Traffic Estimation, Prediction and Provisioning in IP Networks

TRAFFIC ESTIMATION, PREDICTION AND PROVISIONING IN IP NETWORKS

BY

SHAHROOZ BEHDIN, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2012) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Traffic Estimation, Prediction and Provisioning in IP Networks
AUTHOR:	Shahrooz Behdin B.Sc., (Electrical and Electronics Engineering) Shiraz University, Shiraz, Iran
SUPERVISOR:	Dr. Ted. H. Szymanski
NUMBER OF PAGES:	xvii, 134

To my parents and grandparents who taught me to stand firm on human standards

Abstract

The study of Internet traffic behavior in a real IP network is the subject of this thesis. Traffic Matrix of a telecommunication network represents the exchanged traffic volume between the source and destination nodes in the network and is a critical input for network studies. However, in most cases, traffic matrices are not readily available. Existing network management protocols such as the 'Simple Network Management Protocol' (SNMP) have been used to gather other observable measures, such as link load observations. The first part of this thesis reviews famous methods and approaches that try to infer and estimate the source-destination traffic matrix from the observable link loads.

Another important subject in networks is to predict bandwidth requirements in the future. The second part of this thesis reviews some existing methods and approaches of traffic prediction. Recently a traffic prediction method which uses multiple Time-Series analysis, each operating on a different time-scale, has been proposed. This method uses multiple 'AutoRegressive Integrated Moving Average' (ARIMA) filters to predict the future bandwidth requirements. Each ARIMA filter operates on a different time scale, i.e., quarter-hour, hour, day, and week. The proposed method associates a weight with each ARIMA filter, and adjusts the weights according to which filter

is currently the most accurate predictor. A review of this newly proposed method is presented. Extensive experimental results have been gathered to test the robustness of the method. The filter coefficients of each ARIMA filter have been varied, and the accuracy of the predicted traffic has been measured. Extensive experimental measurements indicate that the model is very robust, and that large changes to each filter's coefficients have only a small effect on the accuracy. In all cases we evaluated, the method is very robust, predicting short-term future traffic demands with typically $\approx 95\%$ success rates.

Acknowledgements

I am deeply indebted to my supervisor, Professor Ted H. Szymanski, for his support and inspiration. His advice and constructive feedback have encouraged me to properly complete the thesis. He always made himself available for discussion and provided many insights on the research. I consider myself fortunate to have had a chance to work with him as my supervisor.

My deepest gratitude goes to my beloved family who have always been supportive in my life, and provided me the opportunity to continue my education. They have always encouraged me to work hard in my studies. I also would like to thank Dr. Mohammad Bagheri for providing me comments on writing the thesis. I am very thankful to all of my friends for their mental support and being friendly. I enjoyed every minute of working in the Next Generation Networks Laboratory at McMaster University.

Notation and abbreviations

ACF AutoCorrelation Function

AIC Akaike Information Criterion

ARCH AutoRegressive Conditional Heteroskedasticity

ARIMA AutoRegressive Integrated Moving Average

BE Best-Effort

BIC Bayesian Information Criterion

COP Coefficient Of Preference

DCM Discrete Choice Model

DP Distance Parameter

EFC Exact Forecast Curve

EM Expectation Maximization

EWMA Exponentially Weighted Moving Average

GARCH Generalized AutoRegressive Conditional Heteroskedasticity

IID Independent Identically Distributed

IP Internet Protocol

IPF Iterative Proportional Fitting

ISP Internet Service Provider

LPC Lower Probability Curve

MEB Mean Excess Bandwidth

MLE Maximum-Likelihood Estimation

MMSE Minimum Mean Square Error

MSB Mean Satisfied Bandwidth

MVUE Minimum-Variance Unbiased Estimator

NMSPE Normalized Mean Squared Prediction Error

OD Origin-Destination

PACF Partial AutoCorrelation Function

PDF Probability Distribution Function

PoP Point of Presence

QoS Quality of Service

RE Relative Error

SNMP Simple Network Management Protocol

TE Traffic Engineering

TM Traffic Matrix

TSW Time Sliding Window

UPC Upper Probability Curve

VOQ Virtual Output Queue

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the field of Traffic Engineering (TE), the knowledge of network traffic characteristics has an important role to play. In Traffic Engineering, the capability to estimate or predict future traffic demands can help improve our understanding of network. In this chapter we introduce the concept of Traffic Matrix Estimation and Prediction. We present a short background on the importance of having Traffic Matrix estimation tools.

In a telecommunication network, traffic flows between source and destination nodes, or Origin and Destination nodes. In a graph model of a network $G(V, E)$, the vertices $v \in V$ typically denote routers, and the edges $e \in E$ denote links between routers. Let there be N routers in the network, i.e., $|V| = N$. A $N \times N$ Traffic Matrix can be defined, where element $X(s, d)$ represents the volume of the traffic between the source s and the destination d. The Traffic Matrix can also be defined as a column vector with N^2 elements. Sources and Destinations can be defined at any granularity level such as nodes, routers, or Point-Of-Presence (POP). In our work, we refer the source and destinations as routers in a backbone (or wide-area) IP network.

The Traffic Matrix can be an input to the many Traffic Engineering problems. The knowledge of a Traffic Matrix can be used in tasks such as routing, load balancing and QoS provisioning. The goal of a routing algorithm is typically to maximize the traffic load which can be delivered. The goal of load balancing is typically to make the edge loads throughout the network as even as possible. The goal of QoS provisioning in a Differentiated Services network can be to determine the bandwidth requirements and priorities associated with each Differentiated Services traffic class on each link. One of the original design goals of the current INTERNET is to keep the complexity at the intelligent endpoints [8], so that intermediate routers are relatively simple. This goal allows the Internet to scale to a large number of relatively simple interconnected routers. Traffic Engineering tools using knowledge of the Traffic Matrix can assist in improving routing, load balancing and QoS provisioning.

The current Internet is not managed under the supervision of any single unity, or 'Internet Service Provider' (ISP). Each ISP faces different challenges in order to plan for its own resources, topology configurations and equipments needed. The Traffic Matrix (TM) plays an important role as an input to the design, management and decision support tools in an ISP.

Management in the existing Internet is not an automated process. The traditional management approach involves direct human monitoring of link loads and manual intervention, in a centralized control approach of small networks. The large scale of the current Internet makes the traditional managing approach less reliable, since the traditional approach involves direct human monitoring and intervention in a centralized control model. As a consequence, the concept of a 'Future Internet' has been explored by many researchers. The concept of a self-governing 'Autonomic Future Internet'

that governs itself in a decentralized approach, by removing human intervention, has been proposed in recent years [7]. The self-governing loop typically consists of different components, a monitoring component, and data analysis component, and finally a decision-making component.

In general, we can categorize prior research into Traffic Matrices into two major groups. Using figure 1.1, that is borrowed from [21], these concepts are explained.

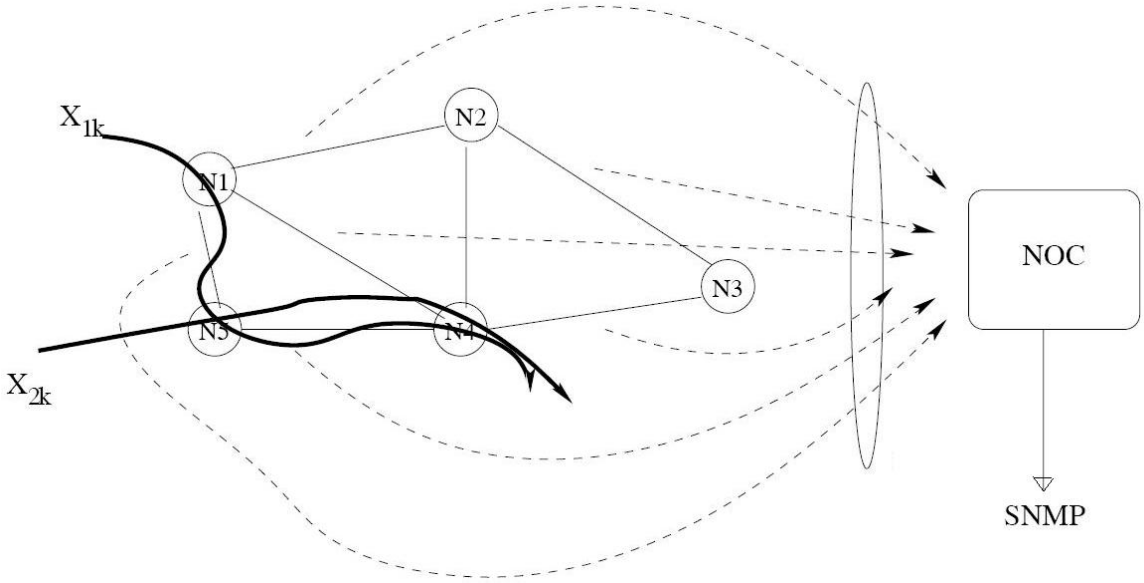

Figure 1.1: TM vs Link Loads, [21]

The dashed lines represent the amount of traffic volume flowing through each links. In traffic engineering context, they are called link loads. On the other hand, the solid lines, show the amount of traffic volume between the source nodes and the destination nodes. A column vector showing the traffic volume exchanged between nodes, is called Traffic Matrix of the network. Two major subjects of research is explained in the following sections.

Traffic Matrix Estimation and Inference

As stated earlier, knowledge of the source-destination Traffic Matrix is not usually available. The volume of traffic moving from a source to a destination can be expressed as the number of bytes or packets in a given time interval, and can be called the 'byte-count' or 'packet-count'. Monitoring all network traffic between nodes to compute a source-destination Traffic Matrix can be resource consuming, and it is not usually done. As a result, there have been many research efforts to estimate the source-destination Traffic Matrix by inference techniques.

One utility tool that nearly all networks have available is the *'Simple Network Management Protocol'* (SNMP). This utility reports the traffic volumes on the links of the network. The source-destination Traffic Matrix can be inferred from the data delivered by SNMP tool. If we denote collected link loads on the network as $y(e)$, the Traffic Matrix expressed as a column-vector to be $x(e)$ and routing matrix as R, then the general relation between Traffic Matrix column vector and measured link loads can be written as:

$$y(e) = R \times x(e)$$

We take a comprehensive look at different approaches used to infer TM from link loads in chapter 2.

Traffic Matrix Forecast and Prediction

Another rich area of research is Traffic Matrix prediction methods. Given the observable link counts on the network and tools to infer the Traffic Matrix, another

important goal will be to predict the TM in the near-term future, based on the history of previous Traffic Matrices and previous link loads on the network.

In chapter 3, we explain different methods of prediction, such as Kalman filters, Flip-Flop filters and Time Series analysis.

However, these two areas of research (Traffic Matrix Inference and Traffic Matrix Prediction) are close and they occasionally overlap and cannot be distinguished clearly.

Contribution of the thesis

In our thesis, we review a recently-proposed Traffic Matrix and link-load prediction algorithm, that can predict the near-term future traffic demands based upon the past traffic history [37]. As shown in [9], at the packet-level Internet traffic often shows a self-similar behaviour and long-range dependencies. Different approaches have been proposed to model these dependencies and use them for prediction. One of the main approaches for traffic prediction is the time series analysis.

The recently-proposed traffic prediction algorithm we review, uses multiple Autoregressive integrated Moving Average (ARIMA) filters [37]. Each filter operates on a different time scale, i.e., quarter-hour, hour, day and week. The algorithm considers the traffic flow between each source-destination pair as a time series, and applies multiple ARIMA filters to each time-series. A sequence of observed data that are ordered in time is called a 'time series'. One of the main characteristic of time series is the importance of the order of observations [1]. While in other statistical analyses the order of observed data is not considered, in time series analysis the order of observations is explicitly recognized. The general form of ARIMA filter can be written

as:

$$X_n = \sum_{i=1}^{p} \varphi_r X_{n-i} + \sum_{j=1}^{q} \theta_r \varepsilon_{n-j} + \varepsilon_n, n > 0$$

where X is random variable, representing network traffic in our work, p and q are autoregressive and moving average orders, respectively and ε is considered as a sequence of uncorrelated random variable representing traffic changes. In literature ε is called innovation [26].

