Longer sampling windows (20 and 30 minutes) increase the errors in the data. For smaller sampling windows (10 and 15 minutes) as the sampling is over many different dominant nodes the result is smaller errors in the sampled data.

5.6 Summary

The main contribution of this chapter is the validation of our method that it can be applicable in both communication networks and also in WSN for measuring interdependencies and identify dominant network elements. Our method helps to develop a novel, real-time, adaptive spatial sampling technique for wireless sensor networks. The technique is based upon principles from time series analysis. Time series concepts enable time-efficient, scalable and decentralized implementation of the adaptive spatial sampling technique. The algorithm shows promising results in terms of data reduction and data accuracy. In the future, the algorithm needs to be made more adaptive by means of changing the clustered architecture as well as choice of dominant nodes within each time scale. For instance, the time scale is fixed and it can be made adaptive based on the spatial correlation changes.

Chapter 6

Construction of Functional Topology of Networks

6.1 Introduction

Strong knowledge on network topology is required for optimum resource utilisation, network provisioning, analyse performance of the network etc. However,this information is often unavailable due to restricted access to information (for security purposes) describing their configurations and reliability issues relating to information provisioning methods. To overcome this problem, and to gain knowledge about how resources are shared in the network, in this chapter we propose an algorithm to reconstruct a network topology from endpoint traffic measurements. All of the required data are collected at the endpoints (destinations) and they contain information about the source node, destination node and arrival time. We use the cross-correlation method, with data obtained from the endpoints, to measure traffic flow interdependencies, which are assumed to exist due to shared resources. The algorithm presented herein creates a network topology using interdependency measurements of traffic flows, which describe the connectivity of resources. We call this topology the "functional topology" of the network.

We measure interdependencies of traffic flows in network and present this dependency as means of shared resource in functional topology. We can detect the dependency between traffic flows if the network is congested (buffer utilization more than 70%). So, we can say all the nodes presented in functional topology also represent congestion level of network. Often third party stakeholder for example

CDN providers need this information, so that, they can map their infrastructure according to traffic congestion points of the network. It can help them to utilise their resource more efficiently. For peer to peer network, it can choose the peer from less congested part of network in order to avoid large delays. The topological information is also can be helpful for data center topologies for optimize resource utilization.

In this chapter, first of all, we provide a detailed description of the algorithm we create to construct a functional topology, which is then followed by an example. Then, we apply our algorithm to two different networks to construct their functional topologies, following which we compare their physical and functional topologies to understand the accuracy of our algorithm. We identify some issues in our algorithm, but we modify it to overcome these issues and produce an accurate functional topology for the network later in the chapter.

6.2 Algorithm for constructing a functional topology

The cross-correlation matrix is the main building block of our algorithm. In the previous chapters, we showed how a cross-correlation matrix is constructed with time series data collected at the destination. There are two types of cross-correlation coefficients found in a correlation matrix, namely positive and negative. In the following section, we describe why different types of correlations occur and what we can assume for each type in our algorithm. A cross-correlation matrix contains coefficients ranging from 1 to -1, where 1 represents strong correlation, -1 represents anti-correlation and 0 represents statistical independence. It is possible that noise may be introduced while calculating cross-correlation coefficients, so we need get rid of it, in order to increase the accuracy of our algorithm. We use cut-off cross-correlation coefficients to achieve this aim, as discussed in Chapter 3. Our assumptions for the algorithm are given below.

1. A *negative cross-correlation coefficient* means sharing resources. For example, two traffic flows pass through one router in their path, and when one flow passes though the buffer of the router, it blocks the path for the other flow and vice versa. This creates a negative cross-correlation between two traffic flows. Therefore, in this algorithm, we assume that if two traffic flows have a negative cross-correlation, then they share a common resource along their path.

2. The *positive cross-correlation coefficient* is related to transitivity and occurs as a result of flows

sharing resources. For example, flows f_1 and f_3 can have a transitive relationship, because they both share resources with traffic flow f_2.

3. We consider congested networks and assume that all correlation coefficients greater than the cut-off cross-correlation coefficients represent system-specific interactions.

In this algorithm, all cross-correlation coefficients larger than the cut-off cross-correlation coefficient are considered as a way of explaining shared resources. In this regard, we actually filter out all random interactions or noise from the cross-correlation matrix. We make some exceptions for considering the correlation coefficients of dominant flows in the network obtained in previous chapters, and we consider all dominant network flows regardless of their cross-correlation coefficients. After filtering out all random interactions from the cross-correlation matrix, the matrix is ready to be provided as an input into the algorithm.

We run more than 30 experiments with small topologies to check consistency of our experiment data as described in chapter 3. From those experiments, we find that our assumption mentioned above hold the ground truth. From our assumptions, we can see that negative cross-correlation coefficients constitute two flows sharing a common resource, while a positive correlation describes transitivity among traffic flows. We mentioned earlier that the functional topology explains the connectivity of resources shared amongst traffic flows in a network. To explain resource sharing, the algorithm first considers all of the negative cross-correlation coefficients, before also looking at positive cross-correlation coefficients in the correlation matrix for validation purposes. We explain the algorithm in detail below.

Algorithmic description of our algorithm is provided later in this section. There are two main parts to the algorithm, namely its initialisation and the main execution. Step 1 to 7 in algorithmic description of the algorithm belongs to initialisation part and rest of the steps (8 to 14) belongs to execution part. In the initialisation part, we first deduce anti-correlation matrix $\mathbf{A_{ij}}$ and correlation matrix $\mathbf{B_{ij}}$ from the cross-correlation matrix (step 1 to 3 shows construction of cross-correlation matrix in the algorithm). We use the cut-off correlation coefficient ∇ as a condition for filtering out noise in both matrices. If the correlation coefficients of the anti-correlation matrix are smaller than the cut-off correlation coefficients, ∇, we define that coefficient as 1, and as 0 otherwise. For the correlation matrix, we define that coefficient as 1, if the coefficient is greater than the cut-off correlation coefficient ∇, and 0 otherwise. Now we focus on the anti-correlation matrix, as coefficients of this matrix represent shared resources between

traffic flows (from the first assumption).

Four lists of resources are created for this algorithm (step 7 in algorithmic description). The first resource list, **R**, contains all of the shared resources of anti-correlation matrix $\mathbf{A_{ij}}$. Initially, this list contains a null value. The second list is called the resource per flow list, F_i, and initially it contains the sources and destination information of the traffic flow. The other two lists are called the ordered list, OF_i, and the utilisation list, UF_i, both of which have a null value initially and are derived from the resource per flow list F_i. Next, we enter into the execution part of the algorithm.

In this execution part, the algorithm first considers anti-correlation matrix $\mathbf{A_{ij}}$. It checks all of the entries (only considers elements that have a value of 1) and then numbers and appends them into resource list **R** (step 8 in algorithmic description). Here, we can mention that the numbering is done due to ordering purposes – it helps us to avoid any loop creation in the functional topology, as we maintain an order in our resource list. After creating resources list **R**, it subsets all of the resources shared by a flow and appends them into resource per flow list F_i (step 9 in the description of algorithm). Now, list F_i has all of the information about flow (source, destination and all shared resources with other flows). Next, we rearrange this flow information into source, resources shared with other flows and destination, which we place into ordered list OF_i (step 10), which in turn provides a possible connectivity of flow f_i with other flows. To check the feasibility of this connectivity, we assume that traffic flows in the network and use binary logic to evaluate flow constraints (step 11 of algorithmic description). Flow constraints come from the anti-correlation matrix, $\mathbf{A_{ij}}$, and the correlation matrix $\mathbf{B_{ij}}$. To evaluate them, we transform our ordered list into utilisation list UF_i. We assume the binary logic as follows:

if two flows share a common resource, then, one can pass and other has to wait.

Logic 0: means that the flow has to wait for other flow to pass and can not reach at the destination

Logic 1: means that flow can reach to its destination

In this way, we evaluate all of the flow constraints from both matrices. If any possible connectivity from ordered list OF_i can satisfy all of the flow constraints, then we obtain a functional topology for the network. We discuss the evaluation of flow constraints in detail in the working example 1 section. Figure 6.1 shows the flow chart of the algorithm.

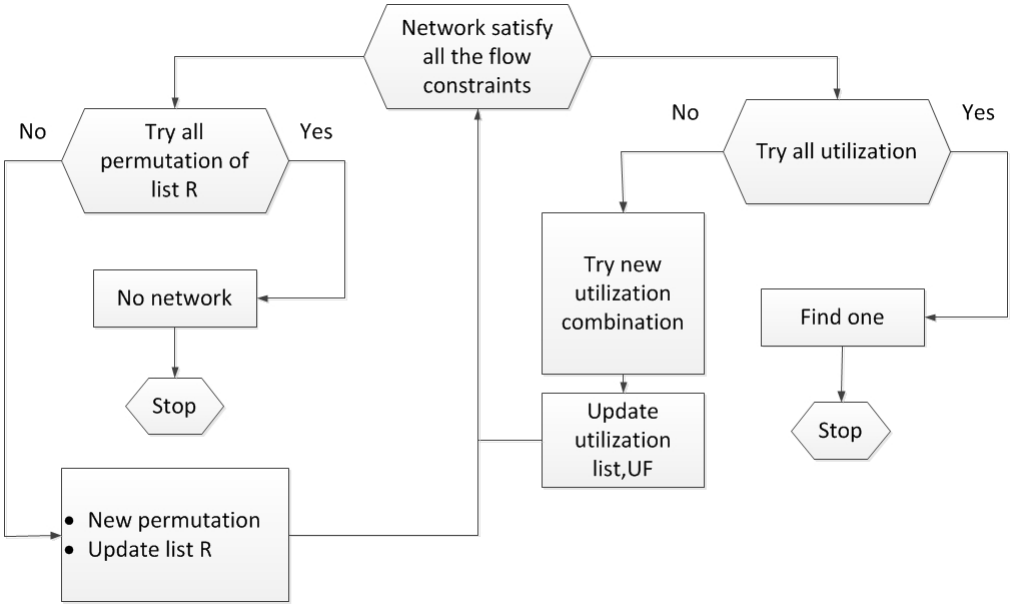


Figure 6.1: Flow chart of the algorithm

From the flow chart above (Figure 6.1), we can see the work flow of the algorithm. As mentioned earlier, we maintain an order in resource list **R** and create other lists (resource per flow F_i), by sub-setting list **R**. Ordered list OF_i is derived from list F_i and describes possible flow connectivities, f_i, with other flows. We use utilisation list UF_i to check the feasibility of this connectivity, by evaluating flow constraints. If the connectivity described by ordered list OF_i can satisfy all flow constraints, then we can get a network. If it does not satisfy all of the flow constraints, then we need to try a new utilisation combination of each network flow (see the right-hand side of Figure 6.1 and start from step 10 of the algorithm). We need to check whether any combination of utilisation list satisfies the flow constraints and then update utilisation list UF_i accordingly. If we do not get any combination in the utilisation list, then the algorithm receives a new permutation in the order of resource list **R** (left-hand side of figure 6.1 and start from step 8 of the algorithm). With the new order in resource list **R**, we can check all possible combinations of utilisation list, to satisfy flow constraints. If it can satisfy these flow constraints, then we store that particular permutation order of list **R** and utilisation list UF_i. We also update other lists accordingly. If it cannot satisfy flow constraints, then we do not get a result. Algorithmic description of the construction of a functional topology is given below. The algorithm have computational complexity on the order of $O(n^2)$.

Algorithm : Algorithmic description of the construction of a functional network topology

1: define the duration of sampling window, Δt
2: collect time data $d_i(t)$, $t \in (0, w)$ for the $i = 1, \ldots n$ traffic flows
3: evaluate the cross–correlation matrix $\mathbf{C}$ with elements $C_{ij} = (\langle d_i d_j \rangle - \langle d_i \rangle \langle d_j \rangle)/(\sigma_i \sigma_j)$. Where $\langle d_k \rangle$ denotes temporal average and σ_k the standard deviation for node k.
4: Choose the cut-off cross-correlation coefficients, ∇
5: Construct anti-correlated matrix, $\mathbf{A_{ij}}$,

$$\mathbf{A_{ij}} = \begin{cases} 1, & \text{if } C_{ij} < -\nabla \\ 0, & \text{otherwise} \end{cases}$$

6: Construct correlated matrix, $\mathbf{B_{ij}}$,

$$\mathbf{B_{ij}} = \begin{cases} 1, & \text{if } C_{ij} > \nabla \\ 0, & \text{otherwise} \end{cases}$$

7: Construct the lists:

- List 1: list has all of the resources used by a flow.
  For example, $f_i = \{s_i, d_i\}$ (initially)
- List 2 (OF_i): list has ordered resources used by flow
  For example, $f_i = \{\emptyset\}$
- List 3 (UF_i): list contains if the resources are used by all flows or not.

8: Create resource per flow vector,$\mathbf{R}$ using matrix, $\mathbf{A}$
if A_{ij} = 1, then append vector $\mathbf{R}$ with new resource r_n
For example, $\mathbf{R} = \{\mathbf{r_a}, \mathbf{r_b}, \mathbf{r_c}, ..., \mathbf{r_n}\}$
9: Update list F_i
if A_{ij} = 1, then append list F_i with new resource r_n
For example, $F_i = \{s_i, d_i, r_a, r_b, r_c, ..., r_n\}$
10: Update list OF_i according to list $\mathbf{R}$ with source at the beginning and destination at the end
For example, $OF_i = \{s_i, r_a, r_b, r_c, d_i\}$
11: Evaluate utilisation vector (UF_i), choose which flows are arriving at the destination
if $\mathbf{A_{ij}}$ = 1 and UF_i = 1, then UF_j = 0
For example, there are three flows in the network
$OF_1 = \{s_1, r_a, r_b, d_1\}$ = $UF_1 = \{1, 1, 1, 1\}$
$OF_2 = \{s_2, r_a, d_2\}$ = $UF_2 = \{1, 0, 0\}$
$OF_3 = \{s_3, r_b, d_1\}$ = $UF_3 = \{1, 0, 0\}$
12: If there are cases not defined, use values from matrix, B_{ij} or define them U_n = 1
13: Assign flows per resources
14: Create edge list from list OF_i and plot graph of functional network topology

We will now describe our algorithm by using two working examples. We consider small networks for these working examples and thereafter apply the algorithm to a large topology to create a functional topology.

6.3 Working example 1

In this working example, we consider a network consisting of four traffic flows. We consider a heavily congested network (utilisation of shared buffers at more than 70%). Traffic flows 1 and 2 share common buffers of node 2, traffic flows 1 and 3 share common buffers of node 4 and traffic flows 2 and 4 share buffers of node 5. As traffic flows share (congested) resources in the network, we expect to see strong negative cross-correlation coefficients among traffic flows. We also expect to see transitivity due to two pairs of flows sharing common resources. Our algorithm takes inputs from the cross-correlation matrix and creates lists of resources, as described in the earlier section. It checks the feasibility of the connectivity of shared resources by evaluating flow constraints. If it satisfies all of the flow constraints, then we get a functional network topology.

In this thesis, we assume that we do not have any prior knowledge of physical network. But in order to validate our results of constructing functional topology, later in this section, we compare the differences between physical and functional topologies. As we discuss earlier that functional topology cannot be matched exactly with physical topology as it presents only the important nodes that causes dependency among traffic flows. This comparison is really important to understand the difference between physical topology and functional topolology. The physical topology of the network is given in Figure 6.2.

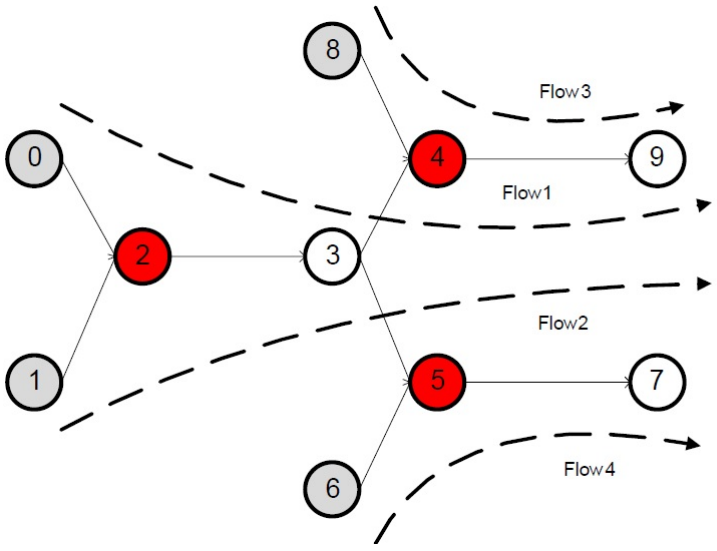


Figure 6.2: Network topology:working example1

First, we discuss the initialisation part of the algorithm. The cross-correlation matrix, **C**, for this network is given below:

$$\mathbf{C} = \begin{pmatrix} 1.000 & -0.3112 & -0.8706 & 0.1215 \\ -0.3112 & 1.000 & 0.1301 & -0.8688 \\ -0.8706 & 0.1301 & 1.000 & -0.0012 \\ 0.1215 & -0.8688 & -0.0012 & 1.000 \end{pmatrix}$$

From the results in the previous chapter, we determine that cut-off cross-correlation ∇ is 0.01. Next, we separate the anti-correlation and correlation matrices from the cross-correlation matrix **C**. If the entry of the matrix if less than ∇, then we replace the value with 1 and store it in the anti-correlation matrix. Similarly, if the entry value of the correlation matrix is greater than ∇, then we replace it with 1 and store it in the correlation matrix. The anti-correlation and correlation matrices are given below:

Anti-correlation matrix,

So, the anti-correlation matrix is:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Here, we set the diagonal value to zero.

Correlation matrix (after setting the diagonal at zero):

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

All of the lists (**R**, F_i, OF_i, UF_i) have null values, as we mentioned in the initialisation part of the algorithm. In the execution part of the algorithm, we first create resource list **R**, using the anti-correlation matrix if A_{ij} = 1. For this network, list **R** contains all of the shared resources used by the four traffic

flows.

$$\mathbf{R} = \{\mathbf{r_a}, \mathbf{r_b}, \mathbf{r_c}\}$$

After creating resource list **R**, we create list F_i. List F_i contains the source, destination and all of the shared resources used by flow f_i in the network. List F_i for this network is given below:

$$F_1 = \{s_1, d_1, r_a, r_b\}$$
$$F_2 = \{s_2, d_2, r_a, r_c\}$$
$$F_3 = \{s_3, d_1, r_b\}$$
$$F_4 = \{s_4, d_2, r_c\}$$

We order list F_i according to the order of list **R** to avoid loops in the functional network topology. The ordered list OF_i is given below:

$$OF_1 = \{s_1, r_a, r_b, d_1\}$$
$$OF_2 = \{s_2, r_a, r_c, d_2\}$$
$$OF_3 = \{s_3, r_b, d_1\}$$
$$OF_4 = \{s_4, r_c, d_2\}$$

We can see that ordered list OF_i describes a possible connectivity with different flows using shared resources. To check the feasibility of this connectivity in the ordered list, we transform it into utilisation list UF_i, in which we use binary logic to evaluate flow constraints. We derive these flow constraints from the anti-correlation matrix **A** and correlation matrix **B**. From the anti-correlation matrix, if A_{ij} = 1 and UF_i = 1, then UF_j = 0 and vice versa. From the correlation matrix, if B_{mn} = 1 and UF_m = 1, then UF_n = 1 and vice versa. By evaluating the utilisation list, if it satisfies all of the flow constraints, then we will get the functional topology of the network. Here, we take the first utilisation list and define all of its elements as 1. Then we continue to apply flow constraints to all of the other lists according to the first utilisation list. Table 6.1 shows the ordered and utilisation lists.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_a, r_b, d_1\}$	$UF_1 = \{1,1,1,1\}$
$OF_2 = \{s_2, r_a, r_c, d_2\}$	$UF_2 = \{1,0,\star,0\}$
$OF_3 = \{s_3, r_b, d_1\}$	$UF_3 = \{1,0,0\}$
$OF_4 = \{s_4, r_c, d_2\}$	$UF_4 = \{1,1,1\}$

Table 6.1: Utilisation list, UF_i for a network with four flows

We can see from the anti-corrable 6.1 that all of the elements of UF_1 are set to 1. First, we consider flow constraints from the anti-correlation matrix **A**. Flows 1 and 2 have negative correlations, which is represented by sharing common resource r_a from the ordered lists OF_1 and OF_2. Therefore, if $OF_1 = 1$, then $OF_2 = 0$. As a result, we update the list UF_2. Here, we do not know whether resource r_c has been used or not, so we denote it as not defined, $\star$. We will decide this value on the basis of other flow constraints. Flow 3 also shares the same resource with flow 1, and so we update $UF_3 = 0$ as $UF_1 = 1$. Flow 4 does not share any resource with flow 1. It only shares resources with flow 2, which has $UF_2 = 0$. Therefore, we can set $UF_4 = 1$. As we set r_c to 1 for UF_4, then it must be 0 for UF_2, so we update the table for all utilisation vectors.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_a, r_b, d_1\}$	$UF_1 = \{1,1,1,1\}$
$OF_2 = \{s_2, r_a, r_c, d_2\}$	$UF_2 = \{1,0,0,0\}$
$OF_3 = \{s_3, r_b, d_1\}$	$UF_3 = \{1,0,0\}$
$OF_4 = \{s_4, r_c, d_2\}$	$UF_4 = \{1,1,1\}$

Table 6.2: Updated utilisation list UF_i for a network with four flows

Next, we consider flow constraints from correlation matrix **B**. We can see that flow 1 is correlated with flow 4, and flow 2 is correlated with flow 3. Therefore, according to flow constraints in the correlation matrix, if $UF_1 = 1$, then $UF_4 = 1$. Table 6.2 shows us that $UF_1 = UF_4 = 1$. As flows 2 and 3 are correlated, if $UF_2 = 1$, then $UF_3 = 1$ and vice versa. From Table 6.2, we can see $UF_2 = UF_3 = 0$. As a result, it satisfies all of the flow constraints and we have a functional topology of the network, as shown in Figure 6.3.