In the recently-proposed traffic prediction algorithm that we review, each filter operates on a different time scale, i.e., quarter-hour, hour, day and week. The algorithm assigns a dynamic weight to each filter, and combines all the filter predictions according to their weights, to yield a single predictor. The weights change dynamically, according to how accurate each filter is.

The prediction algorithm is tested on real traffic demand matrices reported in 15-minute intervals for the 'Geant' European backbone network, in a 4-month period. One of the main goals of this thesis is to test the robustness of the proposed algorithm. The filter coefficients for each filter have been varied, and the accuracy of the model was then measured. Our experiments indicate that routers can estimate their future link traffic demand with excellent precision.

Structure of the thesis

Chapter 2

In chapter 2, we review different methods for inferring the source-destination Traffic Matrices from the link counts obtained from the SNMP tool.

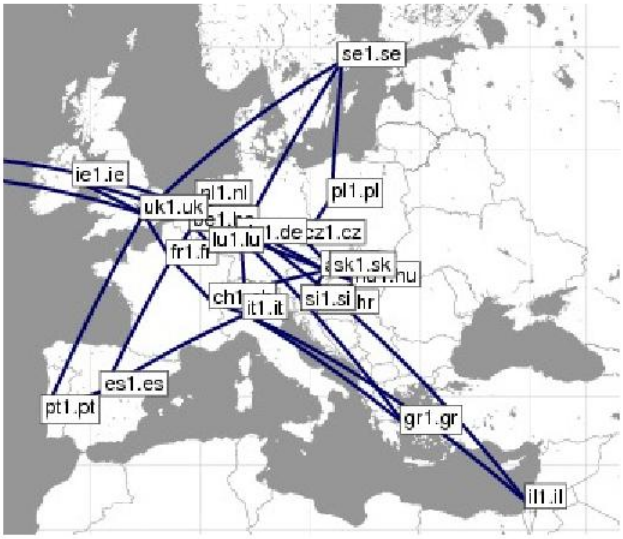

Figure 1.2: Geant Network Topology (Borrowed from http://sndlib.zib.de/)

Chapter 3

In chapter 3, we review several different forecasting approaches. Kalman filters are discussed first and then flip-flop filters are introduced. Several different utilities used in time series analysis are discussed.

We will also explain how prediction algorithms model can be used in a proposed Future Internet structure in [36] and [29].

Chapter 4

In chapter 4, we will review the recently proposed traffic prediction algorithm in [37].

Chapter 5

In chapter 5, we present our experimental results on evaluating the robustness of the proposed algorithm. Several filter coefficients have been varied, and the accuracy of the model has been evaluated. We use real measurements of the link loads reported from Geant network to verify our model. Some of our experimental results have been published in a conference paper in the 2012 IEEE ICC Workshop on the Future Internet V (FutureNet V), held in Ottawa on June 2012 [37]. The conference paper is entitled *'Traffic Provisioning in a Future Internet'*.

Chapter 6

Chapter 6 presents the conclusions, where we conclude our experiments.

Chapter 2

Traffic Matrix Estimation Methods

In this chapter, we will study different approaches to estimate the Traffic Matrix from the Link Loads. We will explain different Traffic Matrix estimation and inference methods in detail. There has been numerous research conducted in this context, and some effort has been made to classify these methods [6]. However we only consider the most famous and basic approaches that have gained sufficient attention. Methods such as Gravity models, Choice model, Tomogravity, Bayesian and Linear Programming are reviewed. The key difference between these models is the source of extra information they are using to make an estimate in an ill-posed situation.

2.1 Review of Estimation Methods

The Traffic Matrix (TM) is a crucial input for different applications ranging from control, measurement, QoS provisioning, to some other security applications like intrusion detection. However, Traffic Matrices are not readily available in large operational IP networks. In networks on the other hand, we can gather information of the traffic

volume going through the links.

In a large-scale IP network, with n nodes and L links, there are $N = n^2$ Origin-Destination flows. The Traffic Matrix (TM) of a network represents the traffic volume exchanged between Origin and Destination nodes of a network. The Link Loads, however represent the load of traffic on each of the links. The Traffic Matrix and Link Loads can be denoted as $x(t) = (x_1(t), x_2(t), x_3(t), \ldots, x_N(t))^T$ and $y(t) = (y_1(t), y_2(t), ..., y_L(t))^T$, respectively. $x_n(t)$ stands for the traffic volume, exchanged between OD pair n at time slot t, and $y_n(t)$ shows the traffic volume passing through link n at time slot t [16]. In a large IP network, the Traffic Matrix and the Link Loads are correlated to each other with the routing matrix, that can be shown as $R = (R_{ij})_{L \times N}$, where R_{ij} is equal to 1 if the OD pair j uses the link i in the network. The 'Simple Network Management Protocol' (SNMP) is used to gather the traffic load on network links. On the other hand Routing Protocols can also be considered as a known parameter, as we have the knowledge of the network configuration.

The relation between Link Loads and Traffic Matrices can be shown in the equation 2.1:

$$y(t) = Rx(t) \tag{2.1}$$

By using the SNMP and gathering information of Link Loads, and also given the knowledge of the routing algorithm from the network configurations, the only unknown variable in the equation 2.1 appears to be Traffic Matrix, $x(t)$. Therefore, theoretically the TM and vector x, can be found by solving equation 2.1.

This process of calculating the TM based on Link Loads and routing protocols cannot be trusted in the large-scale IP networks, when the number of the nodes is much larger than the number of the links, i.e. $N >> L$. In that situation the equation

$y(t) = Rx(t)$ is considered as a highly under-constrained linear problem that can have infinite solutions for the x(t). Fitting a TM into equation 2.1 makes this estimation an ill-posed (under-constrained) problem. The main challenge in the literature is how to overcome this problem. Different research has been conducted in academia to solve this problem. The key difference between these methods is how they bring in and use different extra information sources to make the problem identifiable.

2.2 Gravity Models

The simple Gravity Model is considered as the most basic estimation method [6]. We know Newton's famous gravity law:

$$F = K \times \frac{M_1 \times M_2}{D^2}$$

where F is the attraction force in Newtons between masses $M1$ and $M2$ in Kilograms, D is the distance between these two masses in Meters, and K is the gravitational constant which is approximately equal to $6.674 \times 10^{-11} Nm^2kg^2$.

In the field of networking, we can use this gravity law for TM estimation. This model is therefore named after Newton's Law of gravitation. Newton's famous gravity law states that the attractive force between two objects is proportional to the product of the masses of the two objects individually, and is inversely proportional to a function of the distance between those two objects. In networks we can also model the data exchanged between two nodes (s,d) as proportional to the product of the total volume of traffic transmitted from the source s and received by the destination d individually, and inversely proportional by a function related to the distance between

that OD pair.

Although this method looks a bit unfamiliar in the field of networks, the main justification behind the Gravity Model is that if we have knowledge of the total traffic patterns of individual nodes in the network, the gravity method can estimate the exchanged traffic volume between pairs of nodes based on how busy nodes are, and how nodes are connected.

The basic gravity model is proposed by Kowalski [10] as following:

$$x_{ij} = k_i \frac{O_i T_j}{d_{ij}^{\alpha_i}} \tag{2.2}$$

with the O_i being whole traffic originating at node i, T_j the whole traffic terminating at node j. d_{ij} is defined as a function of distance between node i and j. α is called distance parameter. x_{ij} presents the amount of traffic exchanged between node i, the source node, and node j, the destination node. k is the normalizing constant that can be obtained as:

$$k_i = \frac{O_i - x_{ii}}{\sum_{j \neq i} \frac{O_i T_j}{d_{ij}^{\alpha_i}}}$$

As we can see in equation 2.2, there are two attraction terms that are multiplied together, and then divided by a distance factor.

At the second step, they try to fit the model to the real data of the network, to find the only unknown part of the equation 2.2, i.e., the distance parameter. In [10] they claim that if the equation 2.2 holds, then by substituting the real traffic

measured from the network for x_{ij}, then we can write:

$$\frac{b_{ij}}{T_j} \approx k_i \frac{O_i}{d_{ij}^{\alpha_i}} \tag{2.3}$$

where b_{ij} is the real measurement of network traffic. They found the left-hand side of equation 2.3 very scattered such that it cannot be used to fit a smooth curve.

Therefore they used another condition to fit their model to the real traffic measurements. To fit the model to the data, they try to minimize equation in 2.4:

$$\sum_{i \neq j} |x_{ij} - b_{ij}| \tag{2.4}$$

Alternatively:

$$\sum_{j \neq i} \frac{(x_{ij} - b_{ij})^2}{x_{ij} + bij} \tag{2.5}$$

where x_{ij} is the computed exchanged traffic between node i and j, and b_{ij} is the real network traffic between source node i and destination node j.

The distance factor, α_i that minimizes one of the 2.4 or 2.5 is the distance parameter (DP) of the exchange i. After obtaining α_i, equation 2.2 can be used to calculate x_{ij} mathematically.

In the last step, they define the COP (Coefficient of Preference) between node i and j, c_{ij}, to find the perfect fit between the mathematical model and the measured data:

$$c_{ij} = \frac{x_{ij}}{b_{ij}} \tag{2.6}$$

where c_{ij} is the COP between node i and j, x_{ij} is the mathematically calculated traffic between node i and j using equation 2.2, and b_{ij} is the real measured traffic between node i and j.

The final estimate of traffic exchanged between source node i and destination node j can be calculated as:

$$x'_{ij} = c_{ij} k_i \frac{O_i T_j}{d_{ij}} \tag{2.7}$$

where the final estimate, x'_{ij}, is the final fit between data and the model.

In [11] the authors consider the estimation process as a two-step process, the first is to extend the gravity model by bringing in the ingress and egress link load information and finding a Traffic Matrix estimate as the initial point for the second step. In the second and more accurate step, they apply quadratic programming in the potential solution space by using the tomography model (defined ahead). These potential solutions are admitted by comparing to the gravity model solution.

To extend the gravity model, they use extra link load information. They first categorize the nodes and links to two groups. All links and nodes internal to the network are considered as backbone nodes and backbone links, while the other nodes and links are counted as edge nodes and edge links. To exploit more information they consider the network as connected to other autonomous systems via edge links. An edge link itself can be in two states. The edge link can be an 'access link' which connects costumers to the network, or a 'peering link' that connects non-costumer autonomous systems to the network. This is illustrated in the figure 2.2.

The authors then assumed that the traffic going through the backbone of the network from one peer to another is negligible. The main idea here is to form a friction factor. They formed this by combining the normalizing and coefficient functions all

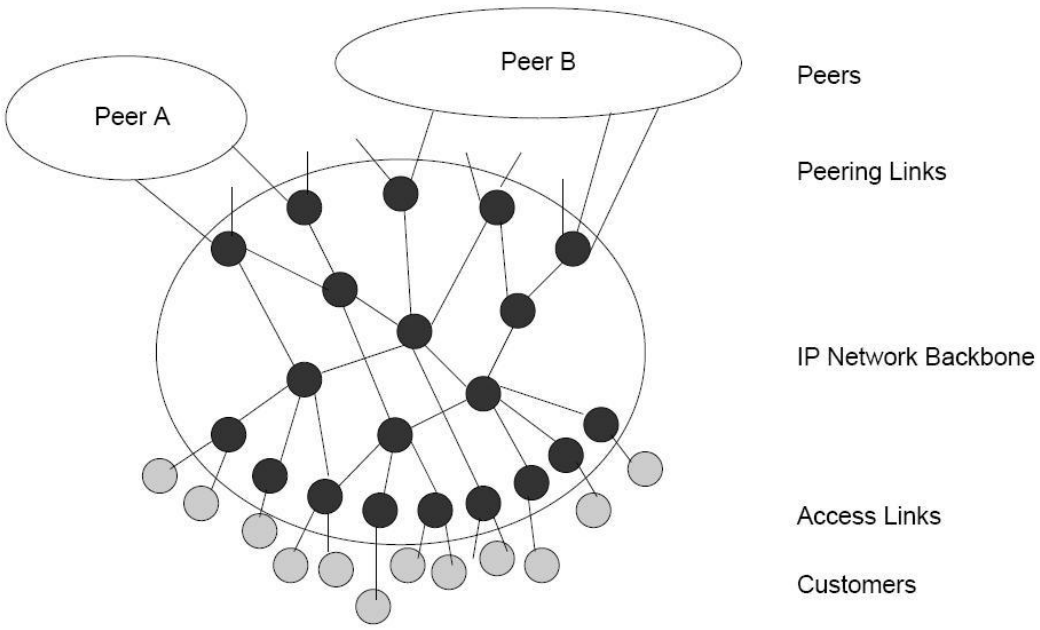

Figure 2.3: Categorize Nodes and Links, [11]

together to have a simplified form of gravity model:

$$x_{ij} = \frac{O_i \times T_j}{f_{ij}} \tag{2.8}$$

where x_{ij} represents the traffic entering the network from source node i and leaving the network from destination node j. f_{ij} is the matrix element representing the friction factor. Comparing this equation with original gravity model equation 2.2, one can observe that friction factor stands for the normalizing constant and the distance function. As explained in [11] the friction factor contains the locality information

specific to different source-destination pairs. It explains how different nodes are connected with each other.

The authors noticed that inferring the $R \times R$ friction factor matrix between all source-destination pairs is the equivalent problem of inferring the Traffic Matrix. As a result, they try to estimate the friction factor. In the final gravity model they are targeting to build, the exchanged traffic volume between nodes is proportional to the traffic originating from the source node and exiting the destination node.

To estimate the friction matrix and build the final model, they incorporate the link load information obtained by the aforementioned link classification. The authors adapt the gravity model to treat customers and peering links differently.

Each peer P_i has a set of peering edge links connecting that peer to the network. Each customer is connected to the network via access links $a_1, a_2, \dots$. Edge links are labeled as $p_1, p_2, \dots$. The set of all links connecting peer P_i to the network is $\bar{P}_i$.

They claim, based on the dominant INTERNET policies, that we can represent the traffic in one of the following forms:

1. Outbound Traffic from access link to peering link:

Their first assumption is that the traffic from a single access link to a given peer exits the network on the same peering link. They denote the peering link used for traffic from access link to the peering link by $X(a_i, P_j)$. With this assumption in mind they consider that the traffic exiting to a specific peer comes from each access link proportional to the traffic originating at that access link. (The set of all access links is denoted by A)

Therefore we can write the equation for the outbound traffic between the access link $a_i \epsilon A$ to peering link $p_m \epsilon P_j$ as:

$$T_{outbound}(a_i, p_m) = \frac{T^{in}_{link}(a_i)}{\sum_{a_i \epsilon A} T^{in}_{link}(a_k)} T^{out}_{peer}(P_j) \tag{2.9}$$

if $p_m = X(a_i, P_j)$.

In equation 2.9, $T_{outbound}(a_i, p_m)$ is the outbound traffic from access link $a_i \epsilon A$ to peering link $p_m \epsilon P_j$, $T^{in}_{link}(a_i)$ is the traffic entering the network from the access link a_i, $T^{out}_{peer}(P_j)$ is the traffic exiting the peering link (p_m).

2. Inbound traffic from peering link to access link:

Here they also assume the proportionality rule. Traffic entering the network from each peer is split among the access links in proportion to the outbound traffic, which can be written as:

$$T_{inbound}(p_i, a_j) = T^{in}_{link}(p_i) \frac{T^{out}_{link}(a_j)}{\sum_{a_k \epsilon A} T^{out}_{link}(a_k)} \tag{2.10}$$

where $T_{inbound}(p_i, a_j)$ is the inbound traffic from peering link p_i to access link a_j, $T^{in}_{link}(p_i)$ is the traffic entering the network from peering link p_i and $T^{out}_{link}(a_j)$ is the traffic exiting the network from link a_j.

3. Internal traffic from access link to access link:

They consider the traffic going through the backbone and exiting an access link as proportional to the traffic originating at the other access links.

The following equation represents the internal traffic model:

$$T_{internal}(a_i, a_j) = \frac{T^{in}_{link}(a_i)}{\sum_{a_k \epsilon A} T^{in}_{link}(a_k)} T^{out}_{internal}(a_j) \tag{2.11}$$

where $T_{internal}(a_i, a_j)$ is the traffic entering the network from access link a_i and exiting the node a_j, $T^{in}_{link}(a_i)$ is the traffic entering the network from access link a_i

and $T^{out}_{internal}(a_j)$ is the portion of the whole traffic volume exiting the access link a_j excluding the portion originating at the peering links.

2.3 Tomogravity

The term 'Network Tomography' was first introduced in [12] as determining the TM in a network, by using the measured traffic volume flowing along directed links of the network. The authors in [11] use this approach to fit the possible gravity solutions to the equation 2.1. As mentioned in [6], the main deficiency of the gravity model is that it doesn't use the link count information. Therefore the authors in [12], proposed to use the final gravity model, x_g as a initial point for the Tomogravity model.

They then refine the gravity model solution by calculating a least-square function and minimizing the Euclidean distance to the gravity model solution subject to the tomographic constraint. The process is to solve the following quadratic programming problem of the L_2 norm of a vector.

$$\begin{aligned} &\min \ ||(x - x_g)|| \\ &s.t. \quad ||Rx - y \ || = 0 \end{aligned} \tag{2.12}$$

where $||.||$ stands for L_2 norm and x_g is the gravity model solution. This equation tries to find the closet solution to the gravity model, that also satisfies constraint $||Rx-y|| = 0$. Another possible objective function is equation 2.13 where the weighted

least squared solution are being considered:

$$\begin{aligned} &\min \ ||\frac{(x - x_g)}{\mathbf{w}}|| \\ &s.t. \quad ||Ax - y \ || = 0 \end{aligned} \tag{2.13}$$

$\mathbf{w}$ is the weight vector, and dividing is performed component-wise. They use $||\frac{(x-x_g)}{\mathbf{w}}||$ as the objective function to minimize the distance to the gravity model solution based on the Tomogravity constraint.

2.4 Linear Programming

The relationship between link counts and the OD traffic volumes can be sorted into a set of linear equations. To find the Traffic Matrix, we need first to define an objective function and then solve the problem with some standard techniques. However defining a suitable objective function is a subject of much research. As a simple example, the authors in [13] considered $||x||$ as the objective function:

$$\begin{aligned} &\min \ ||x||^2 \\ &s.t. \quad y = Rx \end{aligned} \tag{2.14}$$

They then mentioned that this approach is not efficient for the TM estimation problem, since it deals mostly with the OD pairs that are have comparable sizes. Another objective function has been proposed in [14]. We know that the load on a link is the sum of all OD traffic volumes flowing through that link. Therefore they built a weighted sum of all OD traffic volumes as the objective function:

$$
\begin{aligned}
&max \sum_{j=1}^{N} w_j x_j \\
&s.t. \sum_{j=1}^{N} R_{lj} x_j \leq y_l \\
&s.t. \sum_{l=(i,j)} y_l R_{lk} - \sum_{l=(j,i)} y_l R_{lk} = \begin{cases} x_k & \text{if j is the source of k ,} \\ -x_k & \text{if i is the source of k ,} \\ o & \text{otherwise.} \end{cases}
\end{aligned} \tag{2.15}
$$

where x_j represents the traffic volume between OD pair j, and w_j represents the weight assigned to the OD pair j. N, is the number of OD pairs. R is the routing matrix. The first constraint is the link constraint, which ensures that the volume of traffic assigned to each link is less than or equal to the capacity of the link.

The second constraint is the flow conservation constraint. As explained in [6], it makes sure that the amount of flow entering a node is equal to the amount of the flow exiting that node, except for sources and destinations.

The main issue associated with this approach is finding the suitable weight set. Weights assign a value to each of the OD pairs, based on the interaction between the pair. Goldschmidt proposed weights representing the path length for each OD pair. In [14] the author conclude that constant weights, i.e. $w_i = 1 \ \forall i$, may not be suitable as this approach gives more bandwidth to the closer nodes, while nodes with multiple hops between them will get zero bandwidth.

the authors in [16] counted the link counts constraint as hard constraints, rather

than statistical data. In other methods such as the Bayesian approach or EM methods, the link counts are used as statistical data. Here the objective function depends on the weight function. For example, the function that is the linear combination of all demands tries to maximize the load on all links in the network. As we are trying to maximize the objective function, the model tries to assign more bandwidth to the OD pairs with the larger weights.

2.5 Statistical Approach

The authors in [16] categorized the Bayesian and the Expectation Maximization as statistical approaches, with different specific distributions for different Traffic Matrix components. In the statistical approach, the procedure basically tries to find a specific distribution for the OD flows.

2.5.1 Bayesian Approach

In the Bayesian approach, as explained in [16], the main idea is to find the conditional probability distribution for the Traffic Matrix elements, given the knowledge of the link counts and a prior distribution. To achieve that, we need to have a prior distribution for the Traffic Matrix, $p(X)$. In [17] they proposed a Poisson distribution as the prior distribution for the traffic volumes between OD pairs:

$$x_i \sim Poisson(\lambda_i)$$

where x_i is the i-th OD pair, and λ_i is the mean rate of OD pair i, x_i.

After assuming a prior distribution for the traffic, the next step is to determine the

parameters of that distribution. Usually, as in Bayesian approach, this step involves the mean rates of the traffic between each OD pairs. The mean rates between OD pairs can be denoted as a vector $\Lambda = \{\lambda_1, \lambda_2, .., \lambda_N\}$, where N is the number of OD flows. Since Λ is unknown we need to define a prior value for the mean values, that leads us to a joint model $P(x, \Lambda)$, so that we can reach the distribution of OD pairs. As a result the goal is to compute the $P(x, \Lambda|y)$, that is the joint distribution of x and Λ conditioned on the observed link counts, y.

It is, however, computationally hard to compute the posterior distribution $P(x, \Lambda|y)$. Therefore, iterative Markov Chain Monte Carlo methods are used to draw a large number of samples to represent a complete histogram of the desired distribution [16].

This iterative approach starts with assuming a prior x^0 for the Traffic Matrix, that can be an outdated TM, and continues by taking two steps iteratively. First step is to draw a value for Λ from equation 2.16:

$$\Lambda^i = p(\Lambda|x^i, y) \tag{2.16}$$

and then draw a value for x^{i+1} using equation 2.17:

$$x^{i+1} = p(x|\Lambda^i, y) \tag{2.17}$$

using the obtained Λ^i, from equation 2.16. In equations 2.16 and 2.17, x represents TM, Λ represents the vector of mean rates, and y represents observable link counts.

In equation 2.16 and 2.17, x represents traffic matrix, $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$ denotes the vector of mean rates of OD flows and y is the observed link counts. Here i represents the iteration step number.

They continue this loop until a feasible solution is found. The feasibility here is the positivity constraint on the Traffic Matrix elements.

The authors in [17] went further to compute the Traffic Matrix in an easier way. They reordered the routing matrix columns as $R = [R_1 R_2]$ with R_1 being a nonsingular $r \times r$. With this approach, they also rewrite the Traffic Matrix as $x = [x_1\ x_2]$. With this formulation, it can be shown that x_1 can be calculated using $x_1 = R_1^{-1}(y - R_2 x_2)$, [17]. So it only suffices to compute the estimate for the x_2; $p(x_2|\Lambda, y)$. After that we can compute the rest of the Traffic Matrix elements, x_1, with straightforward algebra.