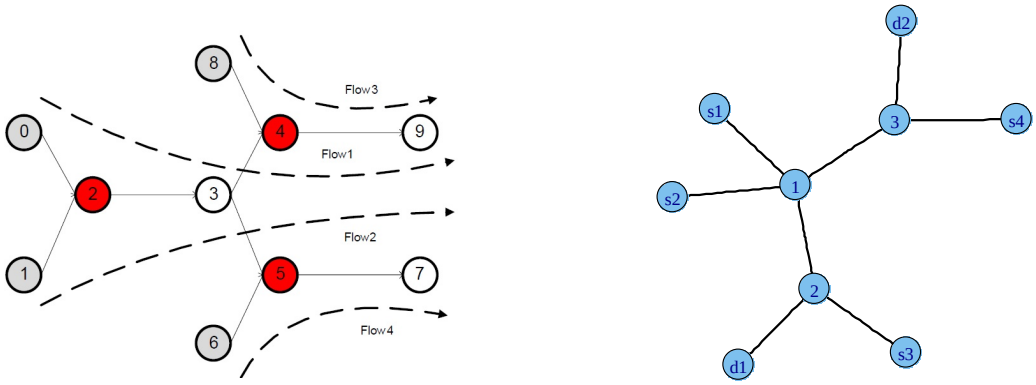


Figure 6.3: Physical topology and functional topology for working example 1

Figure 6.3 shows both the physical and functional topology of working example 1. Traffic sources 0, 1, 6 and 8 from the physical topology are represented as s1, s2, s3 and s4 accordingly in the functional topology. Destination nodes 9 and 7 are represented by d1 and d2 in the functional topology. We can see node 1 is the common resource shared by flows 1 and 2, generated by s1 and s2 accordingly. We can also see that flow 1 shares node 2 as a common resource with flow 3 (generated from s3) before reaching destination d1. It shows that flow 2 also shares node 3 with flow 4 (generated from s4) and both flows terminate at d2. We note that node 3 in the physical topology has no influence on the functional topology, as it is an over-provisioned node and has no effect on creating correlations between flows. As the node does not have any function on the flows, the functional topology ignores it. From the comparison stated above, we can see our algorithm accurately constructing source, destination and important intermediate (nodes that have effects on flows) resources. From this analysis, we can see that our algorithm has successfully constructed a functional topology in working example 1.

6.4 Working example 2

In this section, we consider a network with six traffic flows. The topology of the network is given in Figure 6.4. Figure 6.4 shows that there are six traffic sources and two destinations (node 12 and 13). In this network, flows 1 and 2 share the heavily congested buffer of node 2. Afterwards, flow 1 shares

common resources with flows 3 and 5 in nodes 4 and 9, respectively. On the other hand, flow 2 shares a common buffer with flows 4 and 6 in node 11, before ending at destination (node 13). Here, we can see that one flow shares one common resource with more than one flow. For example, flow 2 shares the buffer of node 11 with flows 4 and 6 and creates dependencies among these flows. Our algorithm treats the dependencies among these traffic flows individually and adds more resources to represent dependency in the functional topology. Therefore, actually, we create more resources in the functional topology than the number of resources present in the physical topology. Due to this approach of our algorithm, sometimes the utilisation list is not able satisfy all of the flow constraints. In this case, we need further investigation to check the feasibility of the connectivity produced by ordered list OF_i. We encounter these events while creating a functional topology for this working example. We also discover an accuracy problem with our algorithm and overcome the problem by modifying the algorithm, as discussed at the end of this chapter.

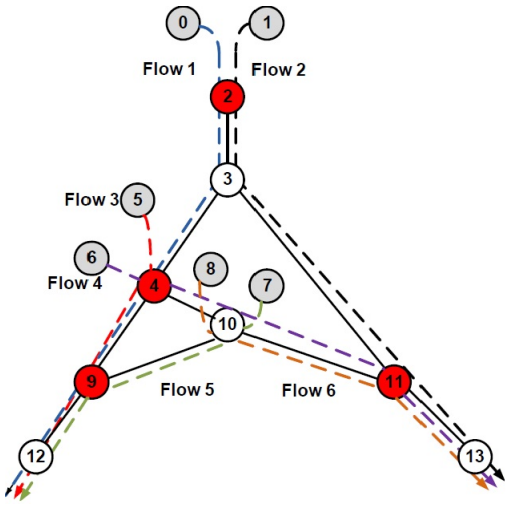


Figure 6.4: Network topology: working example 2

As mentioned earlier, we consider cut-off cross-correlation ∇ is 0.01. From Chapter 3, we can see that flows 1, 2, 3, 4 and 6 are identified as dominant network flows. Flow 5 is not dominant, as it has low traffic rates compared with flows 1 and 3. Nonetheless, it has negative correlation coefficients greater than the cut-off correlation coefficients, so we consider all six flows in our cross-correlation matrix. The initialisation part of the algorithm is given in Appendix B. Here, we start with the execution part of our algorithm, which deals with the generation of different lists, using anti-correlation matrix **A**.

Initially, all of the lists have null values. In the execution part of the algorithm, we create resource list **R**, by using the anti-correlation matrix if $A_{ij} = 1$. List **R** for this network is:

$$\mathbf{R} = \{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}, \mathbf{r_5}, \mathbf{r_6}, \mathbf{r_7}\}$$

We derive lists F_i and OF_i according to list **R**.

List, F_i	Ordered list, OF_i
$F_1 = \{s_1, d_1, r_1, r_2, r_3\}$	$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$
$F_2 = \{s_2, d_2, r_1, r_4, r_5\}$	$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$
$F_3 = \{s_3, d_1, r_2, r_6\}$	$OF_3 = \{s_3, r_2, r_6, d_1\}$
$F_4 = \{s_4, d_2, r_4, r_7\}$	$OF_4 = \{s_4, r_4, r_7, d_2\}$
$F_5 = \{s_5, d_2, r_3, r_6\}$	$OF_5 = \{s_5, r_3, r_6, d_1\}$
$F_6 = \{s_6, d_2, r_5, r_7\}$	$OF_4 = \{s_4, r_5, r_7, d_2\}$

Table 6.3: Ordered list OF_i for working example 2

Now, we evaluate the utilisation vectors derived from ordered list OF_i. As per our algorithm we set all of the elements of F_1 to 1. First, we check all of the flow constraints from the anti-correlation matrix. We assume that if the destination is 0, then we can say that flows passing through the resources should be 0. As such, we deduce our utilisation vectors.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$	$UF_1 = \{1,1,1,1,1\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,0,0,0,0\}$
$OF_3 = \{s_3, r_2, r_6, d_1\}$	$UF_3 = \{1,0,0,0\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1,1,1,1\}$
$OF_5 = \{s_5, r_3, r_6, d_1\}$	$UF_5 = \{1,0,0,0\}$
$OF_6 = \{s_4, r_5, r_7, d_2\}$	$UF_6 = \{1,1,0,0\}$

Table 6.4: Utilisation list for working example 2

Next, we check flow constraints from the correlation matrix. Correlation matrix **B** shows us that flow 2 is correlated with flows 3 and 5. We can see from the utilisation vector that UF_2, UF_3 and UF_5 are 0, which satisfies the flow constraints. From correlation matrix **B**, we also can see that flow 1 is correlated with flows 4 and 6, so UF_1, UF_4 and UF_6 should be 1. However, the utilisation vector also shows that

UF_6 is not 1. For further investigation, we set all elements of OF_2 to 1 and continue to check whether it satisfies all of the flow constraints.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$	$UF_1 = \{1,0,0,0,0\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,1,1,1,1\}$
$OF_3 = \{s_3, r_2, r_6, d_1\}$	$UF_3 = \{1,1,1,1\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1,0,0,0\}$
$OF_5 = \{s_5, r_3, r_6, d_1\}$	$UF_5 = \{1,1,0,0\}$
$OF_6 = \{s_4, r_5, r_7, d_2\}$	$UF_6 = \{1,0,0,0\}$

Table 6.5: Utilisation list, where OF_2 is set to 1 for working example 2

Table 6.5 shows the utilisation vector, where OF_2 is set to 1. Here, we can see that UF_1, UF_4 and UF_6 are 1, since flow 1 is correlated with flows 4 and 6. On the other hand, UF_2, UF_3 and UF_5 are not 1, though flow 2 is correlated with flows 3 and 5. To investigate this issue, we pick flows 1, 2, 4 and 6. From Table 6.5, we can see:

if $UF_2 = 1$

then $UF_1, UF_4, UF_6 = 0$

It means flow 2 shares resources with flows 1, 4 and 6. From Table 6.5, we can see:

if $UF_1 = 1$

then $UF_2 = 0$ and $UF_4 \vee UF_6 = 1$

This condition provides us with combinations of utilisation vectors when $UF_1 = 1$.

$UF_1 = \{1,1,1,1,1\}$	$UF_1 = \{1,1,1,1,1\}$
$UF_2 = \{1,0,0,0,0\}$	$UF_2 = \{1,0,0,0,0\}$
$UF_4 = \{1,1,1,1\}$	$UF_4 = \{1,0,0,0\}$
$UF_6 = \{1,0,0,0\}$	$UF_6 = \{1,1,1,1\}$

Table 6.6: Two combinations of utilisation vectors when $UF_1 = 1$

We now understand that due to these two sets of combinations, either UF_4 or UF_6 is 1 while $UF_1 = 1$ and $UF_2 = 0$. This satisfies the flow constraints from the correlation matrix. This analysis is also applicable to flows 2, 5 and 5, so now we have a functional network topology, as it satisfies all of the flow constraints.

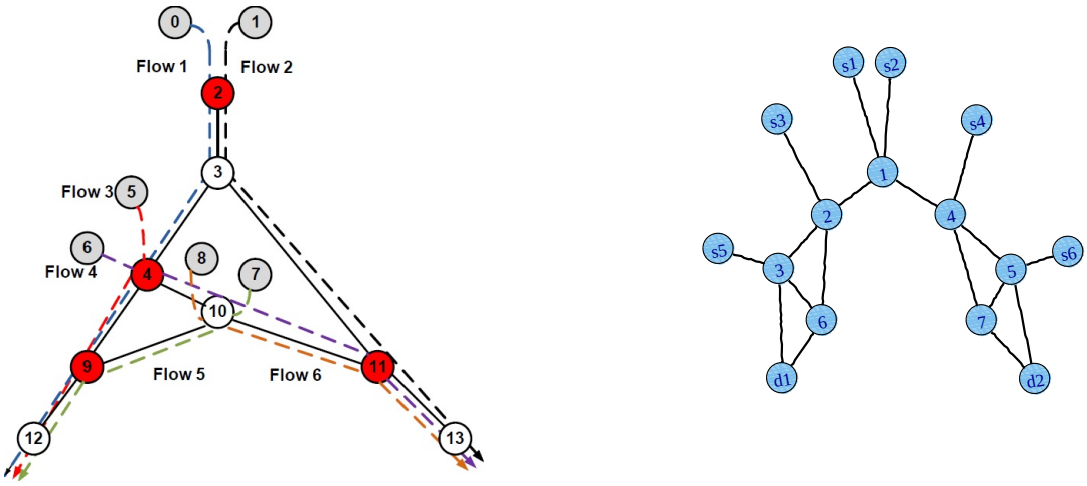


Figure 6.5: Physical topology and functional topology for working example 2

Figure 6.5 shows both the physical and the functional topology of working example 2. In the functional topology, source nodes are notated by an *s* and a flow number, while the destination nodes are labelled with a *d* and a destination number. From the physical topology, we can see that flows 1 and 2 share two common resources (nodes 2 and 3) between themselves. Here, node 3 is over- provisioned and has no effect on creating dependencies between these two flows. Therefore, our algorithm ignores this node and just shows node 1 (it represents node 2 in the physical topology) as a shared resource in the functional topology. Flows 4 and 6 share resources 10 and 11 among themselves. Our algorithm ignores node 10 in the functional topology, as it is over-provisioned. It only considers the effect of node 11 (congested buffer), which creates dependencies among flows 2, 4 and 6. As this node creates dependencies among three flows, our algorithm considers these dependencies individually. The algorithm represents node 4 of the physical topology, using three nodes (nodes 4, 5 and 7) in the functional topology. The same happens to flows 1, 3 and 5. Flows 1 and 3 share common resources 4 and 5. As both nodes are congested (utilisation around 80%), we can consider the effect of these two nodes as being similar. In this case, we can only consider node 5 of the physical topology, which we can see represents three nodes (2, 3 and 6) in the functional topology. From this analysis, we observe that our algorithm produces more resources than actual resources presented in the physical topology, where one resource is shared by more than two

flows. In this case, we need further investigation to check the feasibility of using utilisation lists. We discuss the solution (modification of the algorithm) to this problem at the end of this chapter. With a modified algorithm, we can represent one common resource shared by more than two flows instead of multiple resources in the functional topology. It also simplifies feasibility checks using utilisation lists.

6.5 Applying the algorithm to a large topology to construct a functional topology

In this section, our network consists of 17 traffic flows for applying our algorithm and constructing a functional topology. It has 17 sources and four destinations. Here, we consider a heavily congested network (buffer utilisation of nodes at around 80%) which is larger than the other two working examples discussed in the previous sections. Due to the structure and size of the network, dependencies are created in different parts of the network. Thus, our algorithm considers these dependencies and constructs a clustered functional topology, which helps us understand that a functional topology can be clustered, as not all of the parts of network are dependent on each other. This analysis is useful, and we compare each cluster in the functional topology with associated counterparts in the physical topology, the latter of which is given in Figure 6.6.

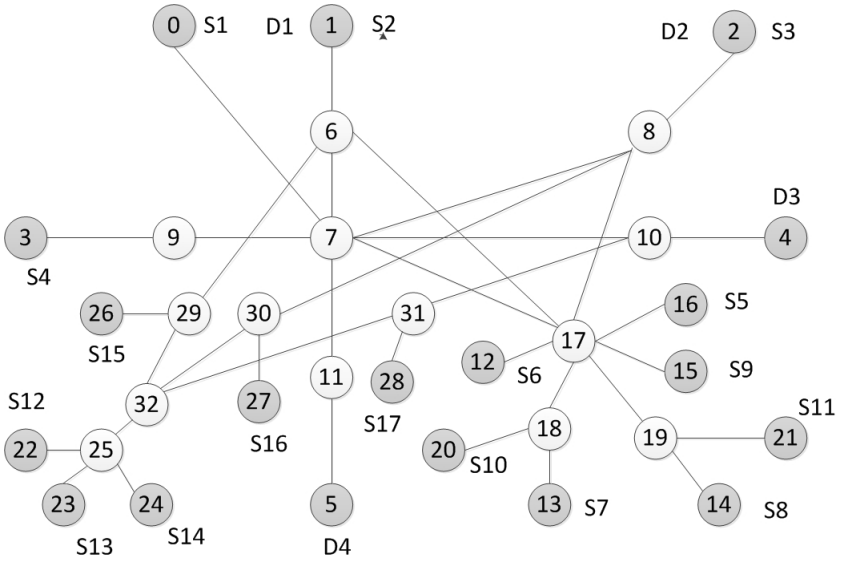


Figure 6.6: Network topology of a 17-route network

We now set the algorithm's parameters. We consider the cut-off cross-correlation threshold as 0.01 for both the negative and the positive cross-correlation coefficients. We take note of all of the dominant

flows' (obtained from interdependencies measurement chapter) correlation coefficients for creating the functional topology. For example, we consider flows 8 and 10 are anti-correlated with each other. Their minimum cross-correlation coefficients (-.0078) are found with sampling windows of 100 seconds. Correlation coefficients between flows 8 and 10 for other sampling windows (less than 60 seconds) are sometimes found less than cut-off cross-correlation coefficient. However, we consider flows 8 and 10, as they have been dominant nodes for sampling windows of 60 seconds onwards. We now create an anti-correlation matrix after considering all of these parameters. All of the lists have initially null values during the initialisation part of the algorithm. Both the anti-correlation and the correlation matrices are given in Appendix B. Here, we start from the execution part of the algorithm.

We create resource list **R** by using the anti-correlation matrix, where $A_{ij} = 1$. List **R** for this network is:

$$\mathbf{R} = \{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}, \mathbf{r_5}, \mathbf{r_6}, \mathbf{r_7}, \mathbf{r_8}, \mathbf{r_9}, \mathbf{r_{10}}, \mathbf{r_{11}}, \mathbf{r_{12}}, \mathbf{r_{13}}, \mathbf{r_{14}}, \mathbf{r_{15}}\}$$

Now, we deduce lists F_i and OF_i according to list **R**. Ordered list OF_i and utilisation list UF_i are also showing in Table 6.7.

List, F_i	Ordered list, OF_i	Utilisation list, UF_i
$F_1 = \{s_1, d_3, r_1, r_2, r_3\}$	$OF_1 = \{s_1, r_1, r_2, r_3, d_3\}$	$UF_1 = \{1,1,1,1,1\}$
$F_2 = \{s_2, d_3, r_1, r_4, r_5\}$	$OF_2 = \{s_2, r_1, r_4, r_5, d_3\}$	$UF_2 = \{1,0,0,0,0\}$
$F_3 = \{s_3, d_4, r_6, r_7\}$	$OF_3 = \{s_3, r_6, r_7, d_4\}$	$UF_3 = \{1,1,1,1\}$
$F_4 = \{s_4, d_4, r_6, r_8\}$	$OF_4 = \{s_4, r_6, r_8, d_4\}$	$UF_4 = \{1,0,0,0\}$
$F_5 = \{s_5, d_4, r_7, r_8\}$	$OF_5 = \{s_5, r_7, r_8, d_4\}$	$UF_5 = \{1,0,0,0\}$
$F_6 = \{s_6, d_1, r_{10}\}$	$OF_6 = \{s_6, r_{10}, d_1\}$	$UF_6 = \{1,1,1,1\}$
$F_7 = \{s_7, d_1, r_9, r_{11}, r_{12}\}$	$OF_7 = \{s_7, r_9, r_{11}, r_{12}, d_1\}$	$UF_7 = \{1,0,0,0,0\}$
$F_8 = \{s_8, d_2, r_{13}\}$	$OF_8 = \{s_8, r_{13}, d_2\}$	$UF_8 = \{1,1,1\}$
$F_9 = \{s_9, d_2\}$	$OF_9 = \{s_9, d_2\}$	$UF_9 = \{1,1\}$
$F_{10} = \{s_{10}, d_2, r_{11}\}$	$OF_{10} = \{s_{10}, r_{11}, d_2\}$	$UF_{10} = \{1,1,1\}$
$F_{11} = \{s_{11}, d_1, r_{10}, r_{12}, r_{13}\}$	$OF_{11} = \{s_{11}, r_{10}, r_{12}, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0,0\}$
$F_{12} = \{s_{12}, d_1\}$	$OF_{12} = \{s_{12}, d_1\}$	$UF_{12} = \{1,1\}$
$F_{13} = \{s_{13}, d_2, r_{14}\}$	$OF_{13} = \{s_{13}, r_{14}, d_2\}$	$UF_{13} = \{1,1,1\}$
$F_{14} = \{s_{14}, d_3, r_2, r_4, r_{15}\}$	$OF_{14} = \{s_{14}, r_2, r_4, r_{15}, d_3\}$	$UF_{14} = \{1,0,0,0,0\}$
$F_{15} = \{s_{15}, d_1\}$	$OF_{15} = \{s_{15}, d_1\}$	$UF_{15} = \{1,1\}$
$F_{16} = \{s_{16}, d_2, r_{14}\}$	$OF_{16} = \{s_{16}, r_{14}, d_2\}$	$UF_{16} = \{1,0,0\}$
$F_{17} = \{s_{17}, d_3, r_3, r_5, r_{15}\}$	$OF_{17} = \{s_{17}, r_3, r_5, r_{15}, d_3\}$	$UF_{17} = \{1,0,0,0,0\}$

Table 6.7: Lists for a network with 17 flows

From table 6.7, we can see that utilisation list satisfy all of the constraints from the anti-correlation matrix. Now we look up to correlation matrix and apply flow constraints on utilisation list UF_i. Correlation matrix **B** shows that flow 6,7,8,10 and 11 are correlated. We separate these flow information and apply flow constraint from the correlation matrix. We can see that flow 6,7 and 8 are correlated among themselves. First, we look at the utilisation list of flow 6,7 and 8. From table 6.7, we can see that,

$$UF_6 = UF_8 = 1$$

This time we set UF_6 to 0, to check the utilisation of the other flows.