In the Bayesian approach, the whole model needs a good prior estimate for x. In this approach the prior estimate is combined with the link count information, therefore it has a large influence on the final solution. The authors, for example, mentioned that choosing a uniform prior estimate may lead to a over-estimation of low rates and vice vera, an under-estimation of higher rates.

2.5.2 Expectation Maximization Algorithm

Most of the Traffic Matrix estimation methods bring in some extra source of information. In the Maximum likelihood Estimation (MLE) the source of extra information is the second moment of the Traffic Matrix elements. The MLE approach usually is in need of several consecutive link count measurements, therefore we can consider last K link count observations. As the size of the Traffic Matrix is quite large, numerical methods are used to find the likelihood estimate.

On the other hand, the EM algorithms are typically used in cases with missing data. The authors in [6] explained why EM is applicable in this context. In the Traffic Matrix estimation context, the TM cannot be observed, but a smaller set of linear

combination of the traffic volumes can be observed in the link count measurements. However, in the EM methods, a parametric distribution should be assumed for the missing data.

The Expectation Maximization algorithm is a numerical method to solve the MLE problems. Here we present the studies done in the [15]. We follow the EM method with the same symbols and notation they used. In EM method, the distribution considered for the traffic between OD pairs is Gaussian, i.e. $x \sim Normal(\lambda, \Sigma)$, where λ is the mean rate and Σ denotes the covariance matrix. The traffic volumes between each pair of nodes are considered as independent random variables with Gaussian distribution. Also because of the relation between the link counts and OD pairs, $Rx = y$, it implies the following formula for the link counts:

$$y \sim Normal(R\Lambda, R\Lambda R')$$

The parameters here are $\Lambda = (\lambda_1, \ldots, \lambda_c)$ that is again the vector of mean rates. The covariance matrix, Σ, can be written as:

$$\Sigma = \phi \; diag(\sigma^2(\lambda_1), \ldots, \sigma^2(\lambda_c))$$

where ϕ is the scale parameter, and $\sigma^2(.)$ is a known relationship between mean and variance. The authors considered the power law form for σ^2 and define it as follows:

$$\sigma^2(\lambda) = \lambda^b$$

with that in mind, the covariance matrix, Σ, can be rewritten as:

$$\Sigma = \phi \ \ diag(\lambda_1^b, \ldots, \lambda_c^b)$$

The assumption that the covariance of OD pairs is diagonal equates to the assumption of independence of the OD pairs traffic volumes.

The authors in [15] believe $b = 2$ can fit their data well. Hence they consider it throughout their researches. The scale parameter, ϕ, also is unknown and should also be estimated along with λ to find the $\Sigma_j = \phi \ \lambda_j^b$. As a result, we have to estimate two sets of parameters that can be shown as $\theta = (\Lambda, \phi)$ together.

This method, considers multiple sets of link loads. If L is the number of links in the network, then $y_t = (y_t^1, \ldots, y_t^L)$ is the loads of all the links at time t. This method uses consecutive set of K SNMP measurements; $(y_1, y_2, \ldots, y_K)$.

The log-likelihood of this set of observed link loads can be computed by the following log-likelihood function:

$$l(\theta|y_1, y_2, .., y_k) = -\frac{K}{2} log|R\Sigma R'| - \frac{1}{2}\sum_{k=1}^{K}(y_k - R\Lambda)'(R\Sigma R')^{-1}(y_k - R\Lambda) \qquad (2.18)$$

where $\theta = (\Lambda, \phi)$ is the set of unknown parameters λ and ϕ needed to be estimated, K is the number of consecutive observed link counts, y_k is the link loads at time k, R is the routing matrix, Σ is the covariance matrix and $\Lambda = (\lambda_1, \ldots, \lambda_c)$ is the vector of mean rates.

The maximum likelihood estimate, $\hat{\theta}$, can be computed as [19]:

$$\hat{\theta} = argmaxl(\theta|y_1, y_2, .., y_k)$$

To solve this optimization problem, EM is used. It runs as an iterative procedure. In this approach a conditional expectation function is defined and then at each step, they try to optimize it [19]. The conditional expectation function is defined as:

$$Q(\theta, \theta^{(i)}) = E(l(\theta|x)|y, \theta^{(i)}) \tag{2.19}$$

In this equation, $Q(\theta, \theta^{(i)})$ is the conditional expectation function that we must optimize at each iteration, y represents the set of observed link counts, θ^i is the $i-th$ estimate of our unknown parameters, and finally the complete data log-likelihood [19] can be obtain from:

$$l(\theta|x_1, x_2, .., x_K) = -\frac{K}{2}log|\Sigma| - \frac{1}{2}\sum_{k=1}^{K}(x_k - \Lambda)'(\Sigma)^{-1}(x_k - \Lambda)$$

where x's stand for the OD flows.

Each step of EM algorithm consists of two steps, the Expectation step and Maximization step. At iteration i, it first calculates the conditional expectation function $Q(\theta, \theta^i)$ using the $i-th$ estimate of θ. In the second step, we draw a new value θ^{i+1} by optimizing the conditional expectation function $Q(\theta, \theta^i)$.

$$\theta^{i+1} = argmaxQ(\theta, \theta^i)$$

In [19] they claim that it can be shown θ^i converges to a minima of the likelihood

function.

To find the maximum of the Q function in terms of θ we need to take the derivate of Q in terms of θ and equal it to zero, $\frac{\partial Q}{\partial \theta} = 0$, to find the parameters that maximizes the expectation.

After the final estimate obtained, $\hat{\theta}$, the traffic volume between OD pair j, at time t can be estimated using $\hat{x}_{j,t} = E[x_{j,t}|\hat{\theta}, y]$. They declared that by knowing the parameter, θ, then $E(x|y, \theta, t > 0)$ has minimum square prediction error for estimating the traffic volume between OD pairs, x.

2.6 Choice Model

The total amount of traffic leaving ingress node (source node) can be calculated by determining all outgoing links' counts. The outgoing traffic volume, that is denoted as O_i, is split to different portions between different egress nodes (destination nodes). The fraction of outgoing traffic initiated at POP i, and destined to POP j, can be shown as α_{ij}. Therefore the traffic transferring from POP i to POP j can be formulated as:

$$x_{ij} = O_i \alpha_{ij}$$

with the constraint $\Sigma_j \alpha_{ij} = 1$.

In this equation x_{ij} is the traffic between source node i and destination node j, O_i is the traffic originating at the source node i and α_{ij} is a factor showing the fraction of traffic originating at the source node i and ending at node j. In fact α explains how the traffic is distributed across the egress POPs. Now the Traffic Matrix estimation

can be reduced to estimating these factors. The authors in [16] and [18] proposed to look at the ingress POPs as decision makers, so that we can use the Discrete Choice Model (DCM) approach.

In general, there are many factors influencing the traffic across the network. But all these factors can be classified in two general categories. One category describes the user behavior, which describes how the users initiate the traffic. The other category is based on the network design and configurations. Once the packet of data is required to be transferred by the user, it is the network configuration that decides which path should be chosen and which egress node is the destination.

As explained in [6] each ingress POP can be thought as a decision maker about which egress node should be the destination. While obviously the ingress nodes are not intelligent devices,this model can be interpreted such that the ingress nodes are making decisions based on different factors and attributes to finally choose the destination. These attributes (w_i^j) can be used to model the decision process. These attributes describe the attractiveness of node j as a possible destination for the source node i (i.e., how likely is it that node i will choose node j as the destination).

The decision process can be modelled using utility maximization criterion. A utility function can be defined as a weighted sum of different attributes (w_i^j):

$$V_d^s = \sum_m \mu_m w_d^s(m) + \gamma_d$$

In this equation, the utility function, V_d, to model the decision making process is defined. w represents different attributes, μ_m is the relative importance of the attribute $w(m)$, γ is the scaling term representing the amount of attractiveness of the egress j, not included in the weighted average of attributes.

The authors proposed two models. First they only considered a single attribute model, which is the total amount of traffic coming into the egress node. The second model adds the total amount of traffic volume leaving the ingress node as the second attribute. They claimed that the second model works better in their works. The second model can be written down as:

$$V_d^s = \mu_1 w_d^s(1) + \mu_2 w_d^s(2) + \gamma_d$$

or simply :

$$V_d^s = \mu_1 w_d(1) + \mu_2 w^s(2) + \gamma_d$$

where V represents the utility function used to model the decision making process, and defined as the weighted average of different attributes. μ's are weights of attributes and w's are attributes. In this model they consider only two attributes.

The probability that source node s will choose node d as the destination can be modelled with a so-called multinomial logit or 'mlogit' model [18]:

$$\alpha_{sd} = \frac{e^{V_d^s}}{\sum_k e^{V_k^s}}$$

Therefore the traffic volume between source and destination can be modelled as:

$$x_{sd} = O_s \frac{e^{V_d^s}}{\sum_k e^{V_k^s}} \tag{2.20}$$

where k is the number of alternative destinations for node s.

In [18] the authors believe the mlogit choice model solution and the gravity model solution can be used as a starting point in the statistical approaches (EM). They compared the outcome of two approaches in figure 2.4.

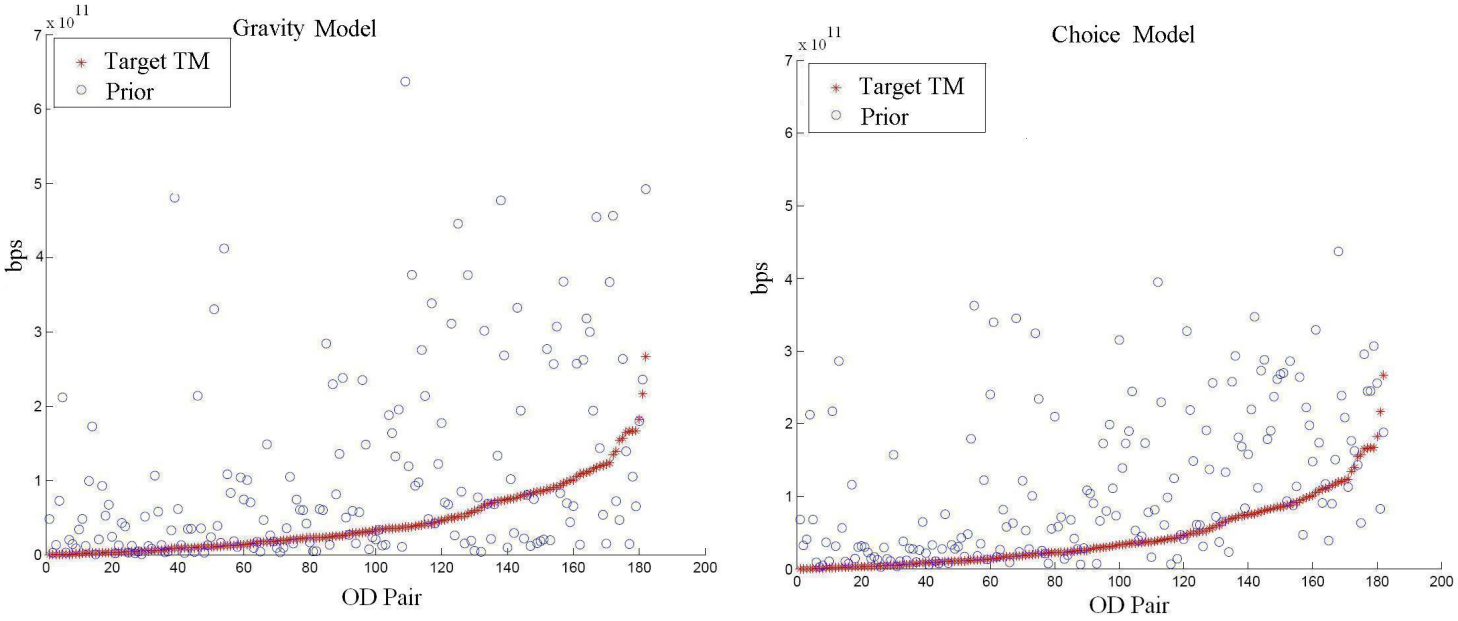

Figure 2.4: Gravity Model, Choice Model, [18]

They compared the starting points produced by gravity model and mlogit choice model with a synthetic traffic matrix. As shown in these figures, the blue circles are prior estimates produced by the mlogit choice model and the gravity model. These circles are scattered around the values of the target Traffic Matrix. Although they are very similar to each other, the circles are closer to the target TM in the mlogit choice model, and they are more variable in gravity model. This result is expected because the mlogit model has free parameters that can be calibrated to get a better accuracy, as opposed to the gravity model which has no free parameters. Given partial direct measurements of the Traffic Matrix, we can find the parameters of this model, μ_1, μ_2, γ_d.