Ordered list OF_i	Set $UF_6 = 1$	Set $UF_7 = 1$	Set $UF_{11} = 1$
$OF_6 = \{s_6, r_9, r_{10}, d_1\}$	$UF_6 = \{1,1,1,1\}$	$UF_6 = \{0,0,0,0\}$	$UF_6 = \{1,0,0,0\}$
$OF_7 = \{s_7, r_9, r_{11}, r_{12}, d_1\}$	$UF_7 = \{1,0,0,0,0\}$	$UF_7 = \{1,1,1,1,1\}$	$UF_7 = \{1,1,0,0,0\}$
$OF_8 = \{s_8, r_{13}, d_2\}$	$UF_8 = \{1,1,1\}$	$UF_8 = \{1,1,1\}$	$UF_8 = \{1,0,0\}$
$OF_{10} = \{s_{10}, r_{11}, d_2\}$	$UF_{10} = \{1,1,1\}$	$UF_{10} = \{1,0,0\}$	$UF_{10} = \{1,1,1\}$
$OF_{11} = \{s_{11}, r_{10}, r_{12}, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0,0\}$	$UF_{11} = \{1,1,0,0,0\}$	$UF_{11} = \{1,1,1,1,1\}$

Table 6.8: Updated ordered list OF_i and utilisation list UF_i for a network with 17 flows

Table 6.8 shows that

$UF_6 = UF_8 = 1$ when we set $UF_6 = 1$ and $UF_{11} = 1$

$UF_6 = UF_{10} = 1$ when we set $UF_6 = 1$ and $UF_7 = 1$

Therefore, it describes how flow 6 is correlated with flows 8 and 10.

$UF_7 = UF_8 = 1$ when we set $UF_7 = 1$ and $UF_{11} = 1$

This describes how flows 7 and 8 are correlated.

$UF_{10} = UF_{11} = 1$ when we set $UF_7 = 1$ and $UF_{11} = 1$

It describes how flows 10 and 11 are correlated.

From the above results we can see how flow constraints are satisfied in the correlation matrix, from the both correlation and anti-correlation matrix. Hence, we get a functional topology of the network.

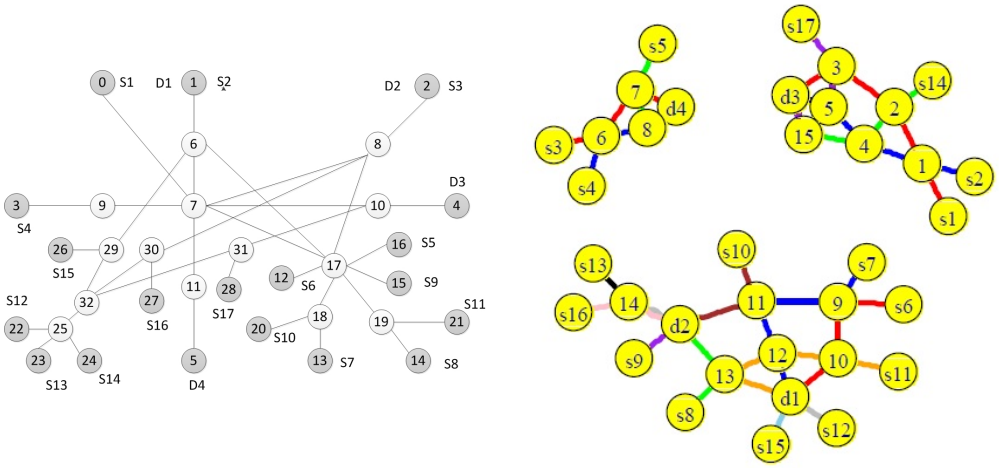


Figure 6.7: Physical topology and functional network topology consist of 17 traffic flows

Functional topology describes the dependencies of each traffic flow with other flows. If a group of flows only depend on themselves in a network, then we would see network clusters form in the functional topology. Figure 6.7 shows the functional topology of a network that has been formed into three clusters. Here, traffic flows 1, 2, 14 and 17 form one, traffic flows 3, 4 and 5 form the second cluster and the rest of the traffic flows form the third cluster. These clusters form because there are no dependencies among the clusters. For example, flows 1, 2, 14 and 17 are only dependent on each other and not dependent on other network flows. Similarly, traffic flows 3, 4 and 5 are not dependent on other network flows. Therefore, the flows which are dependent on each other form a cluster in the functional topology.

We use different colours to highlight individual flow paths in this functional topology. We mentioned earlier (while describing the functional topology for working example 2) that the representation of multiple flows sharing one common resource is different in the functional topology which shows individual flow interdependencies. We can see that traffic flows 3, 4 and 5 share one resource (that creates a negative correlation) in the physical topology, where they are represented by three nodes (6, 7 and 8) to show their dependencies separately. This happens with flows 1, 2, 14 and 17. All of these flows share a congested buffer of node 10 (that creates dependencies among themselves) in the physical topology. The algorithm considers this dependency individually (rather than collectively) and creates nodes 1, 2, 3, 4 and 5 to represent this dependency. In the physical topology, flows 6, 7 and 11 share one common resource (node 6), flows 7 and 10 share node 10 and flows 8 and 11 share node 19. All of these shared nodes are represented by nodes 9, 10, 11, 12 and 13 in the functional topology. Surely the algorithm

creates more nodes to represent the dependencies of traffic flows. We can understand the dependencies of each flow clearly with this approach, but it is difficult for visualisation purposes. In next section, we propose an extension of our algorithm, whereby we present that multiple flows sharing a common resource are plotted with a single resource in the functional topology.

6.6 Modification to the algorithm

In the previous section, we saw that flows passing through one node are represented by different nodes in the functional topology, because a functional topology describes how one flow depends on other flows in the network, even though they pass through the same node. Here, we propose an extension to the algorithm that describes how we represent the flows that pass through the same node can be represented in one node in the functional topology. The extended part of the algorithm is given below.

Algorithm : Description of the extended algorithm used to construct a functional network topology

1: Get the utilisation list UF_i of functional topology.
2: Set $UF_i = 1$ where i is the number of flows in network and separate the lists of $UF_j = 0$ where j is number of flows that share resources with flow i.
3: If there are n number of flows in the separated list, set each of flow's utilisation $UF_n = 1$ and check $UF_j = 0$. If $UF_k \neq 0$, then remove UF_k from the separated list.
4: From the separate lists where $UF_j = 0$ as $UF_n = 1$, we update ordered list of n number of flows present in separate list. We define one resource for these n number flows and update in their ordered list by removing common resources between two flows from the separate list.
5: Update utilisation list UF_i derived from newly updated ordered list OF_i.
6: Apply flow constraints from the correlation and anti-correlation matrix. If it satisfies all of the flow constraints, then we have network.
7: Create edge lists from ordered list OF_i and plot the graph.

First, we consider a small network consisting of six traffic flows, in order to demonstrate the extension of the algorithm as a working example. Afterwards, we apply it to the topology that consists of 17 traffic flows.

6.6.1 Working example

To demonstrate the functionality of the extended algorithm, we consider the network that consists of six flows and was described in the previous section as working example 2. The physical topology is shown in Figure 6.4. We get utilisation list UF_i from the previous section and illustrate it in Table 6.9.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$	$UF_1 = \{1,1,1,1,1\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,0,0,0,0\}$
$OF_3 = \{s_3, r_2, r_6, d_1\}$	$UF_3 = \{1,0,0,0\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1,1,1,1\}$
$OF_5 = \{s_5, r_3, r_6, d_1\}$	$UF_5 = \{1,0,0,0\}$
$OF_6 = \{s_4, r_5, r_7, d_2\}$	$UF_6 = \{1,1,0,0\}$

Table 6.9: Utilisation list for working example 2

As per the algorithm, we first set UF_1 to 1 and then see from Table 6.9 that:

$UF_2 = UF_3 = UF_5 = 0$

As such, we separate these four flows (including UF_1) from the utilisation list and set UF_2 =1

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$	$UF_1 = \{1,0,0,0,0\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,1,1,1,1\}$
$OF_3 = \{s_3, r_2, r_6, d_1\}$	$UF_3 = \{1,1,1,1\}$
$OF_5 = \{s_5, r_3, r_6, d_1\}$	$UF_5 = \{1,1,0,0\}$

Table 6.10: Utilisation list where OF_2 is set to 1

From Table 6.10, we can see that $UF_3 \neq 0$ while $UF_2 = 1$. Therefore, we remove UF_2 from this separate list and set UF_3 and UF_5 to 1, to check our results.

Ordered list OF_i	Set $UF_1 = 1$	Set $UF_3 = 1$	Set $UF_5 = 1$
$OF_1 = \{s_1, r_1, r_2, r_3, d_1\}$	$UF_1 = \{1,1,1,1,1\}$	$UF_1 = \{1,1,0,0,0\}$	$UF_1 = \{1,1,0,0,0\}$
$OF_3 = \{s_7, r_2, r_6, d_1\}$	$UF_3 = \{1,0,0,0\}$	$UF_3 = \{1,1,1,1\}$	$UF_3 = \{1,1,0,0\}$
$OF_5 = \{s_8, r_3, r_6, d_1\}$	$UF_5 = \{1,0,0,0\}$	$UF_5 = \{1,1,0,0\}$	$UF_8 = \{1,1,1,1\}$

Table 6.11: Separate utilisation list of flows 1, 3 and 5

From Table 6.11, we can see that if one flow from the utilisation list is set to 1, then the other two flows become 0. This means that these three flows share the same resource. Therefore, we remove the common resources shared by two flows (for example r_2 between flow 1 and 3, r_3 between flow 1 and 5

and r_6 between flow 3 and 5) and define a single resource, n_1, to these three flows. The updated resource list **R**, list F_i and ordered list OF_i are given below.

$$\mathbf{R} = \{\mathbf{r_1}, \mathbf{n_1}, \mathbf{r_4}, \mathbf{r_5}, \mathbf{r_7}\}$$

We update list F_i and OF_i according to list **R**.

List, F_i	Ordered list, OF_i
$F_1 = \{s_1, d_1, r_1, n_1\}$	$OF_1 = \{s_1, r_1, n_1, d_1\}$
$F_2 = \{s_2, d_2, r_1, r_4, r_5\}$	$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$
$F_3 = \{s_3, d_1, n_1\}$	$OF_3 = \{s_3, n_1, d_1\}$
$F_4 = \{s_4, d_2, r_4, r_7\}$	$OF_4 = \{s_4, r_4, r_7, d_2\}$
$F_5 = \{s_5, d_2, n_1\}$	$OF_5 = \{s_5, n_1, d_1\}$
$F_6 = \{s_6, d_2, r_5, r_7\}$	$OF_4 = \{s_4, r_5, r_7, d_2\}$

Table 6.12: Ordered list UF_i after replacing all common resources for flows 1, 3 and 5 with a single node

Now, we set UF_2 to 1, to find UF_j to 0 from the updated order and utilisation lists.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, n_1, d_1\}$	$UF_1 = \{1, 0, 0, 0\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1, 1, 1, 1, 1\}$
$OF_3 = \{s_3, n_1, d_1\}$	$UF_3 = \{1, 1, 1\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1, 0, 0, 0\}$
$OF_5 = \{s_5, n_1, d_1\}$	$UF_5 = \{1, 0, 0\}$
$OF_6 = \{s_4, r_5, r_7, d_2\}$	$UF_6 = \{1, 0, 0, 0\}$

Table 6.13: Utilisation list where OF_2 is set to 1 for working example 2

UF_1,UF_4, UF_5 and UF_6 become 0 while we set UF_2 to 1, as we can see from Table 6.13. We separate these utilisation flow lists for further checks. We know from Table 6.9 that $UF_4 \neq 0$ while UF_1 = 1. Therefore, we remove UF_1 from this separated list and set UF_5 to 1, to check the others.

Ordered list, OF_i	Utilisation list, UF_i
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,1,1,1,1\}$
$OF_3 = \{s_3, n_1, d_1\}$	$UF_3 = \{1,1,1\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1,0,0,0\}$
$OF_5 = \{s_5, n_1, d_1\}$	$UF_5 = \{1,1,1\}$
$OF_6 = \{s_4, r_5, r_7, d_2\}$	$UF_6 = \{1,0,0,0\}$

Table 6.14: Utilisation list where OF_2 is set to 1 for working example 2

As we can see from Table 6.14 the results remain the same and $UF_2 \neq 0$ while UF_5 = 1. As such, we remove UF_5 from this separated list. Afterwards, we set each of the utilisation lists for flows 2, 4 and 6 to 1, in order to determine if other flows become zero.

Ordered list OF_i	Set $UF_2 = 1$	Set $UF_4 = 1$	Set $UF_6 = 1$
$OF_2 = \{s_2, r_1, r_4, r_5, d_2\}$	$UF_2 = \{1,1,1,1,1\}$	$UF_2 = \{1,1,0,0,0\}$	$UF_2 = \{1,1,0,0,0\}$
$OF_4 = \{s_4, r_4, r_7, d_2\}$	$UF_4 = \{1,0,0,0\}$	$UF_4 = \{1,1,1,1\}$	$UF_4 = \{1,1,0,0\}$
$OF_6 = \{s_6, r_5, r_7, d_2\}$	$UF_6 = \{1,0,0,0\}$	$UF_6 = \{1,1,0,0\}$	$UF_6 = \{1,1,1,1\}$

Table 6.15: Separate utilisation list for flows 2, 4 and 6

From Table 6.15, we can see that if one flow from the utilisation list is set to one, the others become zero. Therefore, flows 2, 4 and 6 share one resource between themselves. We remove resources r_4, r_5 and r_7 and replace them with single node n_2 in the lists. All of the updated lists are given below.

$$\mathbf{R} = \{\mathbf{r_1}, \mathbf{n_1}, \mathbf{n_2}\}$$

We update lists F_i and OF_i according to list **R**.

List, F_i	Ordered list, OF_i
$F_1 = \{s_1, d_1, r_1, n_1\}$	$OF_1 = \{s_1, r_1, n_1, d_1\}$
$F_2 = \{s_2, d_2, r_1, n_2\}$	$OF_2 = \{s_2, r_1, n_2, d_2\}$
$F_3 = \{s_3, d_1, n_1\}$	$OF_3 = \{s_3, n_1, d_1\}$
$F_4 = \{s_4, d_2, n_2\}$	$OF_4 = \{s_4, n_2, d_2\}$
$F_5 = \{s_5, d_2, n_1\}$	$OF_5 = \{s_5, n_1, d_1\}$
$F_6 = \{s_6, d_2, n_2\}$	$OF_4 = \{s_4, n_2, d_2\}$

Table 6.16: Utilisation list, UF_i after replacing all common resources for flows 2, 4 and 6 with a single node

Now, we derive utilisation list UF_i from the updated ordered list OF_i and apply all of the flow constraints from the correlation and anti-correlation matrices. If it satisfies all of the flow constraints, we acquire a modified functional topology.

Ordered list, OF_i	Set $UF_1 = 1$	Set $UF_2 = 1$
$OF_1 = \{s_2, r_1, n_1, d_1\}$	$UF_1 = \{1,1,1,1\}$	$UF_1 = \{1,0,0,0\}$
$OF_2 = \{s_2, r_1, n_2, d_2\}$	$UF_2 = \{1,0,0,0\}$	$UF_2 = \{1,1,1,1\}$
$OF_3 = \{s_3, n_1, d_1\}$	$UF_3 = \{1,0,0\}$	$UF_3 = \{1,1,1\}$
$OF_4 = \{s_4, n_2, d_2\}$	$UF_4 = \{1,1,1\}$	$UF_4 = \{1,0,0\}$
$OF_5 = \{s_5, n_1, d_1\}$	$UF_5 = \{1,0,0\}$	$UF_5 = \{1,0,0\}$
$OF_6 = \{s_4, n_2, d_2\}$	$UF_6 = \{1,0,0\}$	$UF_6 = \{1,0,0\}$

Table 6.17: Updated utilisation vector UF_i for checking flow constraints

Flow constraints from the anti-correlation matrix are satisfied for both cases (while $UF_1 = 1$ or $UF_2 = 1$), as we can see from Table 6.17. Flow constraints from the correlation matrix describe that flow 1 is correlated with flows 4 and 6. On the other hand, flow 2 is correlated with flows 3 and 5. We can see that:

$UF_1 = UF_4 = 1$ while $UF_1 = 1$

if we set $UF_6 = 1$, then $UF_1 = UF_6 = 1$ while $UF_1 = 1$

So, either flow 4 or 6 can be correlated with flow 1, while $UF_1 = 1$, as flows 4 and 6 share the same resources. We also can see:

$UF_1 = UF_4 = UF_6 = 0$ while $UF_2 = 1$

Here, we can understand how flow 1 is correlated with flows 4 and 6. The same analogy is applicable to flows 2, 3 and 5. Therefore, it satisfies all of the flow constraints of the correlation and anti-correlation matrices. The functional topology of the network is given in Figure 6.8.

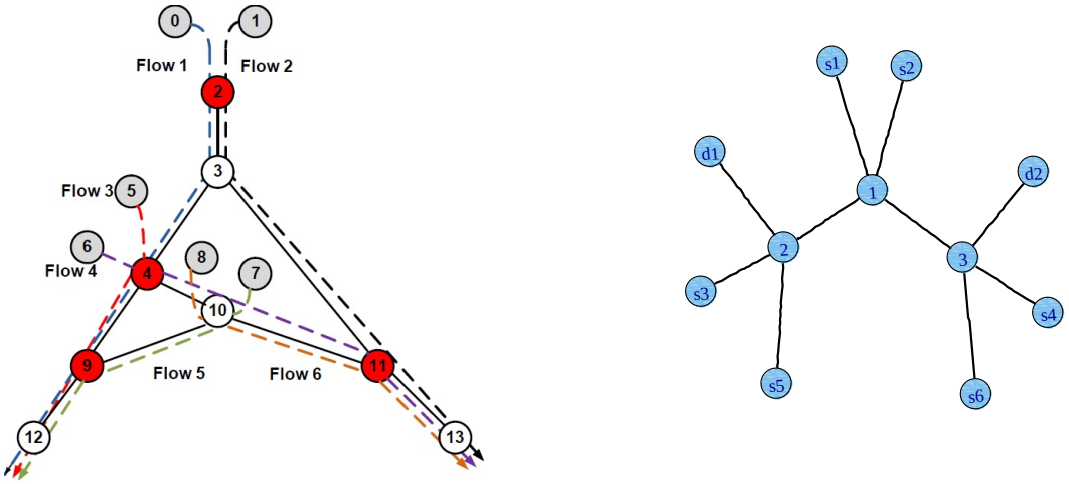


Figure 6.8: Physical topology and modified functional topology of working example 2

In this modified functional topology, we show that multiple flows sharing a common resource are presented as one node. In the previous section, we saw that flows 1, 3 and 5 share resources 2, 3 and 6 in the functional topology. By using the extended algorithm, we remove nodes 2, 3 and 6 and replace them with node 2, which creates dependencies among flows 1 ,3 and 5. Using the same method, we also remove nodes 4, 5 and 7 from the previous functional topology and replace them with three. Therefore, with the help of our modified algorithm, we replace six nodes with two nodes, to represent dependencies of traffic flows in the functional topology. The modified functional topology provides a simplified and accurate network topology that describes how the flows are dependent on others.

6.6.2 Applying the extended algorithm to a large topology to construct a modified functional topology

In the previous section, our algorithm constructed a functional network topology consisting of 17 traffic flows. In this functional topology, dependencies among traffic flows are represented by more nodes presented in the physical topology. Here, we apply our modified algorithm to the network topology consisting of 17 flows, in order to observe the accuracy of our algorithm.

Ordered list, OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, r_1, r_2, r_3, d_3\}$	$UF_1 = \{1,1,1,1,1\}$
$OF_2 = \{s_2, r_1, r_4, r_5, d_3\}$	$UF_2 = \{1,0,0,0,0\}$
$OF_3 = \{s_3, r_6, r_7, d_4\}$	$UF_3 = \{1,1,1,1\}$
$OF_4 = \{s_4, r_6, r_8, d_4\}$	$UF_4 = \{1,0,0,0\}$
$OF_5 = \{s_5, r_7, r_8, d_4\}$	$UF_5 = \{1,0,0,0\}$
$OF_6 = \{s_6, r_9, r_{10}, d_1\}$	$UF_6 = \{1,1,1,1\}$
$OF_7 = \{s_7, r_9, r_{11}, r_{12}, d_1\}$	$UF_7 = \{1,0,0,0,0\}$
$OF_8 = \{s_8, r_{13}, d_2\}$	$UF_8 = \{1,1,1\}$
$OF_9 = \{s_9, d_2\}$	$UF_9 = \{1,1\}$
$OF_{10} = \{s_{10}, r_{11}, d_2\}$	$UF_{10} = \{1,1,1\}$
$OF_{11} = \{s_{11}, r_{10}, r_{12}, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0,0\}$
$OF_{12} = \{s_{12}, d_1\}$	$UF_{12} = \{1,1\}$
$OF_{13} = \{s_{13}, r_{14}, d_2\}$	$UF_{13} = \{1,1,1\}$
$OF_{14} = \{s_{14}, r_2, r_4, r_{15}, d_3\}$	$UF_{14} = \{1,0,0,0,0\}$
$OF_{15} = \{s_{15}, d_1\}$	$UF_{15} = \{1,1\}$
$OF_{16} = \{s_{16}, r_{14}, d_2\}$	$UF_{16} = \{1,0,0\}$
$OF_{17} = \{s_{17}, r_3, r_5, r_{15}, d_3\}$	$UF_{17} = \{1,0,0,0,0\}$

Table 6.18: Ordered list, OF_i and utilisation list UF_i for a network with 17 flows

From Table 6.18, we can see that:

$UF_2 = UF_{14} = UF_{17} = 0$ while $UF_1 = 1$

$UF_4 = UF_5 = 0$ while $UF_3 = 1$

$UF_7 = UF_{11} = 0$ while $UF_6 = 1$

We separate these utilisation lists for further checks. First, we consider the utilisation list for flows 1, 2, 14 and 17.

Set $UF_1 = 1$	Set $UF_2 = 1$	Set $UF_{14} = 1$	Set $UF_{17} = 1$
$UF_1 = \{1,1,1,1,1\}$	$UF_1 = \{1,0,0,0,0\}$	$UF_1 = \{1,1,0,0,0\}$	$UF_1 = \{1,1,1,0,0\}$
$UF_2 = \{1,0,0,0,0\}$	$UF_2 = \{1,1,1,1,1\}$	$UF_2 = \{1,0,0,0,0\}$	$UF_2 = \{1,0,0,0,0\}$
$UF_{14} = \{1,0,0,0,0\}$	$UF_{14} = \{1,1,0,0,0\}$	$UF_{14} = \{1,1,1,1,1\}$	$UF_{14} = \{1,1,1,0,0\}$
$UF_{17} = \{1,0,0,0,0\}$	$UF_{17} = \{1,1,0,0,0\}$	$UF_{17} = \{1,1,1,0,0\}$	$UF_{17} = \{1,1,1,1,1\}$

Table 6.19: Utilisation vector UF_i for checking flow constraints

If one of the utilisation lists is set to 1, then the others become zero, as we can see from Table 6.19. Therefore, flows 1, 2, 14 and 17 share the same resource in the network. We remove all of the common resources between the flows $(r_1, r_2, r_3, r_4, r_5, r_{15})$ and replace them with node n_1.

Next, we consider the utilisation list for flows 3, 4 and 5.

Ordered list, OF_i	Set UF_3 = 1	Set UF_4 = 1	Set UF_5 = 1
$OF_3 = \{s_3, r_6, r_7, d_4\}$	$UF_3 = \{1,1,1,1\}$	$UF_3 = \{1,0,0,0\}$	$UF_3 = \{1,0,0,0\}$
$OF_4 = \{s_4, r_6, r_8, d_4\}$	$UF_4 = \{1,0,0,0\}$	$UF_4 = \{1,1,1,1\}$	$UF_4 = \{1,1,0,0\}$
$OF_5 = \{s_5, r_7, r_8, d_4\}$	$UF_5 = \{1,0,0,0\}$	$UF_5 = \{1,1,0,0\}$	$UF_5 = \{1,1,1,1\}$

Table 6.20: Utilisation vector UF_i flows 3, 4 and 5 for checking flow constraints

We can see from Table 6.20 that flows 3, 4 and 5 share one common resource. Therefore, common resources between two flows(r_6, r_7, r_8) are replaced with one single resource n_2.

Now, we consider the utilisation list for flows 6, 7 and 11.

Ordered list, OF_i	Set UF_6 = 1	Set UF_7 = 1	Set UF_{11} = 1
$OF_6 = \{s_6, r_9, r_{10}, d_1\}$	$UF_6 = \{1,1,1,1\}$	$UF_6 = \{1,0,0,0\}$	$UF_6 = \{1,0,0,0\}$
$OF_7 = \{s_7, r_9, r_{11}, r_{12}, d_1\}$	$UF_7 = \{1,0,0,0,0\}$	$UF_7 = \{1,1,1,1,1\}$	$UF_7 = \{1,1,1,0,0\}$
$OF_{11} = \{s_{11}, r_{10}, r_{12}, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0,0\}$	$UF_{11} = \{1,1,0,0,0\}$	$UF_{11} = \{1,1,1,1,1\}$

Table 6.21: Utilisation list UF_i for flows 6, 7 and 11 for checking flow constraints

The results in Table 6.21 indicate that flows 6, 7 and 11 share one common resource. Consequently, we remove resources r_9, r_{10}, r_{12} from list F_i and ordered list OF_i and replace them with resource n_3. The updated lists are given below.

$$\mathbf{R} = \{\mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}, \mathbf{r_{11}}, \mathbf{r_{13}}, \mathbf{r_{14}}\}$$

Now, we update list F_i and OF_i according to list **R**. Ordered list OF_i is shown in Table 6.22.

List, F_i	Ordered list, OF_i
$F_1 = \{s_1, d_3, n_1\}$	$OF_1 = \{s_1, n_1, d_3\}$
$F_2 = \{s_2, d_3, n_1\}$	$OF_2 = \{s_2, n_1, d_3\}$
$F_3 = \{s_3, d_4, n_2\}$	$OF_3 = \{s_3, n_2, d_4\}$
$F_4 = \{s_4, d_4, n_2\}$	$OF_4 = \{s_4, n_2, d_4\}$
$F_5 = \{s_5, d_4, n_2\}$	$OF_5 = \{s_5, n_2, d_4\}$
$F_6 = \{s_6, d_1, n_3\}$	$OF_6 = \{s_6, n_3, d_1\}$
$F_7 = \{s_7, d_1, n_3, r_{11}\}$	$OF_7 = \{s_7, n_3, r_{11}, d_1\}$
$F_8 = \{s_8, d_2, r_{13}\}$	$OF_8 = \{s_8, r_{13}, d_2\}$
$F_9 = \{s_9, d_2\}$	$OF_9 = \{s_9, d_2\}$
$F_{10} = \{s_{10}, d_2, r_{11}\}$	$OF_{10} = \{s_{10}, r_{11}, d_2\}$
$F_{11} = \{s_{11}, d_1, n_3, r_{13}\}$	$OF_{11} = \{s_{11}, n_3, r_{13}, d_1\}$
$F_{12} = \{s_{12}, d_1\}$	$OF_{12} = \{s_{12}, d_1\}$
$F_{13} = \{s_{13}, d_2, r_{14}\}$	$OF_{13} = \{s_{13}, r_{14}, d_2\}$
$F_{14} = \{s_{14}, d_3, n_1\}$	$OF_{14} = \{s_{14}, n_1, d_3\}$
$F_{15} = \{s_{15}, d_1\}$	$OF_{15} = \{s_{15}, d_1\}$
$F_{16} = \{s_{16}, d_2, r_{14}\}$	$OF_{16} = \{s_{16}, r_{14}, d_2\}$
$F_{17} = \{s_{17}, d_3, n_1\}$	$OF_{17} = \{s_{17}, n_1, d_3\}$

Table 6.22: Updated list F_i and ordered list OF_i for a network with 17 flows

Now, we derive a new utilisation list UF_i from the updated ordered list OF_i and apply all of the flow constraints from the correlation and anti-correlation matrices. If it satisfies all of the flow constraints, then we have a modified functional network topology. Ordered list OF_i and utilisation list UF_i are given in Table 6.23.

Ordered list,OF_i	Utilisation list, UF_i
$OF_1 = \{s_1, n_1, d_3\}$	$UF_1 = \{1,1,1\}$
$OF_2 = \{s_2, n_1, d_3\}$	$UF_2 = \{1,0,0\}$
$OF_3 = \{s_3, n_2, d_4\}$	$UF_3 = \{1,1,1\}$
$OF_4 = \{s_4, n_2, d_4\}$	$UF_4 = \{1,0,0\}$
$OF_5 = \{s_5, n_2, d_4\}$	$UF_5 = \{1,0,0\}$
$OF_6 = \{s_6, n_3, d_1\}$	$UF_6 = \{1,1,1\}$
$OF_7 = \{s_7, n_3, r_{11}, d_1\}$	$UF_7 = \{1,0,0,0\}$
$OF_8 = \{s_8, r_{13}, d_2\}$	$UF_8 = \{1,1,1\}$
$OF_9 = \{s_9, d_2\}$	$UF_9 = \{1,1\}$
$OF_{10} = \{s_{10}, r_{11}, d_2\}$	$UF_{10} = \{1,1,1\}$
$OF_{11} = \{s_{11}, n_3, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0\}$
$OF_{12} = \{s_{12}, d_1\}$	$UF_{12} = \{1,1\}$
$OF_{13} = \{s_{13}, r_{14}, d_2\}$	$UF_{13} = \{1,1,1\}$
$OF_{14} = \{s_{14}, n_1, d_3\}$	$UF_{14} = \{1,0,0\}$
$OF_{15} = \{s_{15}, d_1\}$	$UF_{15} = \{1,1\}$
$OF_{16} = \{s_{16}, r_{14}, d_2\}$	$UF_{16} = \{1,0,0\}$
$OF_{17} = \{s_{17}, n_1, d_3\}$	$UF_{17} = \{1,0,0\}$

Table 6.23: Ordered list OF_i and utilisation list UF_i for a network with 17 flows

Flow constraints from the anti-correlation matrix are satisfied by utilisation list UF_i. Now, we consider flow constraints from the correlation matrix. We can see that flows 6, 7, 8, 10 and 11 are correlated from the correlation matrix **B**.

First, we look at the utilisation list for flows 6, 7 and 8. From Table 6.23 we can see that:

$$UF_6 = UF_8 = 1$$

This time we set UF_6 to 0, to check the utilisation of the other flows.

Ordered list OF_i	Set $UF_6 = 1$	Set $UF_7 = 1$	Set $UF_{11} = 1$
$OF_6 = \{s_6, n_3, d_1\}$	$UF_6 = \{1,1,1\}$	$UF_6 = \{1,0,0\}$	$UF_6 = \{1,0,0\}$
$OF_7 = \{s_7, n_3, r_{11}, d_1\}$	$UF_7 = \{1,0,0,0\}$	$UF_7 = \{1,1,1,1\}$	$UF_7 = \{1,0,0,0\}$
$OF_8 = \{s_8, r_{13}, d_2\}$	$UF_8 = \{1,1,1\}$	$UF_8 = \{1,1,1\}$	$UF_8 = \{1,0,0\}$
$OF_{10} = \{s_{10}, r_{11}, d_2\}$	$UF_{10} = \{1,1,1\}$	$UF_{10} = \{1,0,0\}$	$UF_{10} = \{1,1,1\}$
$OF_{11} = \{s_{11}, n_3, r_{13}, d_1\}$	$UF_{11} = \{1,0,0,0\}$	$UF_{11} = \{1,0,0,0\}$	$UF_{11} = \{1,1,1,1\}$

Table 6.24: Updated ordered list OF_i and utilisation list UF_i for a network with 17 flows

We know that flows 6, 7 and 11 share one common resource between themselves. Table 6.24 shows that

UF_6 =UF_7= UF_8= 1 when we set $UF_{11} = 1$. As flows 6 and 7 are negatively correlated with each other, UF_6 = UF_8= 1, while UF_6 = 1 and UF_7 = 0. It also shows UF_7 = UF_8= 1, while UF_7 = 1 and UF_6 = 0. Therefore, it describes how flows 6 and 7 are correlated with flow 8. As flows 7 and 10 share one resource(r_{11}) with each other, flows 6 and 10 become correlated as a result of sharing the resources of flows 7 and 10. We can see that:

UF_6 = UF_{10}= 1 or 0 when we set $UF_6 = 1$ and $UF_7 = 1$ This describes how flow 6 is correlated with flow 10. As flows 7 and 11 also share the same resources along with flow 6 (n_3), flows 10 and 11 become correlated as well. It shows:

UF_{10} = UF_{11}= 1 or 0 when we set $UF_7 = 1$ and $UF_{11} = 1$

This describes how flows 10 and 11 are correlated. Here, we can see it satisfies all of the flow constraints from both the correlation and the anti-correlation matrices. The modified functional topology of this network is given in Figure 6.9.

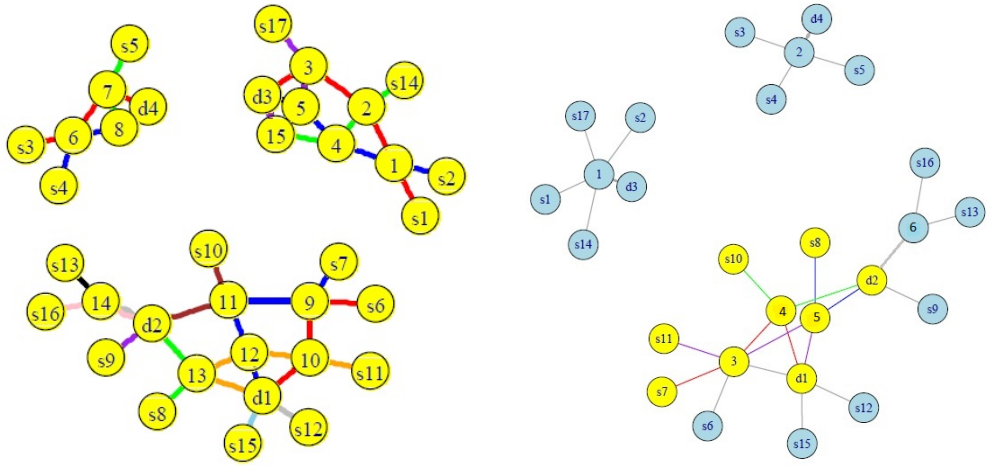


Figure 6.9: Functional topology from the previous section and a modified functional network topology consisting of 17 traffic flows

There are three clusters in the functional topology of the network. These clusters are formed due to dependencies in traffic flows. For example, traffic flows 1, 2, 14 and 17 share one resource among themselves, and this creates dependencies. These traffic flows do not depend on the rest of the flows, and as a result a separate cluster is formed.

In the previous section, we observed that multiple traffic flows sharing one common resource are represented by multiple nodes to describe individual flow dependencies. Here, we replace all of these flows

with a single resource that creates dependencies. For example, flows 3, 4 and 5 share a common resource, and they are represented using nodes 6, 7 and 8 in Figure 6.7. In terms of functions or dependency, the figure shows accurate results but it is difficult for visualisation. In the modified functional topology (see figure 6.9) we remove nodes 6, 7 and 8 and replace them with node 2. Here, it shows that flows 3, 4 and 5 share resource 2 before they terminate at destination d_4. Similarly, we replace nodes 1 to 5 from the previous functional topology and replace them with node 2 in a modified functional topology. We also replace nodes 9 to 13 and replace them with nodes 3, 4 and 5 in a modified functional topology. With the help of our modified algorithm we replace 13 resources with only six nodes in the modified functional topology. This modification helps us to simplify how we visualise the network. Now, we compare our modified functional topology with the physical topology, to check how accurately shared resources are represented in the functional topology.

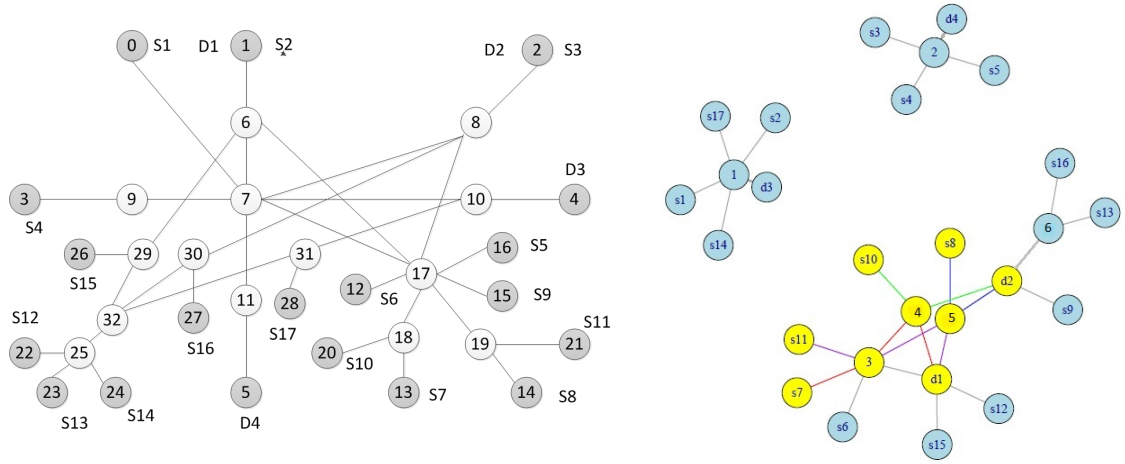


Figure 6.10: Comparison of the physical and the modified functional network topology consisting of 17 traffic flows

Figure 6.10 shows both the physical and the modified functional network topology consisting of 17 traffic flows. We can compare shared resources among traffic flows between these two topologies. Three clusters form in our functional topology. Traffic flows 1, 2, 14 and 17 belong to the first cluster and share one common resource, node 10, before terminating at destination node 4. This congested node 10 creates dependencies among these four flows. Our algorithm represents this dependency, using shared node 1 in the functional topology. These flows pass through other nodes in the physical topology. For example, flows 1 and 2 also share node 7 in the physical topology. but node 7 is an over-provisioned node and does not have any influence on dependency creation between flows 1 and 2. Thus, our algorithm

ignores node 7 and considers node 10 of the physical topology, represented as node 1 in the functional topology. Similarly, it ignores node 31, which is shared by flows 14 and 17, as it does not have any effect on creating dependency between these flows. In the second cluster, traffic flows 3, 4 and 5 share common resources 7 and 11 on their route to destination node 5. Node 11 of the physical topology creates dependencies among these flows. Therefore, this dependency is represented as shared node 2 among these flows in the functional topology. The third cluster contains the rest of the traffic flows 6 to 17. We highlight some nodes and colour their path, to make it easier to observe their paths and shared resources. From the physical topology, we can see the flows 7 and 10 share one congested node 18, and that is represented as node 4 in the functional topology. Shared resource node 18, which creates dependency between flows 8 and 11, is shown using node 5 in the functional topology. Traffic flows 6, 7 and 11 share resources 6 and 17 before terminating at their destination. Between these two resources node 17 is responsible for creating dependencies among these flows. This dependency is represented by node 3 in the functional topology. We can see flows 9, 12 and 15 have no dependency in the network. Therefore, their source nodes are directly connected with the destinations. Flows 13 and 16 share a congested buffer of node 30 in the physical topology. This node is represented as node 6 in the functional topology. We can see that our modified algorithm represents the nodes that create dependencies among traffic flows in the network – it does not represent dependencies among multiple flows using more resources present in the physical topology. From the analysis, we can confirm that it improves accuracy by representing dependencies among flows with the same number of resources that create dependencies among flows, ignores all of the nodes that do not have any influence on dependency creation and simplifies the visualisation of the functional topology. Our algorithm also successfully constructs functional topology of local area network (LAN), which is provided in appendix B.