2.7 Conclusion of Chapter 2

In chapter 2, we took a comprehensive look on famous methods and approaches of estimating the Traffic Matrix of a network. Although the Traffic Matrix of a network is a crucial asset in various applications, it is either impossible or prohibitively expensive to observe it directly from the network.

On the other hand, other available tools in the network such as the *'Simple Network Management Protocol'* (SNMP), can be used to gather Link Loads of the network. Equation 2.1 relates the Link Loads and the TM of a network. In large IP networks, the number of nodes is much larger than the number of links, therefore equation 2.1 is highly under-constrained and we cannot directly compute a single TM from this equation.

Different methods for estimating the Traffic Matrix have been reviewed in this chapter. The basic Gravity model is the first and simplest approach that was reviewed. The authors in [11] extended this approach and used it as an initial point for the Tomogravity approach. Equation 2.1 can be thought as a set of linear equations. As a result some research has been conducted to address the estimation problem from the linear programming perspective. Two statistical approaches were reviewed next. For the Bayesian method, as mentioned in [16], the goal is to find a conditional probability distribution of the OD demands given the measured link counts. In the EM method, a Gaussian distribution is considered for the OD pairs. Then EM is used to estimate the model's parameters. In the last step, an iterative proportional fitting (IPF) algorithm is implemented.

The last reviewed method was choice model. In the choice model, the attractiveness of the destination node to the source node is modelled as a weighted sum

of various attributes. The authors in [16] and [18] thought of the ingress node as a decision maker to decide which node should be chosen as the destination node.

Chapter 3

Traffic Prediction; Methods and Applications

In this chapter we are going to explain Traffic Matrix prediction methods and applications. In general, estimation methods try to build the Traffic Matrix of the network based on the link count information, while prediction methods usually try to predict the TM of the network in the next time-slot or next provisioning interval, given a set of past Traffic Matrix observations and past link count information. Estimation methods were reviewed in chapter 2. Here, we first take a look at applying Kalman filter in estimating the TM. An explanation of using 'flip-flop' filters will be presented next. Our main model is based on time series analysis and ARIMA filters; therefore we take a comprehensive look at famous utilities used in time series analysis, such as ARCH and GARCH filters.

3.1 Traffic Prediction using Kalman filter

One of the famous tools in the prediction context is the Kalman filter. In [21], the authors developed a Kalman filter to estimate and predict the Traffic Matrix. The main idea of their approach is to consider the traffic demands as the state of the system. Therefore the TM is the global state of the network. They define a model for the OD flows to include both temporal and spatial correlation in flows. The authors define a state-space system to incorporate both the OD flow model and the observable link counts. As the OD flows are not directly observable, they call these the 'hidden state' of the network. Once we have a state-space model incorporating both the OD flow model and the observables, the goal is to estimate the future state of the system from the observable link counts. They propose using the Kalman filter for both estimation and prediction. Here we follow their approach using the notation they have developed.

We discussed the relation between observed link counts based on SNMP and OD flows in previous chapter:

$$y_t = A_t \times x_t + V_t$$

where y_t is the observed link count vector, A_t is the routing matrix, and x_t is a vector of OD flow rates. To capture the errors of data collection, the new term V_t has been added; it represents a stochastic process capturing the measurement errors.

The next step is to define a model for the OD flows. In [20], the authors considered a combination of a deterministic diurnal pattern and a zero-mean fluctuation process around it. However, the authors in [21] explained that this model fails to track the Traffic Matrix, due to changes of the traffic demands. That is because their model

neither can be calibrated for the large changes nor allows for little changee.

Conversely, in [21] they considered more flexible model for the OD flows as a combination of three terms:

$$x_t = \hat{x_t} + \delta_t + \eta_t$$

$\hat{x_t}$ is the predictable term, while δ_t is the random noise and the η_t is the 'innovation process' that is introduced to capture the unpredictable behavior of traffic. This term is explained in detail shortly.

Thinking of the OD flows as network states, the authors considered a linear dynamic system to model the TM:

$$x_{t+1} = C_t x_t + W_t$$

The diagonal elements of the state transition matrix C_t captures the temporal correlations within a single flow, while the off-diagonal elements capture spatial correlations across different flows. The whole system model can be written as:

$$\begin{cases} x_{t+1} & = C_t x_t + W_t \\ y_t & = A_t x_t + V_t \end{cases} \tag{3.21}$$

with state noise W_t and measurement noise V_t being uncorrelated zero-mean Gaussian white-noise with covariance matrices Q_t and R_t. In this equation y_t represents the link counts and A_t is the routing matrix. To find the optimal prediction of the network state for next time slot, x_{t+1}, given the past observations, $\{y_1, \ldots, y_t\}$, they follow two steps recursively, the prediction step and the update step.

In the first step, based on the linearity of the system, they predict the next time

slot Traffic Matrix, $\hat{x}_{t+1|t}$, based on the current state estimate of the network, $\hat{x}_{t|t}$. Also as the linear system is affected by noise, it is reasonable to determine the variance of the prediction, $P_{t+1|t}$, based on the updated variance of prediction and Q_t, noise covariance at time t:

$$\begin{cases} \hat{x}_{t+1|t} & = C_t \hat{x}_{t|t} \\ P_{t+1|t} & = C_t P_{t|t} C_t^T + Q_t \end{cases} \tag{3.22}$$

where $\hat{x}_{t+1|t}$ is the predicted TM for time $t+1$ based on the information available up to time t, $\hat{x}_{t|t}$ is the updated estimate of TM at time t, $P_{t+1|t}$ is the variance of prediction for time $t+1$ based on all information available up to time t, and $P_{t|t}$ is the updated variance of prediction. C_t represents the state transition matrix.

In the next step, as time advances and new observations arrive, y_{t+1}, they update the state and prediction variance. This update is based on the innovation process that is the difference between observed value and predicted value, that is:

$$\eta_t = A_{t+1} \hat{x}_{t+1|t} - y_{t+1}$$

where η_t is the innovation process, A represents the routing matrix, $\hat{x}_{t+1|t}$ is the predicted TM of time $t+1$ based on all information available up to time t, and y_{t+1} is the link loads observed at time $t+1$.

They update the estimate and the variance of prediction at time t+1 by following equation:

$$\begin{cases} \hat{x}_{t+1|t+1} = \hat{x}_{t+1|t} + K_{t+1}[y_{t+1} - A_{t+1}\hat{x}_{t+1|t}] \\ P_{t+1|t+1} = (I - K_{t+1}A_{t+1})P_{t+1|t}(I - K_{t+1}A_{t+1})^T + K_{t+1}R_{t+1}K_{t+1}^T \end{cases} \tag{3.23}$$

These updated values, are used in predicting the TM in the next time slot and the prediction variance.

In equation 3.22, K is the Kalman gain matrix. The Kalman filter guarantees that the optimal prediction will be found. The optimality here is defined as minimizing $E[||x_{t+1} - \hat{x}_{t+1}||^2]$. Therefore, the estimator can be categorized as a 'Minimum Variance Unbiased Error Estimator' (MVUE). This can be used to compute the Kalman gain matrix, K_{t+1}. First they compute the estimation error at time t, $\tilde{x}_{t|t} = \hat{x}_{t|t} - x_t$, and then try to minimize the conditional mean-squared estimation error, i.e. $E[\tilde{x}_{t+1|t+1}^T \tilde{x}_{t+1|t+1}|y^t]$.

To execute the Kalman filter we need the matrices A, C, Q and R. matrix A can be obtained from the routing scheme of the network. For the other matrices, the authors proposed a stationary situation in which we can drop the t subscript of these matrices. For obtaining the other matrices, they assume that Netflow is available in the network but expensive to turn on. Netflow is a protocol developed by Cisco System that collects information of the IP packets flowing through the network interfaces [24]. Interfaces where Netflow is enabled can gather the statistics of IP traffic, and send this statistical information to a central Netflow collector. NetFlow is designed to process all IP packets on an interface. On Internet backbones where a large number of simultaneous flows exist this statistic observation process can be costly, because of the processing required for each packet, and because of the overhead

of transferring the information to a central Netflow collector.

As a result, the authors in [21] use Netflow for only limited periods of time to get a set of Traffic Matrices. Having these matrices and implementing the EM algorithm, they calculate the C, Q, R matrices. Afterwards, they use these matrices in their Kalman filter equations based on the stationary assumption. The authors claimed that these matrices can be used for several days with a good accuracy, before the model degrades. However they defined a threshold for the time after which the estimator does not work very well. Once the accuracy degrades sufficiently, they can turn on the Netflow tool to get a new set of statistical data from the network to re-calibrate their matrices. To determine when the system needs calibration, they monitor the innovation process. The innovation process was previously defined as:

$$\eta_{t+1} = y_{t+1} - A_{t+1}\hat{x}_{t+1|t}$$

The variance of the innovation process can be determined as:

$$S_{t+1} = E[\eta_{t+1}\eta_{t+1}^T]$$

Under the Gaussian hypothesis for the noise W_t and V_t, the innovation process at each time slot should within an interval of $\pm 2\sqrt{S_T^{jj}}$ with the probability of more than 95%. The authors use this as the innovation process threshold, meaning that a re-calibration is needed when the innovation process exceeds this threshold, $||\eta_t^j|| \geq 2\sqrt{S_T^{jj}}$, for more than ten consecutive time slots.

They show that using calibration in their model helps achieve better tracking and estimation of the traffic matrix. In the following two images they illustrate how

calibration improves the accuracy. In figure 3.5 calibration is applied only once at the beginning of the experiment. In figure 3.6, calibration is allowed during the experiment too.

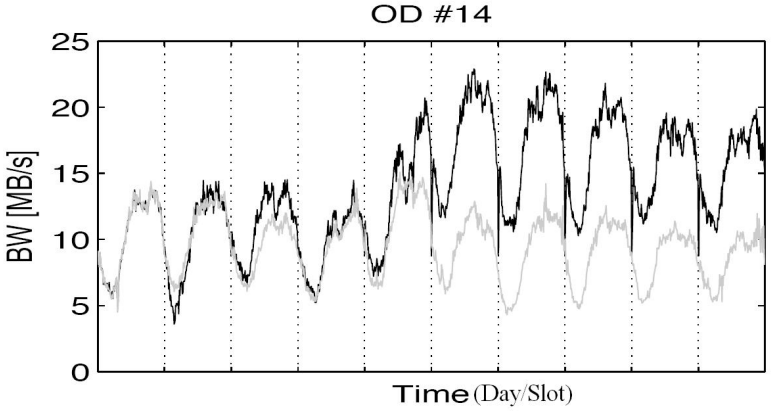

Figure 3.5: Applying Calibration Once, Real (light grey) and inferred OD (dark grey) flows, [21]

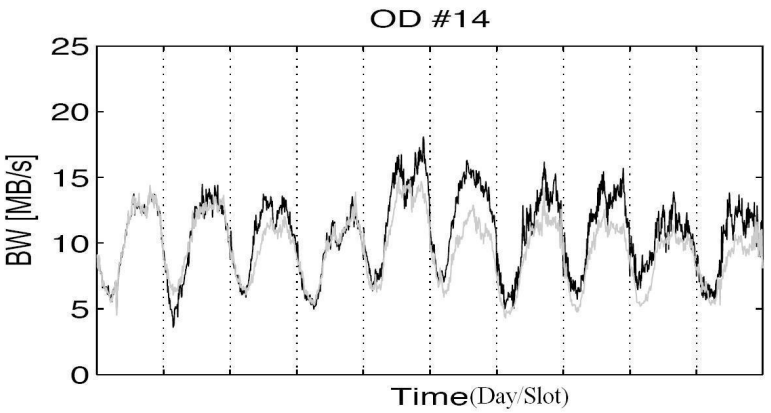

Figure 3.6: Multiple Calibration applied,Real (light grey) and inferred OD (dark grey) flows, [21]

As shown in figures 3.5, after day 5 the system needs to be re-calibrated for flow 14. However, re-calibration may degrade the tracking ability of system for other flows. As shown in figure 3.7, the system can track flow 9 very well, given one calibration applied at the beginning. However, if we recalibrate the system on day 5 then flow 9 cannot be tracked as before, as shown in figure 3.7. It appears difficult to track all flows with sufficient accuracy, as the calibration process to improve the tracking of one flow can degrade the tracking of another flow.