6.7 Summary

In this chapter, we have discussed our algorithm for constructing a functional network topology. We applied our algorithm in three topologies and constructed respective functional topologies. Then, we compared these functional topologies with physical topologies. Initially, we found that functional topology creates more resources to represent traffic flow dependencies than it actually presents in the physical network. We addressed this issue and modified our algorithm to rectify this problem. With our modified algorithm, we constructed functional topologies and compared them with physical topologies. We

found that our algorithm constructed an accurate functional topology compared to its physical topology counterpart, and it did not take up more resources than the physical topology, although occasionally it would ignore some physical topology nodes that had no influence on dependency creation. It is now easier and simpler to visualise a network using a functional topology.

Chapter 7

Thesis Summary and Future Work

7.1 Summary

Knowledge of the topology of a network is vital for being able to provision the network resources efficiently. However usually this information is not available due to proprietary issues or security threats. We have devised a methodology in this thesis for discovering what we call "functional topology" of a network. We use the functional topology to deduce how resources are shared between different traffic flows. A functional topology describes the dependencies of the traffic flows as a graph of interactions, instead of describing the physical connection between the network elements.

Measuring interdependencies of traffic flows is the building block of functional topology. The accuracy of constructing the functional topology relies on dependency measurement results of traffic flows. Thus, we first focus on measuring interdependencies of traffic flows accurately and use these results to construct the functional topology. We choose the cross-correlation method to measure interdependencies of traffic flows in networks. We show that the method is able to produce promising results of measuring interdependencies of traffic flows and constructing functional topology of a network. In summary, major contributions of this thesis are as follows:

7.1.0.1 A novel algorithm to construct functional topology

All the results obtained from the dependency measurement of traffic flows are important for constructing the functional topology. We create an algorithm that produces functional topology of the network. To the best of our knowledge, this is the first algorithm that produces the the functional topology of any communication network. This algorithm takes input from a cross-correlation matrix where anti-correlation is considered as a result of sharing common resources and correlation is considered to occur due to transitive relationship among traffic flows. After creating a model of the functional topology, we apply binary logic to evaluate flow constraints obtained from the correlation matrix. This process works as validation for functional topology of the network. We get the final the functional topology if it satisfies all the flow constraints. Initially, we observe that functional topology creates more nodes than physical topology to represent dependency among traffic flows. It happens where multiple flows share one common resource among themselves. We modify our algorithm such that it can detect that multiple flows share one common resource among themselves and present that common resource as one single node shared by multiple flows in the functional topology. It improves the accuracy of our algorithm for creating functional topology. We find that many nodes that do not have any influence in creating dependencies on traffic flows are not shown in the functional topology. Thus, we can say that the functional topology simplifies the network topology by presenting only nodes that create dependencies among traffic flows in network.

7.1.0.2 Determine suitable sampling windows for measurement and use of different sampling windows to get complete set of interdependencies of traffic flows

In this thesis, we show that our method can measure a complete set of interdependencies of traffic flows in network using different sampling windows according to maximum delays of traffic flows. During our measurements, we find that flows with small delays exhibit strong correlations (peak correlation coefficients) in contrast to flows with large delays, which exhibit weak cross-correlation using small sampling windows. As we consider congested networks, we expect to get peak cross-correlation coefficients for all traffic flows. In order to find the peak correlation coefficients of flows with large delays, we use different sampling windows for sampling time series of packet counts. We find their peak correlations using large sampling windows. From analysis of results of these experiments, we find that we get the largest correlations between traffic flows if we use sampling windows according to the maximum delays

of traffic flows. As different flows in the network have different delays, we suggest to use sampling windows according to maximum delays of traffic flows to measure dependencies among traffic flows. Use of different sampling windows according to maximum delays of flows provide complete set of dependencies of traffic flows.

7.1.0.3 New technique that uses eigendeomposition and spanning tree to identify dominant flows in network

We develop a new technique to identify dominant flows in the network. Dominant flows are the most important flows in the network. They have the most traffic fluctuations and are responsible for creating congestion in the network. This technique uses eigendecomposition to find the dominant flows in the network and spanning tree algorithm is applied to rank the dominant flows according to their contributions. We validate our dependency measurements results using inverse participation ratio (IPR) that provides the number of dominant flows in network. During this process, we match the number of highly correlated traffic flows from cross-correlation matrix with IPR value. We find that IPR value is bigger than this number of highly correlated pairs, which indicates the presence of other flows that contribute to the IPR value. IPR does not provide information about "which" are the dominant flows in network. Thus, in order to identify dominant flows in the network, we develop our new technique. Using our technique, we are able to identify all the dominant flows in the network suggested by IPR value. We show that our technique identifies not only dominant flows that exhibit strong cross-correlations but also many dominant flows that exhibit weak cross-correlation in cross-correlation matrix.

7.1.0.4 Application of our method for developing adaptive spatial sampling technique in Wireless Sensor Network

Using our method, we are able to get complete set of interdependencies of traffic flows in communication network. To validate our method, we apply it on real data obtained from wireless sensor networks. We use the same method to develop an algorithm called the adaptive spatial sampling technique that identifies dominant sensor nodes within clusters and sample only those dominant sensor nodes for a certain period and set other nodes into sleep mode to saves energy for those sensors. Our method is used as part of the algorithm and becomes really useful while identifying dominant sensor nodes within clusters of the network. We show that we can achieve significant amount of data reduction that

saves energy for sensor nodes with without deteriorating accuracy in measurements using our algorithm. Moreover, this application our method as part adaptive spatial sampling technique provides validation for the same method apply to measure interdependencies of traffic flows in network. We show that our cross-correlation method can be used not only wired networks but also in wireless sensor network.

7.2 Limitations of our method

We show that our method can measure complete set of interdependencies of traffic flows in different networks. Using the results of interdependency measurements, we construct functional topology of network. However, our method to measure interdependencies has some limitations. It constructs a cross-correlation matrix. If there are n number of flows, then it constructs an $n \times n$ cross-correlation matrix. If n is a large number in a large network, it gets very expensive in terms of computation. After constructing cross-correlation matrix, we use cut-off cross-correlation coefficients as threshold to remove noise from the matrix. Then, it becomes easier for us to use cross-correlation matrix for further analysis.

We mentioned in earlier chapters that we need a certain level of congestion in our network to be able to calculate the cross-correlations among traffic flows accurately. To be precise, the utilizations of buffers of nodes have to be more than 70%. If the utilization of buffer is below 70%, then our method cannot accurately measure the dependency between traffic flows. So, we always consider congested networks for experiments in our thesis. Uncongested networks or over provisioned networks do not need any further provisioning. Further provisioning on network is only needed when the network reaches to a threshold of congestion. We know, when a network reaches a threshold level of congestion, then our method can measure dependency of traffic flows accurately and construct functional topology that helps to provision the network.

In our method, during eigen-analysis of cross-correlation matrix, we only consider largest eigenvalue and associate eigenvector to create a projection of cross-correlation matrix. One can argue that we might loose some information using only largest eigenvalue for our analysis. But we know that largest eigenvalue contains most of the information about system and it is very cost-effective in terms of computation. We also avoid eigenvalues that represents random interaction by considering only largest eigenvalue in our analysis.

We apply spanning tree algorithm in our technique to identify dominant flows. We know that a spanning tree considers only minimum weight of edges to join all vertices of the graph (here, each flow represents as vertex in spanning tree). As spanning tree only considers minimum weights of the edges to connect vertices, we might loose some information by using spanning tree. We know that flows with strong contributions are always shown by spanning tree. Sometimes, It happens that a flows with strong contribution is not shown in spanning tree (because of other strong candidates) for certain sampling windows. As we use different sampling windows, the flow is shown in spanning tree with other sampling windows if it has genuine strong contribution. Thus, we can argue that we do not loose any useful information using spanning tree.

In the previous chapter, we show that our algorithm constructs an accurate functional topology. We have some limitations of our algorithm. As the functional topology describes dependencies among traffic flows, it has many possibilities of representing traffic interactions as a graph by satisfying flows constraints. If it cannot satisfy the flow constraints, then we need to permute different lists (order list and resource list) for the correct resource order that can satisfy all flow constraints. This permutation can be expensive in terms of computation for large networks. But we observe that large topologies form different clusters in functional topology where a few important nodes are connected with the rest of nodes of the cluster. In those cases, our algorithm can accurately construct functional topology.

7.3 Future Work

In this section, we propose some future directions for functional topology of network. In all our experiments, we use Poisson and Pareto distributed traffic. Here, we only considers UDP as layer 4 protocol. In the future, we will also consider other traffic types and protocols such as TCP while generating traffic for experiments. We already measure interdependencies of traffic flows of LAN. In the future, we are interested to apply our method on real-world networks to measure interdependencies of traffic flows and construct functional topology of the network. We want to evaluate performance and usefulness of a functional topology to third part stakeholders (content providers, peer to peer networks) in future.

In all our experiments, we consider all the traffic flows of the network for our measurements. While applying our methods in real-world networks in future, there will be other traffic flows, which are not of our interest for the measurements. Our method will consider these traffic flows as noise. We want to

evaluate the performance of our method to measure interdependencies of traffic flows and construct a functional topology in the presence of noise in future.

During our experiments, we find that dependency of flows that have small delay and usually close to destinations can be captured using smaller sampling windows. We can measure dependencies of flows that have large delays and pass through multiple intermediate nodes along their path using large sampling windows. With this analysis, we can determine the position (comparative) of nodes that create dependency among flows. We can use this results and feed as input into our algorithm while constructing functional topology of network. We can compare the results of functional topology with physical topology as validation in the future.

We consider only congested networks and measure strong collective dependencies among traffic flows, so that functional topology can describe interdependencies of traffic flows in network. As it can describe how all flows are dependant among themselves, we think, it can also describe catastrophic failure in the future. It will be useful to identify most dependant or important nodes in network and alternate options as disaster recovery in the case of failure. Functional topology simplifies the network by ignoring nodes that do not have any influence on dependency creation. As it can avoid complexity of physical topologies and still can describe dependency among traffic flows, it can be useful for overlay networks.

7.4 Concluding remarks

This thesis is focused on constructing functional topologies of network. In order to construct a functional topology, first we measure interdependencies of traffic flows. We show that our cross-correlation method is able to get a complete set of interdependencies of traffic flows in network. We devise an algorithm that considers all the results obtained in dependency measurements and construct the functional topology of network. We show that our algorithm constructs an accurate functional topology of the network. We find that functional topology simplifies network topology by describing dependencies of traffic flows and presenting nodes that have influence on dependency creation.

Functional topology is constructed to provide knowledge about network topology, so that it can help to provision networks efficiently. It will help third party stakeholders that depend on others' network infrastructure and do not have access to information about network topology. As it can describe depen-

dencies among traffic flows in network, then, it will be able to describe ‘Cascade Effect’ and catastrophic failure in network. We observe that functional topologies simplifies visualization of networks by considering only important nodes that create dependencies among traffic flows. This observation leads us to believe that it can be useful for describing overlay networks.

Appendix A

Experiment Setup and Validation

A.1 Introduction

We use network simulator version 2 (NS-2) to create different network topologies and scenarios to run our experiments. It is a well known simulator that has been used for many years. While creating network topologies in ns2, we use traffic sources that generate Poisson and Pareto distributed traffic in the network. In our experiments, we use FIFO (First In First Out) queue in different nodes of the network. We use simplex and duplex links in our experiments. After creating network topologies, we run our experiment. Simulation run time is discussed later in this chapter. After finishing our experiments, we post process the data generated by simulator using awk scripts and MATLAB. Simulator creates a model, which can map a real system. It is very important task to make a model which generates results according to theory almost accurate to a real system. So, it is really important to check whether experiment data generated by simulator is correct or not. In this chapter, we discuss about validation of our simulator. First, we generate Poisson distributed traffic and validate the results. Then, we discuss about the bias of the simulator, queuing behaviour with different loads and more traffic analysis of Poisson distributed traffic. After that we discuss about simulator run time of each experiments.

A.2 Experiments and simulator

In our simulation experiments, we use Poisson Traffic source to generate traffic data. We analyse the data by MATLAB later. We need to check the performance of our simulator whether it provides us correct results and the results are close to practical scenarios. To generate Poisson source, add different nodes, queues and create a network environment, we need simulator. In our experiment we will create the network scenario in Network Simulator 2(NS2). In this validation section, we start with Poisson distributed traffic source. Here, we consider a source that generates the Poisson distributed traffic, then the traffic comes to a node which has a FIFO queue, traffic come out from the queue and go to the destination. We measure different parameters like inter arrival time, average queue length etc to validate the result.

.

A.3 Proposed design for the experiments

As mentioned earlier, we start with the Poisson distributed traffic source. The proposed design is shown in figure A.1. Here the source generates Poisson traffic that go to the node, which has a FIFO queue. As the traffic is memory less and service time is deterministic, we can call it a M/D/1 queue. The traffic come out of the queue and go to the destination. We take the measurement at queue.

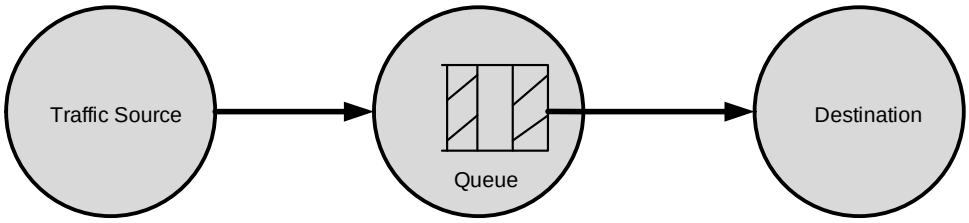


Figure A.1: Generating Poisson Traffic

In NS-2 many queues can be analysed. For our research purpose, M/D/1 would be analysed and validated with the results. From the figure A.1, we can see source sends packets to destination via another node that has FIFO(Droptail) queue is at the end of Node. It should be mentioned that the FIFO(Droptail) is built in function of NS-2. If the packet size is defined the Droptail queue would turn into M/D/1 queue.

An agent is attached with Source. Agent defines the types of traffic means whether it is TCP or UDP. The acknowledgement would send according to the type of traffic. In this case the Agent is UDP.

A.3.1 Traffic source

Traffic source generates packet according to the type of traffic defined by the code. Lots of traffic can be supported by NS-2. For example, it can support CBR, Exponential etc. In this research, the traffic type is Poisson.

A.3.1.1 Poisson distributed traffic in NS-2

Poisson distributed traffic has two basic characteristics. One is the packets will arrive in a particular interval with average of λ. Poisson distributed traffic source can be made using on-off source. On-Off traffic model is basically a two state Markov model. In the on period it generates packets at a fixed rate and during the off period it generates nothing. The on and off period is exponentially distributed. So the probability of on state or off state is given below [79].

$$P_{on} = \frac{T_{on}}{T_{on} + T_{off}} \tag{A.1}$$

$$P_{off} = \frac{T_{off}}{T_{on} + T_{off}} \tag{A.2}$$

We can calculate the probability of on and off by the equation given above. It is very useful to determine the nature of the on-off source.

Poisson distributed traffic source can be derived from on-off source. If T_{on} is set to zero then NS2 generate at least one packet during that time period. One packet generates during one cycle $(MeanT_{on} + MeanT_{off})$.

A.3.1.2 Calculation of the mean of Poisson distributed traffic source from on-off period

If the mean of period is 4ms then NS2 generates a packet on average of 4ms (0ms+4ms). The relation between the mean arrival rate of packets and mean inter-arrival time is given below.

$$\lambda = \frac{1}{MeanT_{on} + MeanT_{off}} \tag{A.3}$$

$$\lambda = \frac{1}{0ms + 4ms}$$

$$\lambda = 250pps$$

A.3.2 Validation of Poisson distributed traffic Source in NS-2 for M/D/1:

Poisson distributed traffic with different three mean have been applied to the model to observe the results of M/D/1. The link between node and destination is 0.04 Mbps (5pps). The buffer size is 1000 packets. The other parameters are given below.

Mean on time	0 ms
Mean off time	250 ms
Simulation time	172800 sec or 20 Days
Packet Size	1000 bytes

Table A.1: Parameters of Poisson distributed traffic source with 250ms idle time

Table A.1 is showing the parameters value for a Poisson distributed traffic source with 250ms idle time. The mean number of packets we generate by the mean idle time is 4pps.

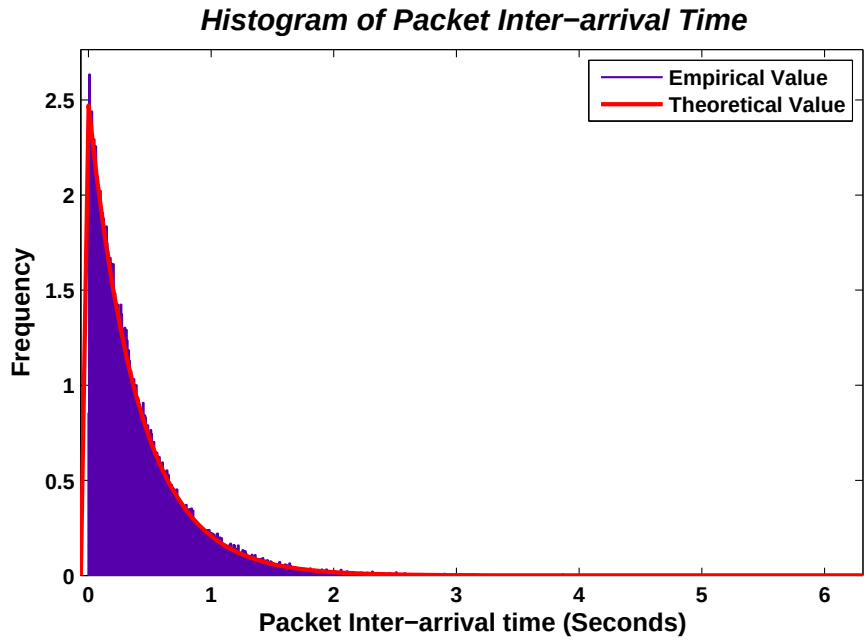


Figure A.2: Comparison of theoretical and experimental distribution of packet inter-arrival time

Fig.A.2 is showing the inter arrival time of the packets with 250ms idle time. As the mean on time is set to zero to generate the Poisson distributed traffic source in NS2, the average mean packet arrival time is more like the mean off period.

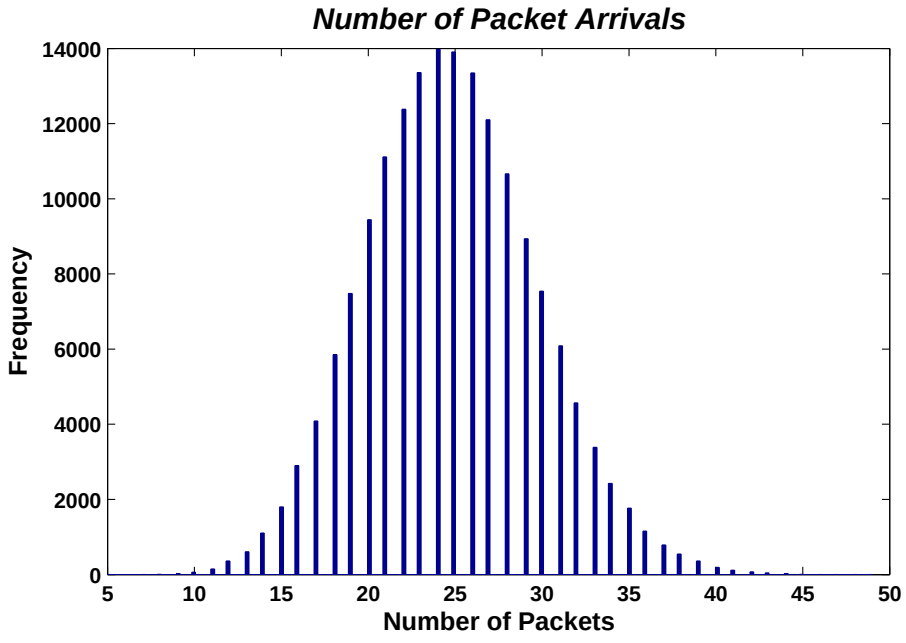


Figure A.3: Histogram of number of packet arrival. Mean:24pps, Sampling Window: 6 seconds

From the fig.A.2, it can be seen that the distribution is ranging from 0.004 sec to more than 6 seconds. The theoretical fit almost match with experimental data except a slight gap at the beginning. This happen because of the bias of the simulator. NS2 cannot produce two packets at a time, which is the bias of simulator. For the bias of NS2, it cannot generate the packets less than 4ms. So we are basically restrict the distribution ranging from 0.004 sec to more than 6 seconds. As MATLAB (we get the theoretical distribution using MATLAB) generates the numbers and it is ranging from zero to around 6.2 seconds. For the bias the frequency of NS2 distribution is around 2.7. On the other hand the distribution from MATLAB generated data is near to 2.5. The experimental result deviates from theoretical results just because of the bias in induced by the simulator.

A.3.2.1 Bias of the experiment

NS2 cannot generate two packets at a time. It means two packets cannot be overlapped by time. So this little non overlapping time of this simulator depends on the time to generate one packet, the size of the packets and the bandwidth of the link. If the packet size is bigger it will take more time to generate the packet. If the bandwidth is higher it will have faster communication. For example, if the packet size is 1000 bytes (8000 bits) and the bandwidth is 2Mbps, time taken to generate one packet is[80],

$$Time = \frac{Packetsize}{Bandwidth} \tag{A.4}$$

$$Time = \frac{8000bits}{2mbps}$$

$$Time = 0.004sec$$

In the above graphs, it can be seen that there is no packet with the inter-arrival time less than 0.004 sec. It proves that there is no overlapping of packets during the experiment. So 0.004 sec is the bias of this experiment.

A.4 Queue length comparison

In this experiment, simple M/D/1 queue has been used. The traffic pattern is Markovian or memory less, the service is determined by the packet size (1000bytes) and the queue has only one server. The formula for number of packets in the buffer is given below[81].

$$Numberof PacketsinBuffer = \frac{\rho^2}{2(1-\rho)}$$

where ρ is the utilization ratio or load

The function qmon is a built-in function of NS2[82]. It provides the information of number of packets in the queue. By using the function the numbers of packets were determined in this experiment. The comparison graph is given below.

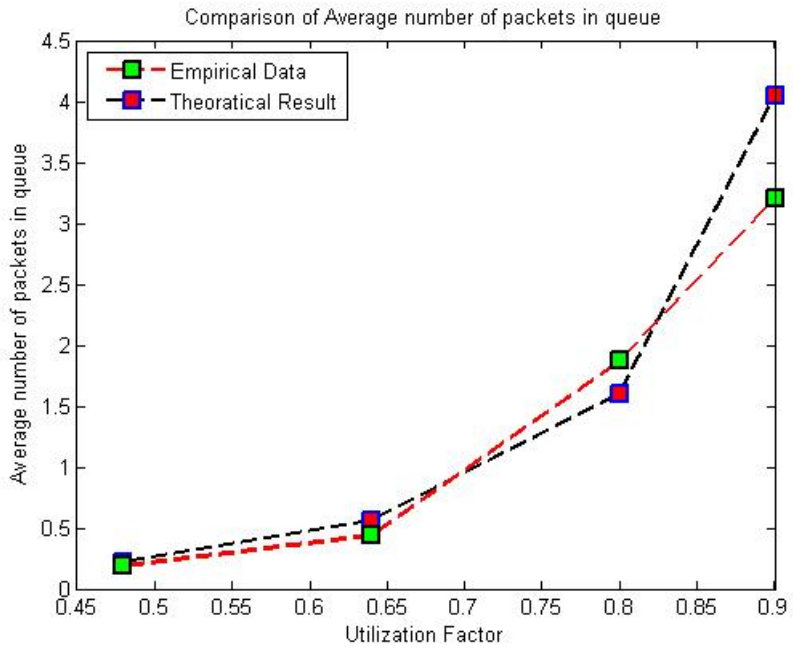


Figure A.4: Comparison of Theoretical data and empirical data of M/D/1 queue length

The fig.A.4 is showing the comparison of theoretical results and empirical results of queue length.The accuracy of theoretical formula is not good enough as the queue behave differently than low utilization ratio. The behaviour pattern would change after utilization ratio 0.8. It can be seen clearly from the

graph that the pattern has been changed after utilization ratio 0.8. But the deviation of change is not too large, so it can be said that the results are quite acceptable.

A.5 Autocorrelation function of traffic

Autocorrelation is basically a dot product with the data itself. There are two purposes to use autocorrelation. They are-

- To check the randomness in data
- To identify an appropriate time series model if the data are not random

The autocorrelation function represent whether the traffic is memory less or not. In this case we will look into some results to check whether the traffic has the memory or not. First we will look into the traffic that generate and arrive in the queue. We use Poisson distributed traffic source. Poisson distributed traffic is memory less traffic. So we expect, the autocorrelation coefficients of Poisson distributed traffic would be close to zero[83]. In this case, we use data of 20 days.

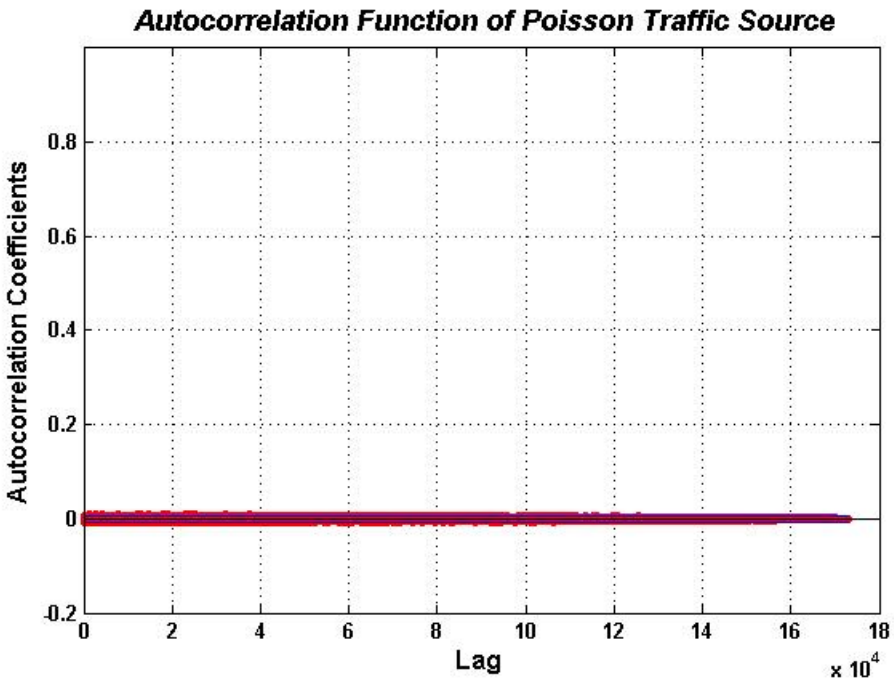


Figure A.5: Autocorrelation function of Poisson distributed traffic source with lags of 20 days

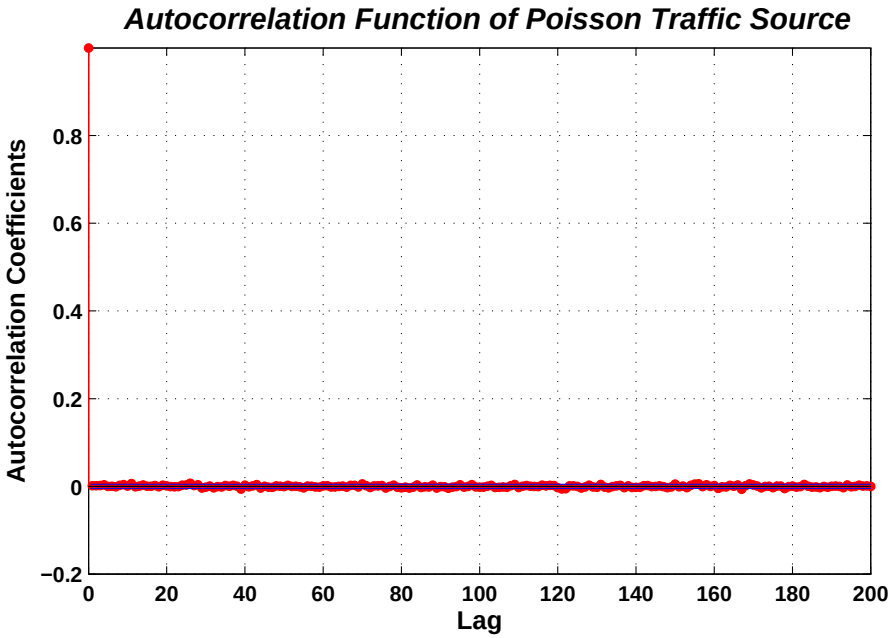


Figure A.6: Autocorrelation function of Poisson distributed traffic source with lags of 200 seconds

To get a clear view, we take 200 seconds lags of autocorrelation function in fig.A.6. In both of the figure (fig.A.5 and fig.A.6), autocorrelation coefficients are almost zero and within 95% interval.

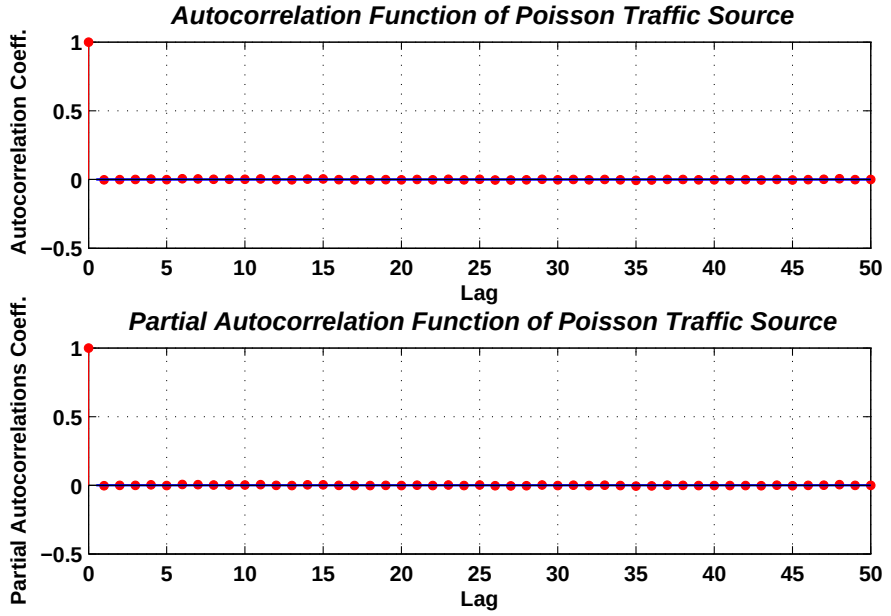


Figure A.7: Comparison of Autocorrelation and Partial autocorrelation function of Poisson distributed traffic source with lags of 50 seconds

Here, we also use partial autocorrelation function and compare with autocorrelation function for validation of memory less traffic.From fig.A.7, we can see the autocorrelation coefficients of 20 days data of packets generation with lags of 50 seconds are close to zero. From the analysis of autocorrelation function of simulator generated data, we can say that the traffic is memory less.

A.6 Ljung-Box Q test

The autocorrelation and partial autocorrelation function are useful tools to measure the presence of autocorrelation. Ljung-Box Q test is more quantative way to measure the presence of autocorrelation in data at multiple lags[84]. This test exhibits a null hypothesis where a series shows no autocorrelation for a fixed number of lags, L, against the alternative data that has some autocorrelation coefficient $\rho(k), k = 1, ..., L$ is non zero.

The test statistics is given below.

$$Q = T(T+2)\sum_{k=1}^{L}\left(\frac{\rho(k)^2}{T-k}\right) \tag{A.5}$$

Where T is the sample size, L is the number of autocorrelation lags and $\rho(k)$ is the sample autocorrelation lags at lag k[85].

We can use several lags to detect autocorrelation by Ljung-Box Q test. At first, we start with three lags of 5, 10 and 15 seconds. You use the packet generation data for 20 days.

Lags(seconds)	5	10	15
h-Test rejection decision	0	0	0
p-Test statistics p-value	0.6062	0.1492	0.0850
Qstat	3.6143	14.5536	22.9588
Critical Value	11.0705	18.3070	24.9958

Table A.2: Results of Ljung-Box Q test for 3 lags

From the result, we are interested in parameter h, which is responsible to reject or fail to reject the null hypothesis. From the standard test,

- h = 1, indicates rejection of the no autocorrelation null hypothesis in favour of the alternative
- h = 0, indicates failure to reject no autocorrelation null hypothesis.

From tableA.2, we can see the h value is zero for all three lags. It means the test fails to reject the no autocorrelation null hypothesis. It means autocorrelation of the data provided for the test is zero. We also take different lags for Ljung-Box Q test.

Lags(seconds)	5	10	15	20	30	50	75	100
h-Test rejection decision	0	0	0	0	0	0	0	0
p-Test statistics p-value	0.5957	0.7057	0.3962	0.5990	0.2389	0.3104	0.6356	0.5334
Qstat	3.6842	7.2073	15.7885	17.8234	35.1021	54.4085	70.1893	98.1585
Critical Value	11.0705	18.3070	24.9958	31.4104	43.7730	67.5048	96.2167	124.3421

Table A.3: Results of Ljung-Box Q test for 8 lags

From the result shown in tableA.3, we can see all the h values are zero. It indicates that the test result is failed to reject the null hypothesis and there is no autocorrelation in the data. From this analysis, we can say that the data provided for the test is memory less which is a property of Poisson distributed traffic.

A.7 Memory test of negative exponential traffic

Poisson distributed traffic has negative exponentially distributed packet inter-arrival time. We have already shown that our experimental data has negative exponentially distributed packet inter-arrival time(fig.A.2). We need to check whether the present state of packet generation depends on the previous state or not. It basically represents memory less property of data. To test this property, we place the initial position of packet arrival time against next packet arrival time in the graph. If one state depends on next state of time in packet generation, then we can see a regular pattern in the graph. If not, we can say one state does not depend on other states while packet generation, which indicates memory less process.

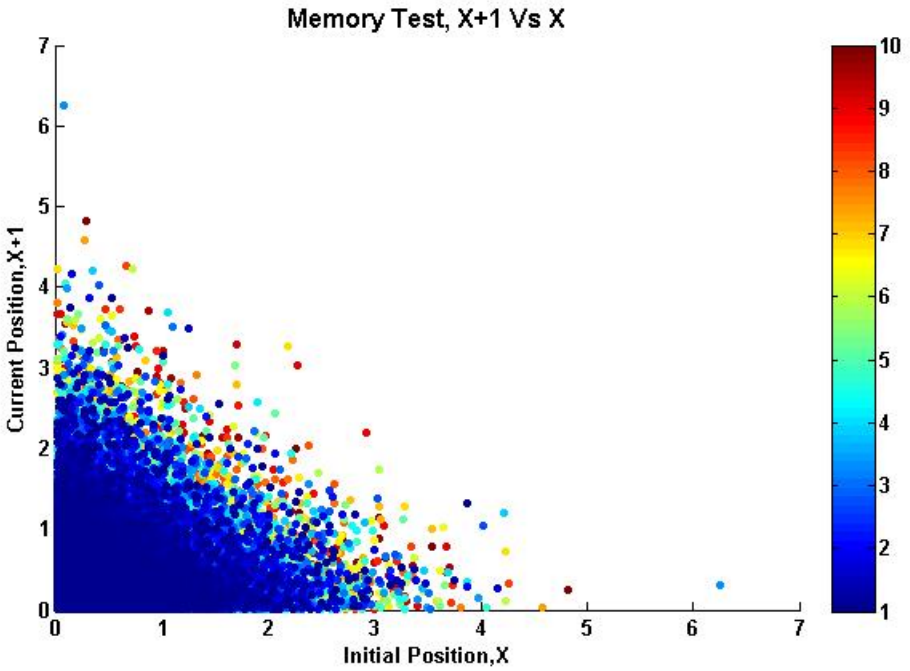


Figure A.8: Memory test of negative exponentially distributed packet inter-arrival time with lag 1

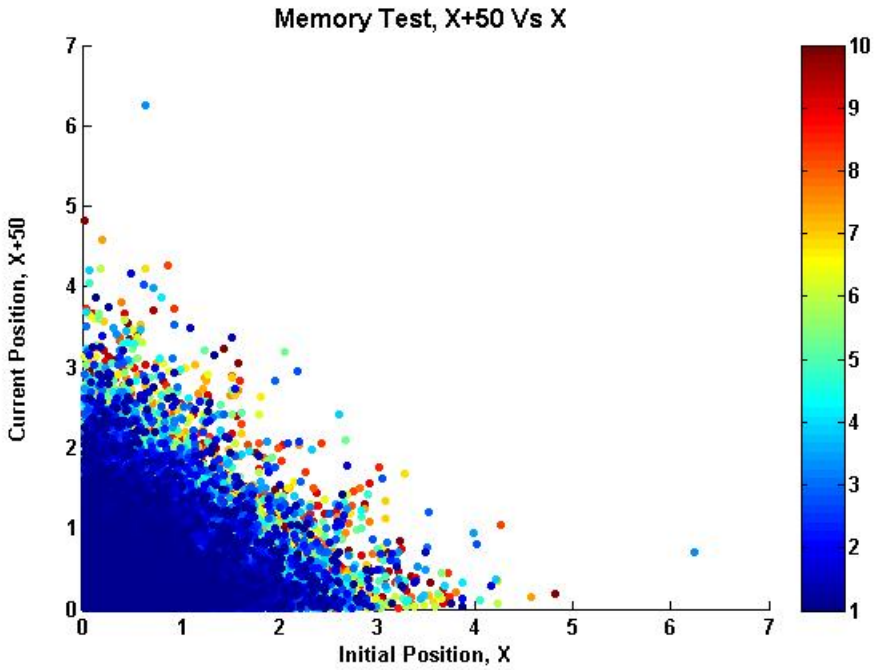


Figure A.9: Memory test of negative exponentially distributed packet inter-arrival time with lag 50

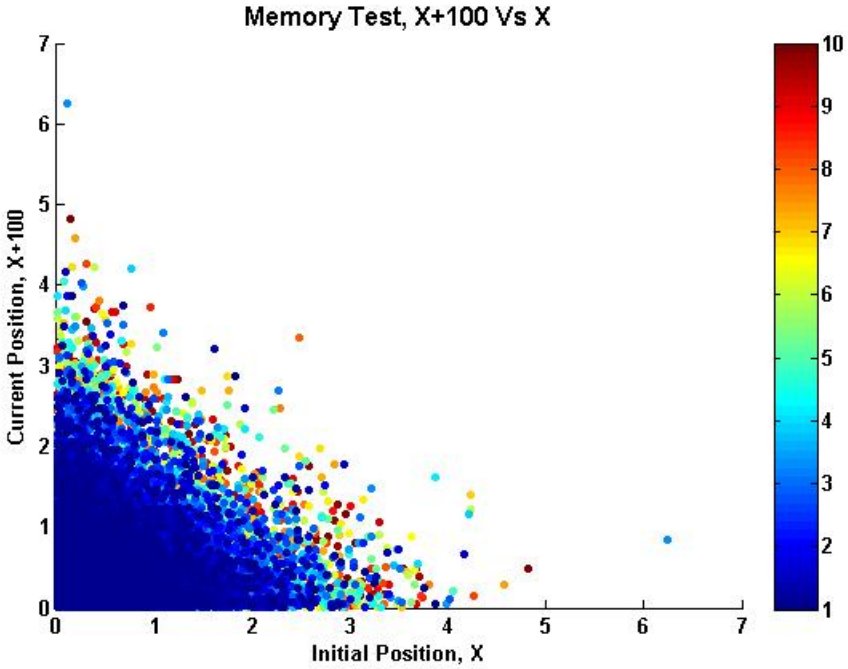


Figure A.10: Memory test of negative exponentially distributed packet inter-arrival time with lag 100

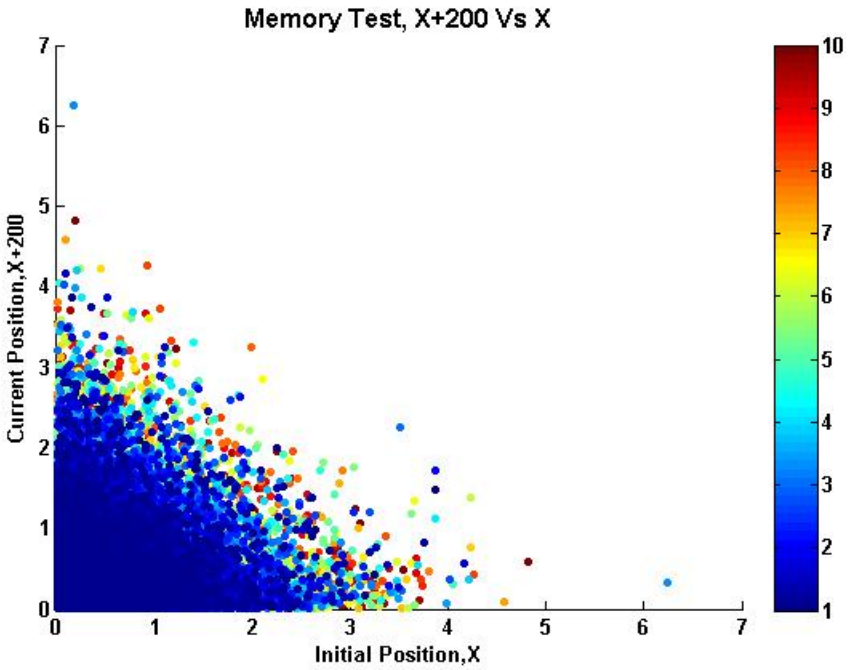


Figure A.11: Memory test of negative exponentially distributed packet inter-arrival time with lag 200

From fig.A.8 we can see that the position of packet inter-arrival time of packets do not form a regular

pattern. With increasing lag in fig.A.9,A.10 and A.11, the pattern of the position of inter-arrival packets remain irregular. We can see the positions of inter-arrival time disperse with bigger lags but still around the mean of the negative distribution of inter-arrival time of packets. This analysis proves that packet inter-arrival time is not dependant on other states. This memoryless property indicates that our generated traffic is Poisson distributed traffic.