3.2 Recursive Flip-Flop Estimator Filters

The authors in [23] proposed an approach to combine two recursive rate-estimators to have a single estimator that can be both agile and stable over time. Agility

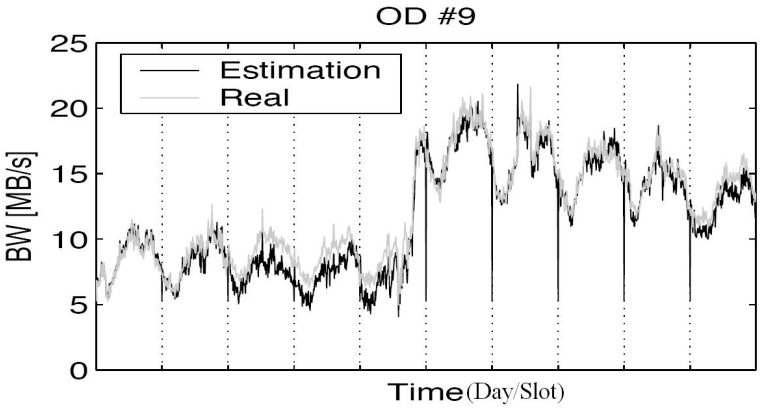

Figure 3.7: Applying Calibration Once, Real (light grey) and inferred OD (dark grey) flows, [21]

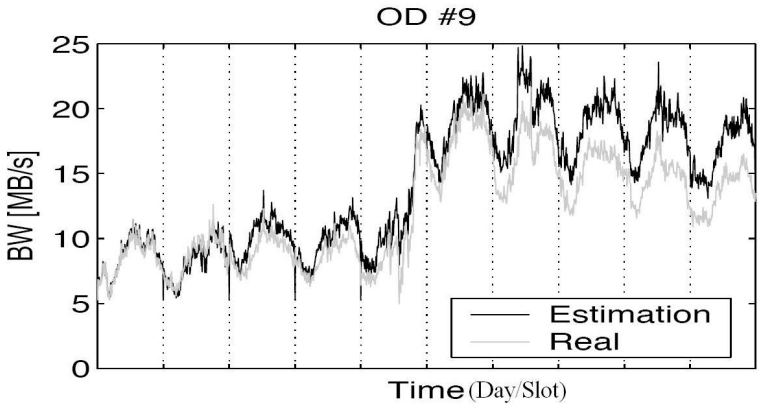

Figure 3.8: Multiple Calibration applied,Real (light grey) and inferred OD (dark grey) flows, [21]

is the ability to track the quick changes of the traffic rates in the network, while stability is the ability to ignore short term transient changes. They used Exponentially Weighted Moving Average (EWMA) and Time Sliding Window (TSW) filters. EWMA is an ideal maximum likelihood estimator while TSW is a rectangular data weighting method used in traffic prediction.

The traffic rate is defined by two metrics, the packet length, b and the inter-arrival time, τ, as $x = b/\tau$. In EWMA the memory is shaped exponentially while in TSW the memory is shaped with a simple rectangular window.

In EWMA, they first determine the difference between the contracted 'ideal' inter-arrival time (contracted in the Service Level Agreement or SLA) and the current observed inter-arrival time. If w denotes the allocated bandwidth, s the current packet size and t is the time between current and previous packet arrivals, then the difference between the contracted and the current inter-arrival times is $d = t - (s/w)$. They apply the EWMA filter to estimate the next average value for this difference based on the current average and last observed d:

$$d_{ave} = (1 - \omega)d_{ave} + \omega d \tag{3.24}$$

where ω is the EWMA time constant and d is the current difference. Equation 3.24 recursively updates the d_{ave} based on its previous value and current d. ω is the time constant of the EWMA. It determines how stable or agile the estimator is. The larger ω, the more stable the estimator will be.

The estimated rate can be determined as:

$$r_{est} = \frac{s}{\frac{s}{r} + d_{ave}} \tag{3.25}$$

where s is the packet size, r is the current rate and d_{ave} is calculated with equation 3.24.

In equation 3.25, an EWMA filter is used to estimate the next arrival time. However, it can be used to estimate the packet size too. They showed that the performance of this method in a system with two EWMA filters with same weight factors, where one filter estimated the packet size and the other filter estimated the inter-arrival times. In their simulation they called this approach as SE.

The other method they used is called TSW, the Time Sliding Window. In this approach, the Window Length, W_L, decides the agility or stability of the estimator. Again the larger W_L, the more stable the estimator will be. The concept of the W_L is shown in figure 3.9. Using the same notation, the estimated rate is defined as:

$$r_{est} = \frac{r_{est,old} \times W_L + s}{now - T + W_L} \tag{3.26}$$

W_L considered how mush information from the past we need to include. Obviously

the larger the W_L the more stable the estimates will be. The concept of T and W_L are shown in figure 3.9. s represents the packet size.

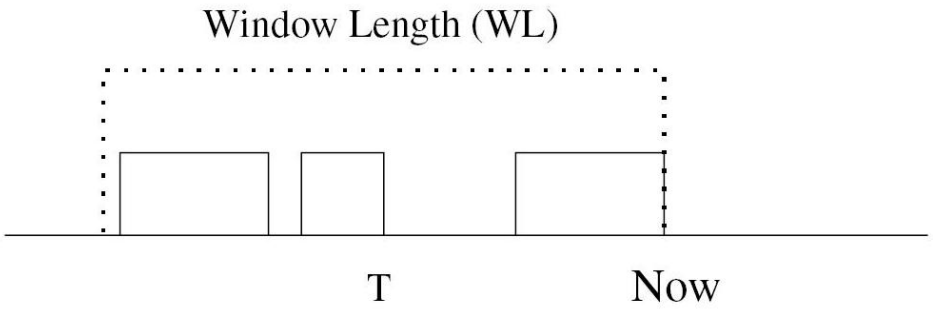

Figure 3.9: Window Length in TSW, [23]

By increasing the W_L, the TSW filter gets more stable. It is obvious as W_L goes to infinity, the estimated rate is always equal to the old estimated rate.

The main contribution of authors in [23] is creating a 'flip-flop' filter. They consider two TSW filters, one with a small W_L, the agile filter, and one with large W_L, the stable filter.

In the normal situation, the output of the stable filter is being used. However if the difference between the estimated rate of these two filters becomes more than a preset value (i.e., more than $C1$ times in a row), then there is a non-transient change in traffic rate that should be considered. At this point they use the output of agile filter for the next time slot. On the other hand, if the difference between output of these two filters becomes more than a preset value (i.e., more than $C2$ times in a row), then the change is persistent, and the algorithm sets the average rate of the stable filter to the average rate of the agile filter:

$$Ave\_Rate\_Stable = Ave\_Rate\_Agile$$

They denote this method in their simulations as the 'Flipflop' filter.

In figure 3.10 they compare their proposed Flipflop filter, with a single TSW filter and the SE filter that is composed of two EWMA filters. The synthetic traffic that is produced with OPNET's built-in traffic tools is shown in solid dotted line.

The two EWMA filters which build the SE filter have a same weight factor of 0.002. It seems the final SE filter performs better when traffic is not changing. However, when the traffic doubles in 30 minutes, the SE filter cannot follow the changes of the traffic.

The single TSW filter has a window length of 80 seconds. Because it is less stable, it works better in case of rapid traffic changes than the SE filter.

Their proposed filter consists of two TSW filters, one with window length of 15, and the other with window length of 80. The Flipflop filter is shown with solid line. The Flopflop filter has an even better performance when a dramatic change in traffic rates happens at 30 minutes. Before that change, however, the Flipflop uses the stable TSW filter with a window length of 80.

3.3 ARCH-Based Traffic Prediction

Time series analysis will be explained throughly in the next chapter where we explain the recently proposed model [37]. In this section we take a comprehensive look at some famous time series utilities, and their applications in previous works. ARCH filters have been in used in econometrics schemes for a long time, [4] [5].

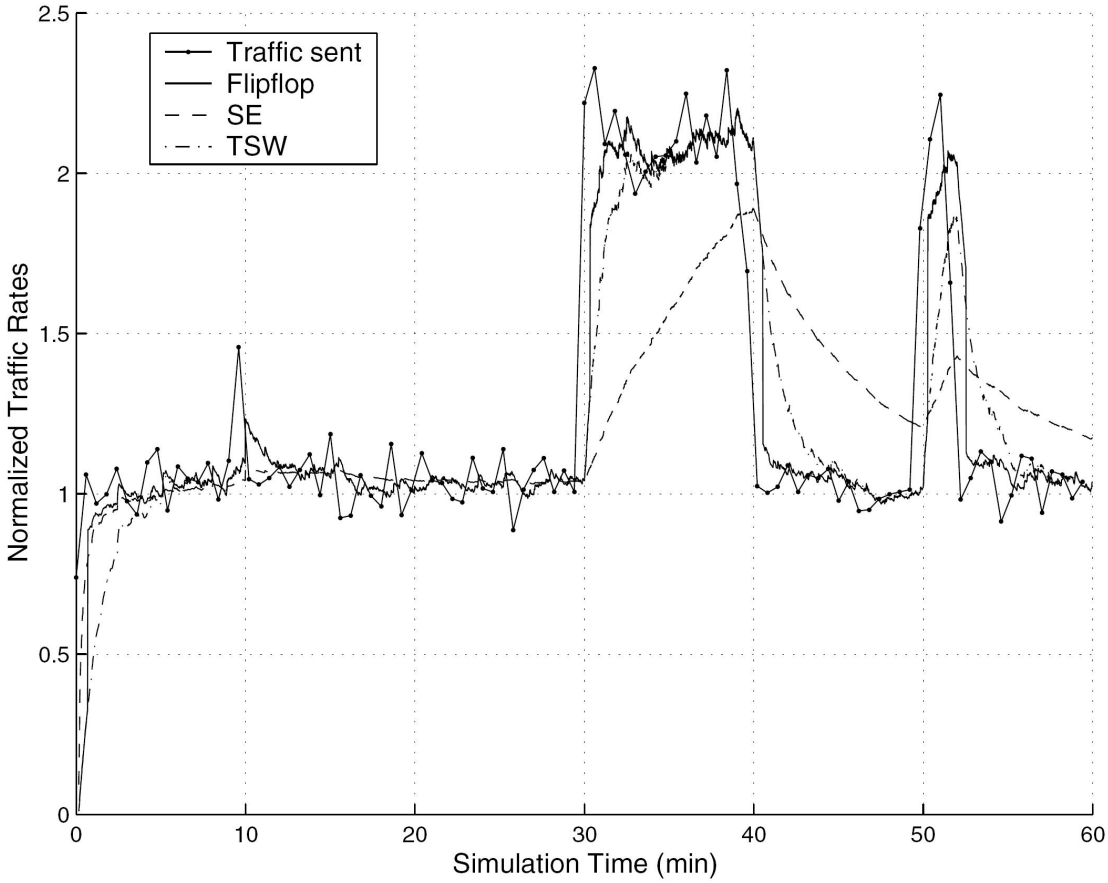

Figure 3.10: Comparing Flip-Flop filter and SE with Real Traffic, [23]

In general in a network with n nodes there are $N = n^2$ OD flows. These OD flows and link loads can be considered as time series. In Time Series analysis the goal is to characterize these flows with different mathematical models. Different approaches have been proposed. In [22] authors use seasonal Autoregressive Conditional Heteroskedasticity (ARCH) to characterize these flows for forecasting.