A.8 Simulation time

Simulation time is very important. Some of the experiments have been reported earlier [5] [6][7] are on real data. To make same kind of scenario in the simulation, it is very necessary to determine the simulation time. The accuracy of the result will depend on the simulation time.

A.9 Convergence check

For checking the convergence of the experiment, we actually take 5001 samples. We need to check the mean value of the results. It is set it to 6000 packets per seconds. We also check the standard deviation. We observed that there was no huge fluctuations for the experimental results. The minimum deviation of 2 or 3 packets from each the experiment. The accuracy was high in that case.

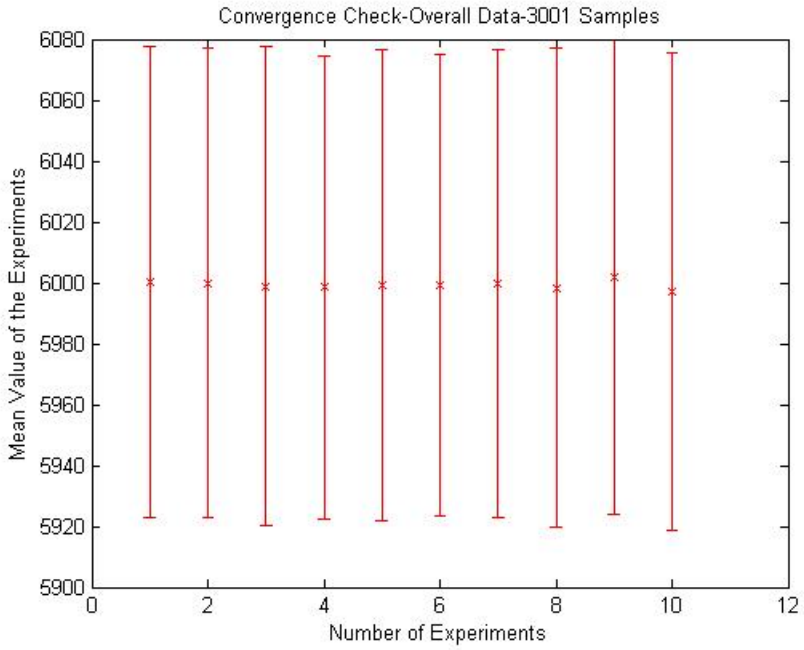


Figure A.12: Convergence check: Convergence check: Overall: for 3001 samples

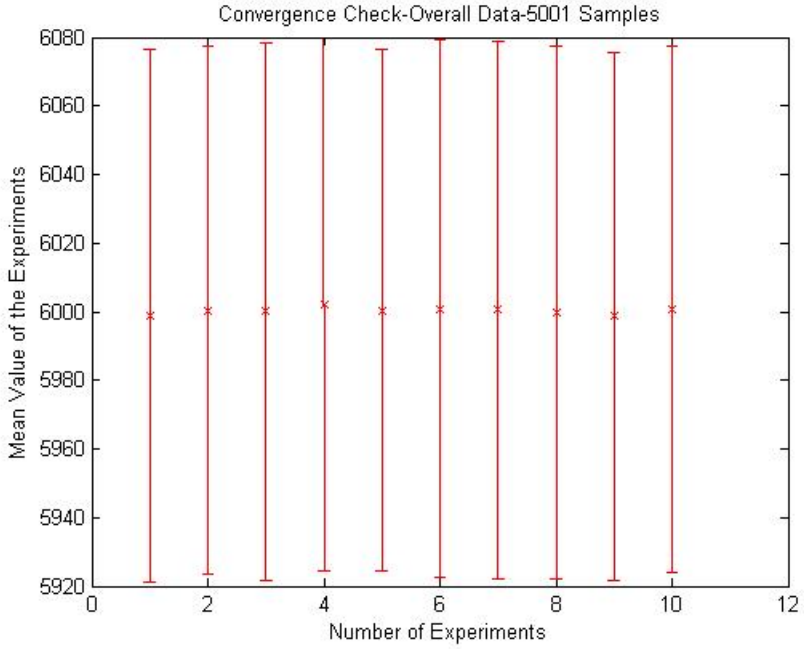


Figure A.13: Convergence check: Convergence check: Overall: for 5001 samples

Fig.A.12 and A.13 show the convergence check for 3001 and 5001 samples accordingly. The mean is almost at 6000 and fluctuations are very less for 5001 samples.

In the overall analysis, 5001 samples experiment is getting converged well. The mean value is almost 6000, more or less about 2 or 3 packets. From this analysis, we set the simulation time of the experiment is around 1800000 seconds or 20 days.

Appendix B

Cross correlation matrices and Functional Topology of LAN

In this chapter, we provide anti-correlation matrices **A** and correlation matrices **B** obtained from cross-correlation matrices for working example 2 and large topology (that contains 17 traffic flows) in chapter 6. We also provide functional topology of LAN at the end of this chapter.

B.1 Correlation matrices for working example 2

Here, we provide correlation and anti-correlation matrices obtained from cross-correlation matrix for working example 2 of chapter 6.

We set the diagonal value to zero. So, Anti-correlation matrix,

$$\mathbf{A} = \begin{vmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{vmatrix}.$$

Correlation matrix (after setting diagonal as zero),

$$\mathbf{B} = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}.$$

B.2 Correlation matrices for large topology

Here, we provide correlation and anti-correlation matrices obtained from cross-correlation matrix for large topology (that contains 17 traffic flows) of chapter 6.

We set the diagonal value to zero. So, Anti-correlation matrix,

$$\mathbf{A} = \begin{vmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{vmatrix}.$$

So, Correlation matrix,

$$\mathbf{B} = \begin{vmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}.$$

B.3 Functional topology of LAN

In this section, we provide functional topology of LAN. First, we provide physical topology of LAN, then functional topology followed by modified version of functional topology of LAN. We apply our extended algorithm discussed in chapter 6 to construct functional topology of LAN. Here, figure B.2 shows functional topology without applying extended version of our algorithm, while figure B.3 shows functional topology after applying extended version of the algorithm. In figure B.3, we highlighted important intermediate nodes of the network.

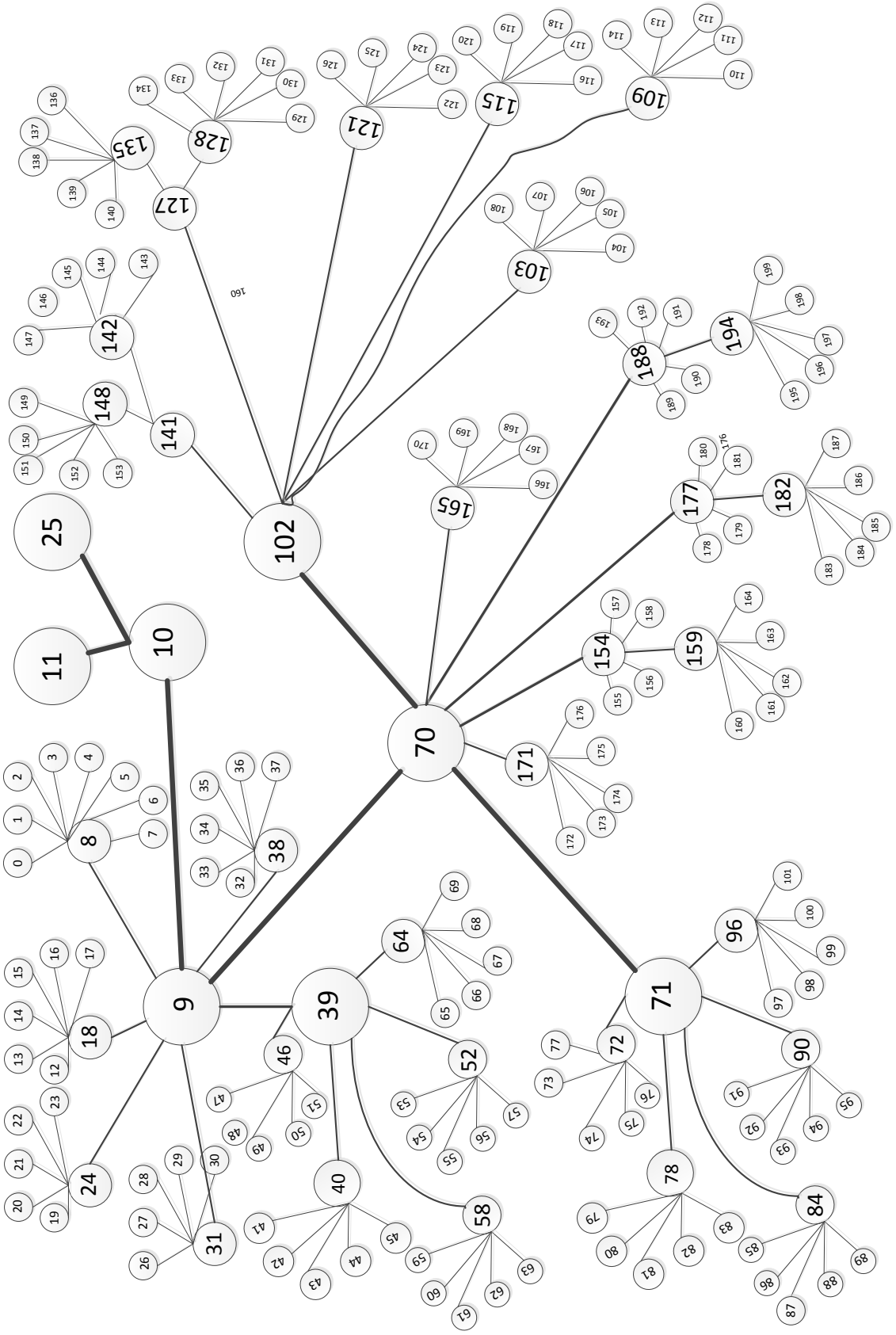


Figure B.1: Physical Topology of local area network (LAN)

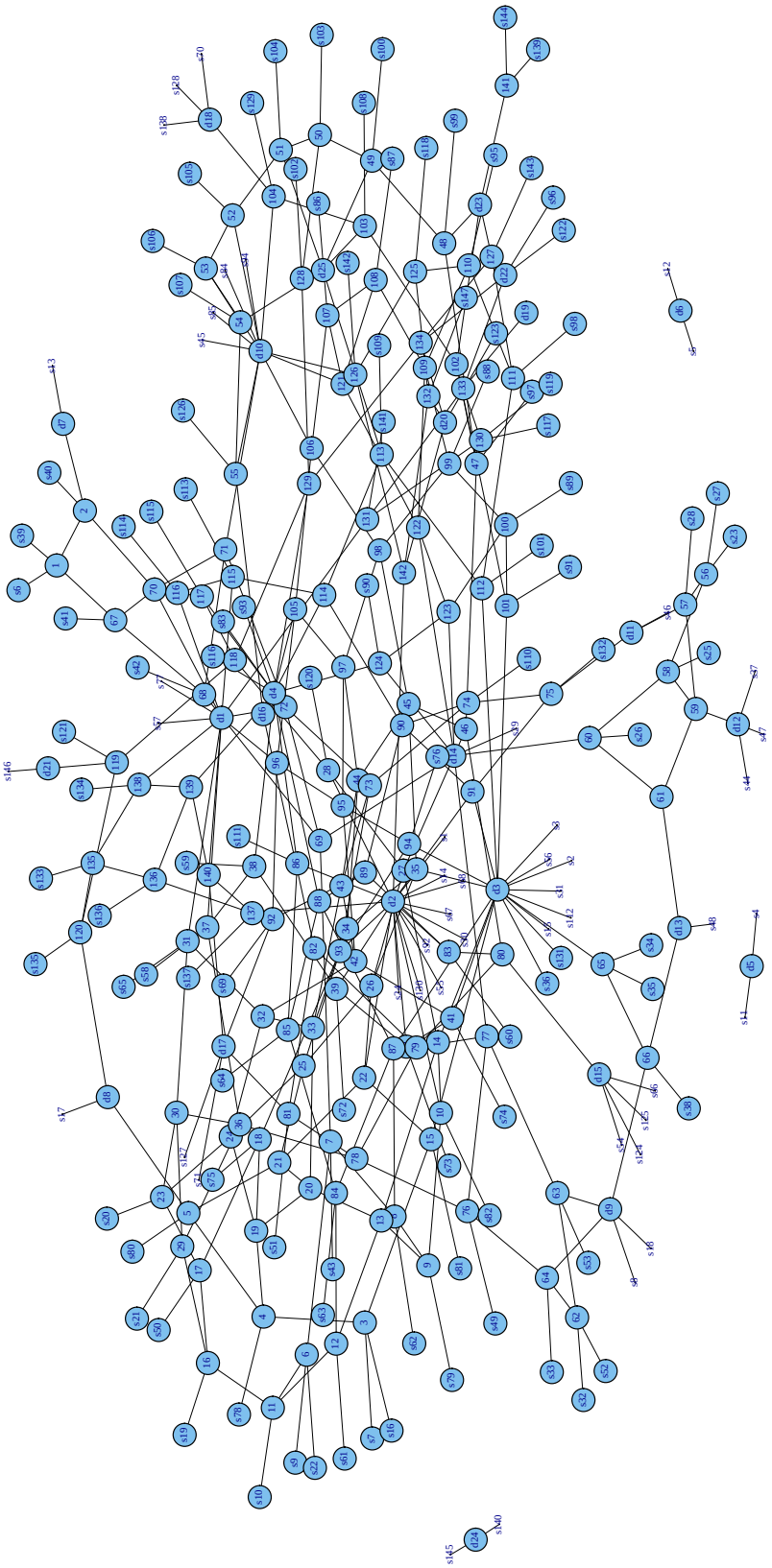


Figure B.2: Functional topology of LAN without applying extended version of our algorithm

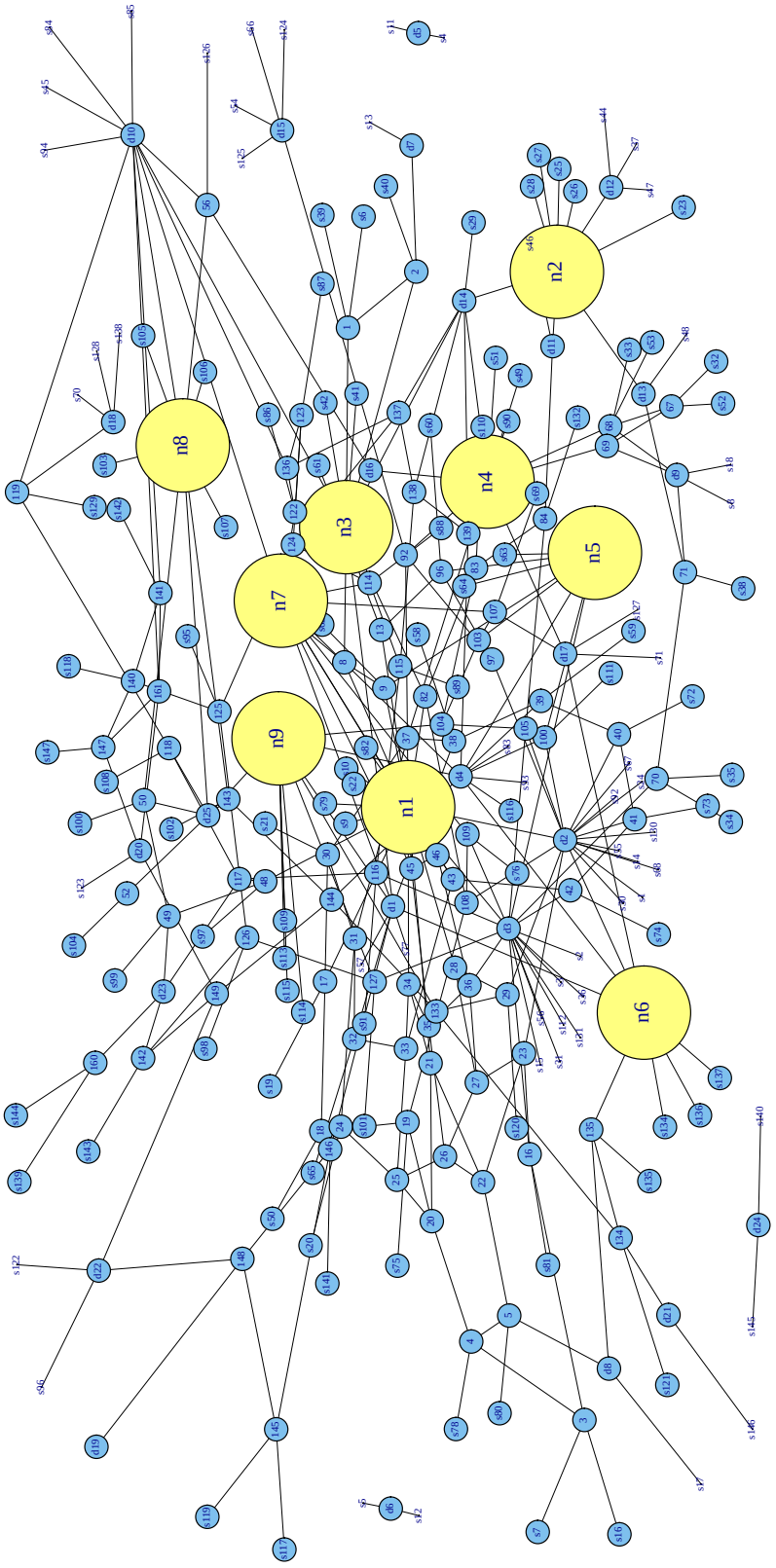


Figure B.3: Functional topology of LAN after applying extended version of our algorithm

Appendix C

Autocorrelation Function of Traffic Growth

In all our experiments, we collect the data at the destination of each traffic flow. We count the packets from each route to the destination f_{ij}(connection from i to j) with equal time intervals(sampling windows). Afterwards, we measure fluctuation in the traffic termed as growth rate g_{ij} which is the logarithmic ratio of two successive counts of the time series[5, 6, 7, 9]. The growth is expressed as

$$g_{ij} = \ln[\frac{f_{ij}(t+\tau)}{f_{ij}(t)}] \tag{C.1}$$

We use traffic growth to calculate cross-correlation coefficients between traffic flows in network. It is really important to check autocorrelation of traffic growth. If the traffic growth is autocorrelated then, it incorrectly calculate cross-correlations among traffic flows. Here, we are providing mathematical proof that autocorrelation coefficients of lag 1 of traffic growth should be around -0.5 (and coefficients for rest of the lags should be close to 0) to be qualified as non-autocorrelated data for calculating cross-correlations. If traffic growth is autocorrelated (meaning lag 1 of traffic growth is greater than 0), then we need to use other tools (such as ARIMA) to remove autocorrelation from data and then use it for calculating cross-correlations.

C.1 Mathematical Proof

The autocorrelation is

$$acf_m = \frac{E[(g_m(t)-\mu_t)(g_m(s)-\mu_s)]}{\sigma(g_m(t))\sigma(g_m(s))} = \frac{E[(g_m(t)g_m(s))]-\mu_m^2)}{\sigma_m^2} \tag{C.2}$$

where L is the lag, $\mu_m = E[g_m(t)] = E[g_m(s)]$ is the mean and $\sigma_m^2 = \sigma^2(g_m(t)) = \sigma^2(g_m(s))$ is the variance. Taking a lag of one, i.e. $s = t + 2\Delta t$

$$E[(g_m(t)g_m(s))] = E\Big[\log\Big(\frac{f_m(t+\Delta t)}{f_m(t)}\Big)\log\Big(\frac{f_m(t+2\Delta t)}{f_m(t+\Delta t)}\Big)\Big] \tag{C.3}$$

Using $\log(f_m(t+\Delta t)/(f_m(t)) = \log(f_m(t+\Delta t)\log(f_m(t))$ and expanding

$$E[(g_m(t)g_m(s))] = -A + 2B - E[\log(f_m(t))\log(f_m(t+2\Delta t))] \tag{C.4}$$

$$= -A + B - (E[\log(f_m(t))\log(f_m(t+2\Delta t))] - B) \tag{C.5}$$

where $A = E[\log(f_m(t+\Delta t)^2]$ and $B = E[\log(fm(t+\Delta t))\log(f_m(t)] = E[\log(f_m(t+\Delta t))\log(f_m(t+2\Delta t)]$.

The variance is

$$\sigma_m^2 = E[g_m^2(t)] = \Big[\log\Big(\frac{f_m(t+\Delta t)}{f_m(t)}\Big)^2\Big] \tag{C.6}$$

$$= E[\log(f_m(t+\Delta t)^2 + \log(f_m(t)^2 2log(f_m(t+\Delta t))\log(f_m(t)] = 2A - 2B \tag{C.7}$$

where $A = E[\log(f_m(t+\Delta t)^2] = E[\log(f_m(t)^2]$ and $B = E[\log(f_m(t+\Delta t))\log(f_m(t)]$. Using the last two equations the autocorrelation is

$$acf_m = -\frac{1}{2} - \frac{E[\log(f_m(t))\log(f_m(t+2\Delta t))] - E[\log(f_m(t))\log(f_m(t+\Delta t))]}{2A - 2B} \tag{C.8}$$

It shows that autocorrelation coefficients of the first lag should be around -0.5.

Figure C.1 shows the autocorrelation of traffic growth of a traffic flow in our experiment.

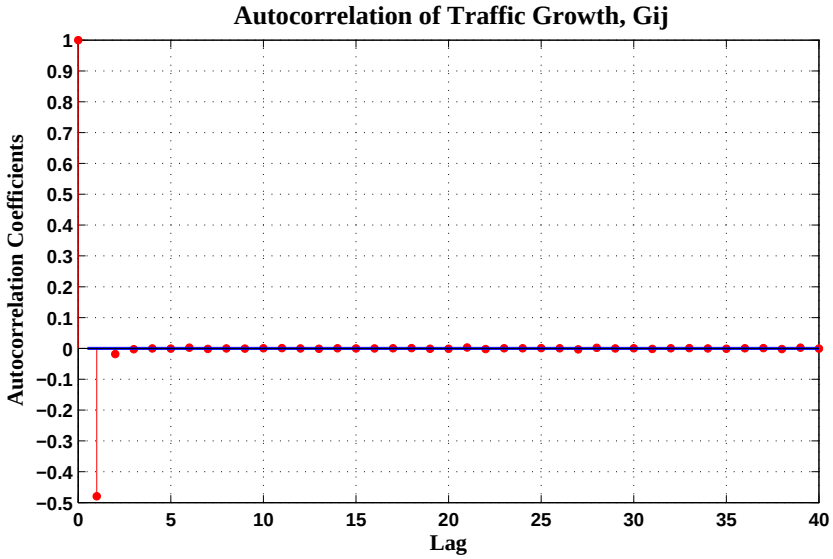


Figure C.1: Autocorrelation of traffic growth, g_{ij}

References

[1] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber, "Pushing cdn-isp collaboration to the limit," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 34–44, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2500098.2500103

[2] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: Provider portal for applications," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 351–362, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402999

[3] R. Landa, E. Mykoniati, R. Clegg, D. Griffin, and M. Rio, "Modelling the tradeoffs in overlay-isp cooperation," in *NETWORKING 2012*, ser. Lecture Notes in Computer Science, R. Bestak, L. Kencl, L. Li, J. Widmer, and H. Yin, Eds. Springer Berlin Heidelberg, 2012, vol. 7290, pp. 223–237. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30054-7_18

[4] R. Landa, E. Mykoniati, D. Griffin, M. Rio, N. Schwan, and I. Rimac, "Overlay consolidation of isp-provided preferences," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, July 2012, pp. 506–511.

[5] M. Barthélemy, B. Gondran, and E. Guichard, "Large scale cross-correlations in internet traffic," *Phys. Rev. E*, vol. 66, p. 056110, Nov 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.66.056110

[6] V. Rojkova and M. M. Kantardzic, "Analysis of inter-domain traffic correlations: Random matrix theory approach," *CoRR*, vol. abs/0706.2520, 2007. [Online]. Available: http://arxiv.org/abs/0706.2520

[7] V. Rojkova, Y. Khalil, A. Elmaghraby, and M. Kantardzic, "Use of simulation and random matrix theory to identify the state of network traffic," in *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, dec. 2007, pp. 647 –652.

[8] V. Rojkova and M. M. Kantardzic, "Delayed correlations in inter-domain network traffic," *CoRR*, vol. abs/0707.1083, 2007. [Online]. Available: http://arxiv.org/abs/0707.1083

[9] V. Rojkova and M. Kantardzic, "Feature extraction using random matrix theory approach," in *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*, Dec 2007, pp. 410–416.

[10] V. B. Rojkova, "Features extraction using random matrix theory," Ph.D. dissertation, Department of Computer Engineering and Computer Science, University of Louisville, 2010.

[11] V. Rojkova and M. M. Kantardzic, "Indicators of network-wide traffic dynamics: Random matrix theory approach," in *Proceedings of the ISCA 20th International Conference on Computer Applications in Industry and Engineering, CAINE 2007*, 2007, pp. 97–102.

[12] G. Bonanno, F. Lillo, and R. N. Mantegna, "High-frequency cross-correlation in a set of stocks," vol. 1, pp. 96–104, October 2001.

[13] M. Tumminello, T. Di Matteo, T. Aste, and R. N. Mantegna, "Correlation based networks of equity returns sampled at different time horizons," *The European Physical Journal B*, vol. 55, no. 2, pp. 209–217, 2007. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2006-00414-4

[14] T. Aste, W. Shaw, and T. D. Matteo, "Correlation structure and dynamics in volatile markets," *New Journal of Physics*, vol. 12, no. 8, p. 085009, 2010. [Online]. Available: http://stacks.iop.org/1367-2630/12/i=8/a=085009

[15] T. Di Matteo, F. Pozzi, and T. Aste, "The use of dynamical networks to detect the hierarchical organization of financial market sectors," *The European Physical Journal B*, vol. 73, no. 1, pp. 3–11, 2010. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2009-00286-0

[16] R. Mantegna, "Hierarchical structure in financial markets," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 11, no. 1, pp. 193–197, 1999. [Online]. Available: http://dx.doi.org/10.1007/s100510050929

[17] T. W. Epps, "Comovements in stock prices in the very short run," *Journal of the American Statistical Association*, vol. 74, no. 366, pp. pp. 291–298, 1979. [Online]. Available: http://www.jstor.org/stable/2286325

[18] J. Yuan and K. Mills, "A cross-correlation-based method for spatial-temporal traffic analysis," *Perform. Eval.*, vol. 61, no. 2-3, pp. 163–180, Jul. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2004.11.003

[19] K. B. K. Mayya and R. E. Amritkar, "Analysis of delay correlation matrices," *eprint arXiv:cond-mat/0601279*, Jan. 2006. [Online]. Available: http://arxiv.org/abs/cond-mat/0601279v1

[20] K. Mayya and M. Santhanam, "Correlations, delays and financial time series," in *Econophysics of Markets and Business Networks*, ser. New Economic Windows, A. Chatterjee and B. Chakrabarti, Eds. Springer Milan, 2007, pp. 69–75. [Online]. Available: http://dx.doi.org/$10.1007/978-88-470-0665-2_5$

[21] J. Kwapień, S. Drożdż, and A. A. Ioannides, "Temporal correlations versus noise in the correlation matrix formalism: An example of the brain auditory response," *Phys. Rev. E*, vol. 62, pp. 5557–5564, Oct 2000. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.62.5557

[22] W.-X. Du, C. H. Thurber, and D. Eberhart-Phillips, "Earthquake Relocation Using Cross-Correlation Time Delay Estimates Verified with the Bispectrum Method," *BULLETIN OF THE SEISMOLOGICAL SOCIETY OF AMERICA*, vol. 94, no. 3, pp. 856–866, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1785/0120030084

[23] S. Thurner and C. Biely, "Random matrix ensembles of time-lagged correlation matrices : derivation of eigenvalue spectra and analysis of financial time-series," *Quantitative Finance*, vol. 8, no. 7, pp. 705–722, 2008. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:0168-ssoar-221153

[24] J. Liu, W. Zhang, J. Yuan, D. Jin, and L. Zeng, "Monitoring the spatial-temporal effect of internet traffic based on random matrix theory," in *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, 2008, pp. 258–265.

[25] J. Liu, D. Jin, J. Yuan, W. Zhang, L. Su, and L. Zeng, "A rmt and pca-based method of monitoring the large-scale traffic pattern," in *Modelling Simulation, 2009. AMS '09. Third Asia International Conference on*, 2009, pp. 698–703.

[26] J. Liu, P. Gao, J. Yuan, and X. Du, "An effective method of monitoring the large-scale traffic

pattern based on rmt and pca." *Journal of Probability and Statistics*, vol. 2010, pp. Article ID 375 942, 16 p.–Article ID 375 942, 16 p., 2010. [Online]. Available: http://eudml.org/doc/228052

[27] E. P. Wigner, "Characteristic Vectors of Bordered Matrices With Infinite Dimensions," *The Annals of Mathematics*, vol. 62, no. 3, pp. 548–564, 1955. [Online]. Available: http://www.jstor.org/stable/1970079

[28] M. L. Mehta, *Random Matrices*, 3rd ed. San Diego,CA: Elsevier, November 2004, vol. 142, ch. Introduction, pp. 1–5.

[29] F. J. Dyson, "Statistical theory of the energy levels of complex systems. i," *Journal of Mathematical Physics*, vol. 3, no. 1, pp. 140–156, 1962. [Online]. Available: http://link.aip.org/link/?JMP/3/140/1

[30] M. L. Mehta and F. J. Dyson, "Statistical theory of the energy levels of complex systems. v," *Journal of Mathematical Physics*, vol. 4, no. 5, pp. 713–719, 1963. [Online]. Available: http://link.aip.org/link/?JMP/4/713/1

[31] T. A. Brody, J. Flores, J. B. French, P. A. Mello, A. Pandey, and S. S. M. Wong, "Random-matrix physics: spectrum and strength fluctuations," *Rev. Mod. Phys.*, vol. 53, pp. 385–479, Jul 1981. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.53.385

[32] T. Guhr, A. MllerGroeling, and H. A. Weidenmller, "Random-matrix theories in quantum physics: common concepts," *Physics Reports*, vol. 299, no. 46, pp. 189 – 425, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0370157397000884

[33] J. L. Toole, N. Eagle, and J. B. Plotkin, "Spatiotemporal correlations in criminal offense records," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 4, pp. 38:1–38:18, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/1989734.1989742

[34] M. Zhu, Q. Wu, Y. Yang, and J. Zhou, "A new approach to identify functional modules using random matrix theory," in *Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on*, 2006, pp. 1–7.

[35] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. N. Amaral, T. Guhr, and H. E. Stanley, "Random matrix approach to cross correlations in financial data," *Phys. Rev. E*, vol. 65, p. 066126, Jun 2002. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.65.066126

[36] V. Plerou, P. Gopikrishnan, B. Rosenow, L. Amaral, and H. Stanley, "A random matrix theory approach to financial cross-correlations," *Physica A: Statistical Mechanics and its Applications*, vol. 287, no. 34, pp. 374 – 382, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437100003769

[37] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. Amaral, and H. Stanley, "Econophysics: financial time series from a statistical physics point of view," *Physica A: Statistical Mechanics and its Applications*, vol. 279, no. 14, pp. 443 – 456, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437100000108

[38] G.-J. Wang, C. Xie, S. Chen, J.-J. Yang, and M.-Y. Yang, "Random matrix theory analysis of cross-correlations in the {US} stock market: Evidence from pearsons correlation coefficient and detrended cross-correlation coefficient," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 17, pp. 3715 – 3730, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437113003403

[39] V. Plerou, P. Gopikrishnan, B. Rosenow, L. Amaral, and H. Stanley, "Collective behavior of stock price movementsa random matrix theory approach," *Physica A: Statistical Mechanics and its Applications*, vol. 299, no. 12, pp. 175 – 180, 2001, ¡ce:title¿Application of Physics in Economic Modelling¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037843710100293X

[40] D. Wilcox and T. Gebbie, "An analysis of cross-correlations in an emerging market," *Physica A: Statistical Mechanics and its Applications*, vol. 375, no. 2, pp. 584 – 598, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437106010351

[41] V. Plerou, P. Gopikrishnan, X. Gabaix, and H. E. Stanley, "Quantifying Stock Price Response to Demand Fluctuations," arXiv.org, Quantitative Finance Papers, 2001. [Online]. Available: http://econpapers.repec.org/RePEc:arx:papers:cond-mat/0106657

[42] V. Plerou, P. Gopikrishnan, X. Gabaix, L. A. N. Amaral, and H. E. Stanley, "Price fluctuations, market activity and trading volume," *Quantitative Finance*, vol. 1, no. 2, pp. 262–269, 2001. [Online]. Available: http://www.informaworld.com/10.1088/1469-7688/1/2/308

[43] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data,"

Journal of the American Statistical Association, vol. 91, no. 433, pp. 365–377, 1996. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1996.10476697

[44] E. Lawrence, G. Michailidis, V. N. Nair, and B. Xi, "Network tomography: A review and recent developments," in *In Fan and Koul, editors, Frontiers in Statistics*. College Press, 2006, pp. 345–364.

[45] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 41–56, Oct. 2006. [Online]. Available: http://doi.acm.org/10.1145/1163593.1163602

[46] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "How to identify and estimate the largest traffic matrix elements in a dynamic environment," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 73–84, Jun. 2004. [Online]. Available: http://doi.acm.org/10.1145/1012888.1005698

[47] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic matrices: balancing measurements, inference and modeling," in *In Proceedings of the ACM SIGMETRICS*. Press, 2005, pp. 362–373.

[48] G. Liang and B. Yu, "Maximum pseudo likelihood estimation in network tomography," *Signal Processing, IEEE Transactions on*, vol. 51, pp. 2043–2053, 2003.

[49] B. Yu, J. Cao, D. Davis, and S. Vander Wiel, "Time-varying network tomography: router link data," in *Information Theory, 2000. Proceedings. IEEE International Symposium on*, 2000, pp. 79–.

[50] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 206–217, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/885651.781053

[51] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 301–312. [Online]. Available: http://doi.acm.org/10.1145/863955.863990

[52] M. Roughan, "First order characterization of internet traffic matrices," *International Statistical Institute. Session (55th: 2005: Sydney, NSW)*, 2005.

[53] A. Tsang, M. Coates, and R. D. Nowak, "Network delay tomography," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2125–2136, 2003.

[54] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F. Lo Presti, S. Moon, and V. Paxson, "The use of end-to-end multicast measurements for characterizing internal network behavior," *Communications Magazine, IEEE*, vol. 38, no. 5, pp. 152–159, 2000.

[55] N. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss," *Information Theory, IEEE Transactions on*, vol. 48, no. 1, pp. 26–45, 2002.

[56] N. Duffield, J. Horowitz, and F. Lo Prestis, "Adaptive multicast topology inference," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001, pp. 1636–1645 vol.3.

[57] B. Xi, G. Michailidis, and V. N. Nair, "Estimating network loss rates using active tomography," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1430–1448, 2006. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1198/016214506000000366

[58] E. Lawrence, G. Michailidis, and V. N. Nair, "Network delay tomography using flexicast experiments," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 5, pp. 785–813, 2006. [Online]. Available: http://dx.doi.org/10.1111/j.1467-9868.2006.00567.x

[59] F. Lo Presti, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal delay distributions," *Networking, IEEE/ACM Transactions on*, vol. 10, no. 6, pp. 761–775, 2002.

[60] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network Tomography: Recent Developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004. [Online]. Available: http://dx.doi.org/10.2307/4144399

[61] B. Xi, "Estimating internal link loss rates using active network tomography," Ph.D. dissertation, The University of Michigan, 2004.

[62] G. M. E. Lawrence and V. Nair, "Maximum likelihood estimation of internal network link delay distributions using multicast measurements," in *Proceedings of the Conference on Information Systems and Sciences*, 2003.

[63] D. Ghita, K. Argyraki, and P. Thiran, "Toward accurate and practical network tomography," *SIGOPS Oper. Syst. Rev.*, vol. 47, no. 1, pp. 22–26, Jan. 2013. [Online]. Available: http://doi.acm.org/10.1145/2433140.2433146

[64] Y. Hyun, A. Broido, and k. claffy, "Traceroute and BGP AS Path Incongruities," Cooperative Association for Internet Data Analysis (CAIDA), Tech. Rep., Mar 2003.

[65] M. Luckie, A. Dhamdhere, k. claffy, and D. Murrell, "Measured impact of crooked traceroute," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 14–21, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1925861.1925864

[66] P. Marchetta, W. de Donato, and A. Pescapé, "Detecting third-party addresses in traceroute traces with ip timestamp option," in *Proceedings of the 14th international conference on Passive and Active Measurement*, ser. PAM'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 21–30. [Online]. Available: http://dx.doi.org/$10.1007/978-3-642-36516-4_3$

[67] R. A. Horn and C. R. Johnson, Eds., *Matrix Analysis*. New York, NY, USA: Cambridge University Press, 1986.

[68] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[69] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *Instrumentation Measurement Magazine, IEEE*, vol. 12, no. 2, pp. 16–23, April 2009.

[70] M. Gupta, L. Shum, E. Bodanese, and S. Hailes, "Design and evaluation of an adaptive sampling strategy for a wireless air pollution sensor network," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, Oct 2011, pp. 1003–1010.

[71] L. V. Shum, M. Gupta, and P. Rajalakshmi, "Data analysis on the high-frequency pollution data collected in india," *CoRR*, vol. abs/1301.7231, 2013.

[72] L. Shum, M. Gupta, E. Bodanese, S. Karra, N. Glover, L. Malki-Epshtein, and S. Hailes, "Bias adjustment of spatially-distributed wireless pollution sensors for environmental studies in india," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*, June 2013, pp. 318–326.

[73] P. Szczytowski, A. Khelil, and N. Suri, "Asample: Adaptive spatial sampling in wireless sensor networks," in *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, June 2010, pp. 35–42.

[74] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, April 2004, pp. 124–133.

[75] S. Lin, B. Arai, D. Gunopulos, and G. Das, "Region sampling: Continuous adaptive sampling on sensor networks," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, April 2008, pp. 794–803.

[76] E. K. Lee, H. Viswanathan, and D. Pompili, "Silence: Distributed adaptive sampling for sensor-based autonomic systems," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ser. ICAC '11. New York, NY, USA: ACM, 2011, pp. 61–70. [Online]. Available: http://doi.acm.org/10.1145/1998582.1998594

[77] C.-H. Lung and C. Zhou, "Using hierarchical agglomerative clustering in wireless sensor networks: An energy-efficient and flexible approach," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, Nov 2008, pp. 1–5.

[78] S. E. Zaman and R. J. Mondragon, "Inferring logical network topology by traffic correlations," in *The 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013) (PGNet2013)*, Liverpool, UK, United Kingdom, Jun. 2013.

[79] M. Menth and S. Muehleck, "Packet waiting time for multiplexed periodic on/off streams in the presence of overbooking," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 4, no. 2, pp. 207–229, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1504/IJCNDS.2010.031187

[80] G. Nogueira, B. Baynat, and A. Ziram, "An erlang-like law for gprs/edge engineering and its first validation on live traffic," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, ser. valuetools '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1190095.1190125

[81] J.Virtamo. (2008) Queuing theory/ the m/g/1 queue. [Online]. Available: $http://www.netlab.tkk.fi/opetus/s383143/kalvot/E_mg1jono.pdf$

[82] F. Eyermann. (2008, June) Simulating networks with network simulator 2 (ns-2). ISSNSM International Summer School on Network and Service Management. [Online]. Available: http://www.aims-conference.org/issnsm-2008/02-ns2-exercises.pdf

[83] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary poisson view of internet traffic," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2004, pp. 1558–1569 vol.3.

[84] G. M. LJUNG and G. E. P. BOX, "On a measure of lack of fit in time series models," *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978. [Online]. Available: http://biomet.oxfordjournals.org/content/65/2/297.abstract

[85] MATLAB. (2013) R2013b documentation: Ljung box q test for residual autocorrelation. Natick, Massachusetts.