ARCH can be viewed as an extended form of ARIMA filters. The general form of

ARIMA filters is:

$$x_t = \sum_{i=1}^{p+d} \varphi_i x_{t-i} + \sum_{k=1}^{q} \theta_k \varepsilon_{t-k} + \varepsilon_t$$

with $\{\varphi_i\}_{i=1}^{p+d}$ and $\{\theta_k\}_{k=1}^{q}$ being the autoregressive and moving average parameters, respectively. Here the $p+d$ and q are autoregressive and moving average orders, respectively. ε_t is considered as a random variable and is called the innovation.

The main difference between ARIMA and ARCH is how to model the innovation. In ARIMA the ε_t is considered as independent and identically distributed normal random variables with mean 0 and variance σ^2 [22]. In an ARIMA filter the innovation variance is independent or $E[\varepsilon_t^2|F_{t-1}] = \sigma^2$; where F_{t-1} includes all information available at time $t-1$. In an ARIMA filter the variance of the innovation doesn't change over time.

In ARCH however, the innovation is considered dependent to the past information, so the variance of the innovation changes over time. This can be shown mathematically as $E[\varepsilon_t^2|F_{t-1}] = \sigma_t^2$. ARCH is a utility that models the changing variance.

To better understand these models, we can start by looking at ARCH(1). In [2], the authors write the innovation part of a ARCH with order 1 as:

$$\varepsilon_t = \sigma_t \eta_t$$
$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2$$

where σ_t^2 is the variance of the innovation, σ_t is the standard deviation of the innovation, η_t is a sequence of independent and identically distributed (iid) random variables with zero mean and unit variance and $\alpha_0 > 0$ and $\alpha_1 \geq 0$ should be chosen in a way to ensure that the unconditional variance of ε_t is finite.

η_t can have a standard normal or a standardized Student-t distribution or a generalized error distribution [2]. The General form of innovation in a ARCH filter with order of m can be written as:

$$\begin{aligned}\varepsilon_t &= \sigma_t \eta_t \\ \sigma_t^2 &= \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2 + \cdots + \alpha_m \varepsilon_{t-m}^2\end{aligned}$$

where ε_t represents the innovation at time t, σ_t^2 is the variance of the innovation and η_t represents a sequence of independent and identically distributed (iid) random variables.

The authors in [22] extend the ARIMA model by rewriting the innovation, ε, as:

$$\begin{aligned}x_t &= \sum_{i=1}^{p+d} \varphi_i x_{t-i} + \sum_{k=1}^{q} \theta_k \varepsilon_{t-k} + \varepsilon_t \\ \varepsilon_t &= \eta_t \sigma_t \\ \sigma_t^2 &= \alpha_0 + \sum_{k=1}^{m} \alpha_k \varepsilon_{t-k}^2\end{aligned} \tag{3.27}$$

By adding $m+1$ parameters, $\{\alpha_i\}_{i=0}^{m}$, to the original ARIMA model, the innovation becomes dependent to the past information. In fact, $E[\varepsilon_t^2 \varepsilon_{t-1}^2] \neq 0$, while $E[\varepsilon_t \varepsilon_{t-1}] = 0$, [22].

They use the data collected in the period of 10 days, shown in figure 3.11, to characterize traffic volume. The authors notice the periodic effect that arises at different times in different days. They extend their model by adding the additive or multiplicative seasonal components to take these periodic effect into account. Assuming

the period of these effects is T then:

$$x_t = \sum_{i=1}^{p+d} \varphi_i x_{t-i} + \sum_{j=1}^{s} \phi_j z_{t-jT} + \sum_{k=1}^{q} \theta_k \varepsilon_{t-k} + \varepsilon_t$$

where T is the period of these periodic effects, $\{\varphi_j\}_{j=1}^{s}$ are additive seasonal autoregressive parameters and s is the seasonal autoregressive order.

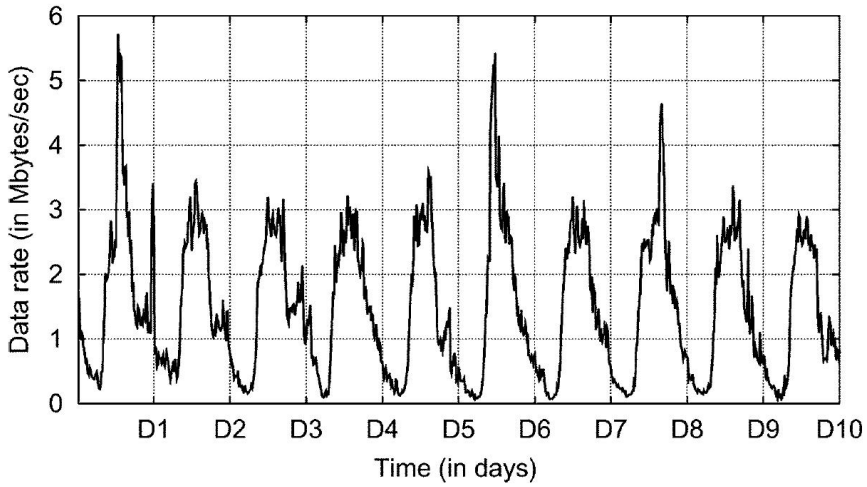

Figure 3.11: 10 Days Traffic Volume, [22]

As explained earlier, in an ARCH filter the variance of the innovation is not constant over time. Innovation can be written as a multiplication of a random variable, η_t and the standard deviation of the innovation, σ_t. Authors in [22], claim that the conditional distribution of innovation process is not normal; in fact it follows a more general distribution. They claim that a heavy-tailed distribution should be considered to accommodate the innovation process. Therefore they consider student's t-distribution for the η_t.

The Student's t-distribution has Probability Distribution Function (PDF) similar

to the PDF of the Normal distribution. The main difference between these two distributions, however, is that the Student's t-distribution has a thicker tail. Authors in [22] use this distribution as they claim the network traffic shows characteristics of heavy-tail processes. In figure 3.12 we used MATLAB to generate the PDF of these two distribution for comparison. As the degree of freedom in the Student's t-distribution increases, it approaches the Normal distribution. In fact the Normal distribution is queivalent to a Student's t-distribution with degree of freedom of infinity.

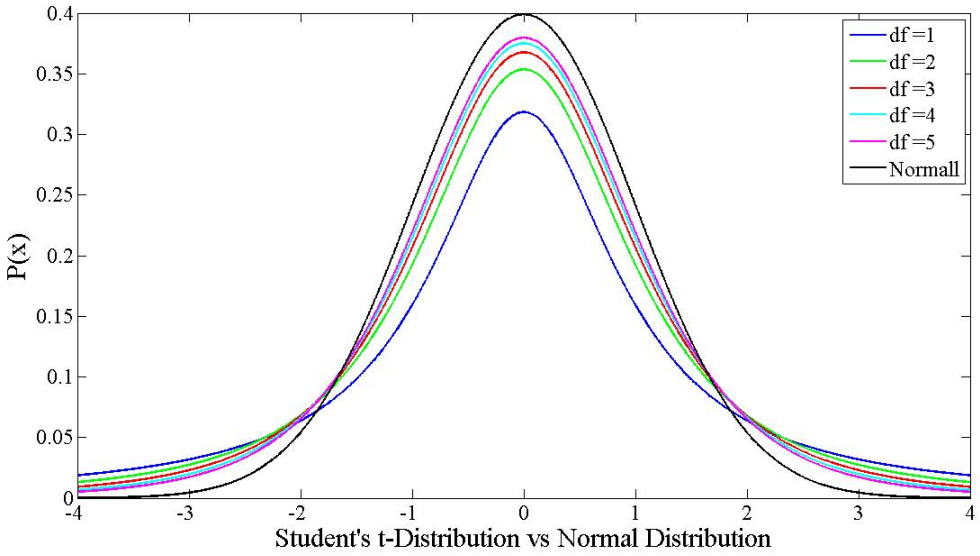

Figure 3.12: PDF of Student's t-Distribution with different Degrees of Freedom vs Normal Distribution

As shown in figure 3.12, the Student's t-distribution has a heavy tail in comparison with the Normal distribution.

The authors in [22] considered the Student's t-distribution instead of Normal Distribution. The model can be summarized as:

$$
\begin{aligned}
x_t &= \sum_{i=1}^{p+d} \varphi_i x_{t-i} + \sum_{j=1}^{s} \phi_j x_{t-jT} + \sum_{k=1}^{q} \theta_k \varepsilon_{t-k} + \varepsilon_t \\
\varepsilon_t &= \eta_t \sqrt{M_t} \\
M_t &= \sigma_t(\frac{\nu - 2}{\nu}) \\
\sigma_t^2 &= \alpha_0 + \sum_{k=1}^{m} \alpha_k \varepsilon_{t-k}^2
\end{aligned}
\tag{3.28}
$$

As we had in previous equations, x_t represents an observable random variable, i.e., the traffic demands. $\{p + d\}$ and q are the autoregressive and moving average orders, respectively. s is the seasonal autoregressive order, with T being the period of those seasonal effects. ε_t represents the innovation, with σ^2 being its variance. $\{\alpha\}_0^m$ are coefficients, m is the ARCH order. η_t is Student-t distribution random variable with unit scale, and with ν degrees of freedom. M_t is the scale parameter.

It is observable in figure 3.11 that there are some peak values, for example in days 1, 6 and 8. These peaks can distort the innovation distribution and statistical estimate of the parameters. To smooth these effects, the authors perform a natural logarithmic transformation to stabilize the data points. The transformation can be written as $w_t = \ln x_t$.

The new series, shown in figure 3.13, is clearly more stable in comparison with the original network traffic shown in 3.11.

Since there is a slow decay of the autocorrelation at integer multiples of T of the series w in figure 3.13, the series cannot be considered as stationary. In the last step

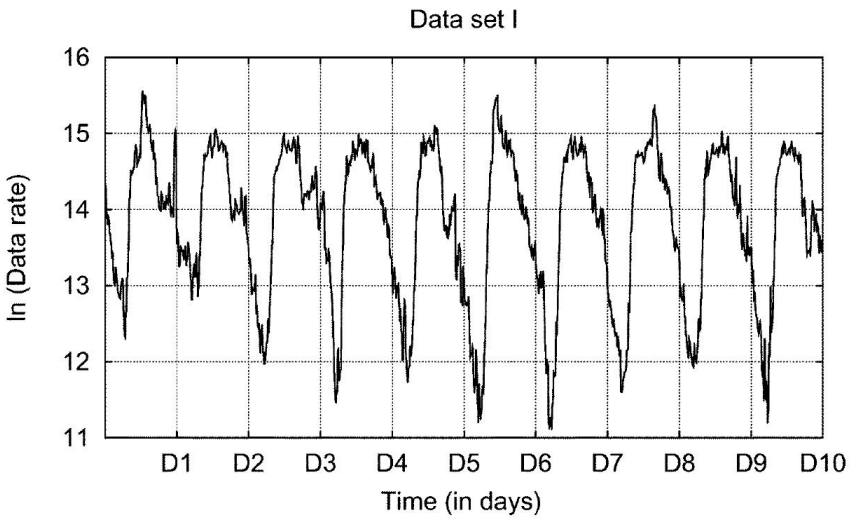

Figure 3.13: Log Transformed Data, [22]

they perform a subtraction to get a stationary series s:

$$s_t = (w_t - w_{t-1}) - (w_{t-T} - w_{t-T-1})$$

where T, as explained before, is the seasonal lag. The obtained time series s_t is shown in figure 3.14. As shown in this figure, the mean of series s_t seems to be stationary.

To identify the statistical model of s_t they used the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and likelihood ratio test. They found values for p, s, q and m to be 4, 2, 0 and 1, respectively. With this, the unknown

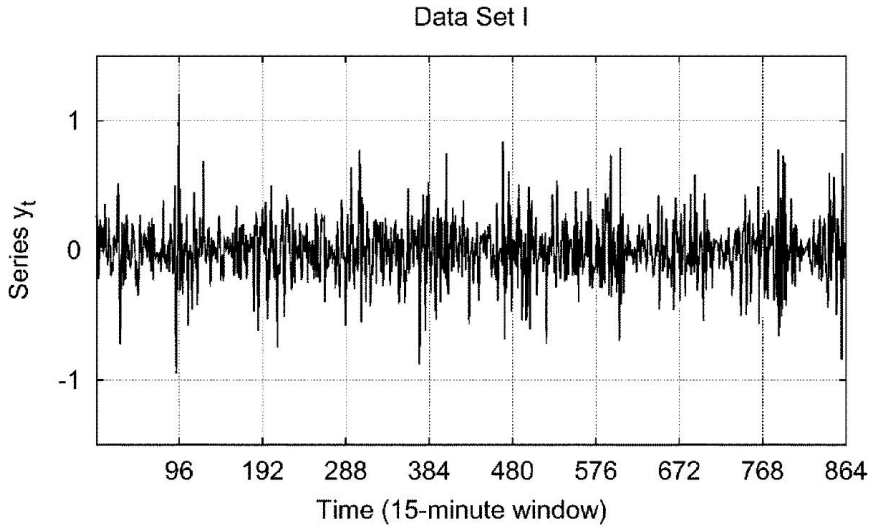

Figure 3.14: First-Differenced log Transformed Data, [22]

parameters will be $\{\varphi_i\}_{i=1}^4$, $\{\phi_j\}_{j=1}^2$, α_0, α_1 and ν in the equation 3.29.

$$\begin{aligned} s_t &= \sum_{i=1}^{4} \varphi_i s_{t-i} + \sum_{j=1}^{2} \phi_j s_{t-jT} + \varepsilon_t \\ \varepsilon_t &= \eta_t \sqrt{M_t} \\ M_t &= \sigma_t(\frac{\nu-2}{\nu}) \\ \sigma_t^2 &= \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 \end{aligned} \tag{3.29}$$

In equation 3.29, the autoregressive order is 4, the seasonal order is 2, the order of moving average is 0. Also the ARCH order, m, is 1. Therefore they need to estimate the autoregressive parameters $\{\varphi_i\}_{i=1}^4$, seasonal autoregressive parameters $\{\phi_j\}_{j=1}^2$, the degree of freedom of the student-t distribution ν, and finally α_0, α_1.

These unknown parameters can be estimated based on the sample maximum likelihood function for the first 10 days of observations, to identify the statistical model

for s_t. For computing the forecast of x_t based on the time series model of s_t, authors proposed two methods, the MMSE forecast computation and Probability-Hop forecasting method. Here we describe how they compute the MMSE forecast.

In Minimum Mean Square Error forecast method, it is known that the forecast is given by the conditional expectation $E[s_t|F_{t-1}]$. Recalling that $w_i = \ln x_t$ and $s_t = (w_t - w_{t-1}) - (w_{t-T} - w_{t-T-1})$ we can write:

$$w_t = \ln x_t = \sum_{i=1}^{5} K_i \ln x_{t-i} + \sum_{j=T}^{T+5} K_j \ln x_{t-j} + \sum_{k=2T}^{2T+1} K_k \ln x_{t-k} + \sum_{s=3T}^{3T+1} K_s \ln x_{t-s} + \varepsilon_t$$

where K's are constants that can be computed from the estimates of φ and ϕ. The target is to compute the forecast of $E[x_t|F_{t-1}]$. They approximated the expected value of the one-step ahead forecast, $E[x_t|F_{t-1}]$, using equation 3.30:

$$\ln(\frac{x_t}{x_{t-1}}) \simeq (\frac{x_t}{x_{t-1}} - 1) \tag{3.30}$$

In the second method, they explain the Probability-Hop Forecasting method, which involves considering the probability limits. Here we follow the exact notation of the authors. If $\hat{x}_{t-1}$ is the one-step forecast of x_{t-1} at time $t-1$, then they consider the 3.31 as the confidence limit for the forecast:

$$\hat{x}_{t-1}^{\pm} = \hat{x}_{t-1} \pm t_{\nu,\frac{\beta}{2}} \sqrt{M_t} x_{t-1} \tag{3.31}$$

where $t_{\nu,\frac{\beta}{2}}$ is the deviate from the student-t distribution with ν degrees of freedom corresponding to the $100(1-\frac{\beta}{2})$ probability limit. In this equation, $\hat{x}_{t-1}$ represents the one-step exact forecast of the x_{t-1}. As an example, they considered $\beta = 0.2$ that

corresponds to the 90% probability limit.

Based on which sign is being used there are two curves, the Upper Probability Curve (UPC) and the Lower Probability Curve (LPC), corresponding to the 90% probability limits. The term $\hat{x}_{t-1}$ is the Exact Forecast Curve (EFC) that is the MMSE forecast curve. All these curves along with the real data itself are shown in figure 3.15.

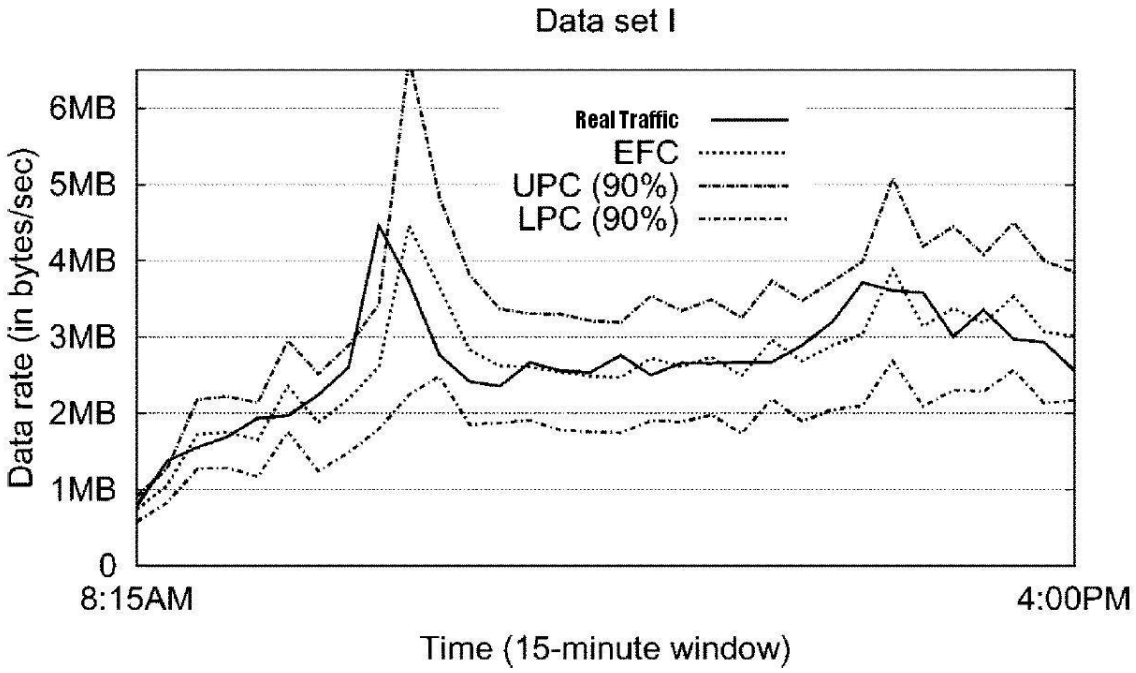

Figure 3.15: Real traffic of part of a day , along with EFC , UPC and LPC, [22]

The authors in [22] used 10 days to build their model. In figure 3.15, they show how their model can track the traffic in day 11. In this figure the observed traffic rates have been found to stay within the 90% probability limits UPC and LPC most of the time [22]. However, they only showed 8 hours of their simulation.

3.4 GARCH-Based Traffic Prediction

In [25] the authors proposed another approach to model the Traffic Matrix as time series. In [22] they used a specific model for the innovation process, while the authors in [25] considered the generalized form of the ARCH to take the innovation process changes into account. They describe their model to capture the variance of the innovation over time. For smoothing the Traffic Matrix, they first performed a log-transformation similar to [22] $w_i(t) = \ln(x_i(t))$. This makes the traffic volumes more stable. The auto-correlation function showed that the obtained series is not yet stationary, so they went further and defined a function to make a stationary trend, slightly different from what we had before in [22]. We use the same notation and call the obtained series as s:

$$s_i(t) = w_i(t) - w_i(t-1) \tag{3.32}$$

where s and w are the obtained stationary series and the log-transformed series of OD flow i, respectively.

The authors in [2] describes the general form of the GARCH. In GARCH, similar to ARCH, the conditional variance of the innovation is considered variable, and dependent to the past information. In ARCH, however, a specific model is considered for the variance, but in the GARCH a more general form is being used. The innovation can be described as:

$$\begin{aligned} \varepsilon_t &= \eta_t \sigma_t \\ \sigma_t^2 &= \alpha_0 + \sum_{i=1}^{m} \alpha_i \varepsilon_{t-k}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2 \end{aligned} \tag{3.33}$$

where the innovation ε_t is considered as a random variable with zero mean and

conditional variance of σ_t^2, η_t is a sequence of independent and identically distributed (iid) random variables. η_t can be considered as a normally distributed random variable or as an alternative, the Student-t distribution can be used to model it. As shown in equation 3.33, the conditional variance σ_t^2 evolves over time.

Because past values of the σ_t process are fed back into the present value, the conditional standard deviation can exhibit more persistent periods of high or low volatility than seen in an ARCH process.

In [25] authors used GARCH to characterize the traffic. They used a slightly different notation. The general form of the GARCH can be written as 3.34. The main difference with the ARCH is the definition of the variance of the innovation process:

$$
\begin{aligned}
& s_t = \sum_{k=1}^{p} \varphi_k s_{t-k} + \sum_{h=1}^{q} \phi_h \varepsilon_{t-h} + \varepsilon_t \\
& \varepsilon_t = \eta_t \sigma_t \\
& \sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i \varepsilon_{t-k}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2 \\
& \sum_{i=1}^{m} \alpha_j + \sum_{j=1}^{s} \beta_s < 1 \\
& \sigma_{\varepsilon_i} > 0, \;\; \alpha_{i,j} \geq 0, \;\; \beta_{i,s} \geq 0,
\end{aligned}
\tag{3.34}
$$

where series s is the obtained stationary series, defined in equation 3.32. $\varphi_k, \phi_h, \alpha_i$ and β_j are the factors to be estimated. In equation 3.34, ε_t is the random variable representing the innovation with σ_t^2 being its variance. It shows that the innovation process is a zero mean random variable with a variance that evolves over time. As explained before, η_t is a sequence of independent and identically distributed (iid)

random variables, with two possible distributions, either the Normal distribution or Student-t distribution.

Similar to [22], they ran the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to determine p, q, m and s. They assume a Student-t distribution for the $\varepsilon_i(t)$.

The series s was obtained from equation 3.32, and w is the log transformation of x. Therefore by using equation 3.30, they approximate the final model for the network traffic as:

$$\hat{x}_t \approx x_{t-1}(1 + \sum_{k=1}^{r+1} h_k \ln x_{t-k})$$

where $\hat{x}_t$ is the final estimate of the network traffic, and h_k's can be computed based on the estimated values of φ_k and ϕ_h. They decided to compare their work with Tomogravity and the 1-Inverse methods of Traffic Matrix estimation. For comparison they use four-day real data/ The first three days of real data is used for building the model, and the last day of data is used to test their model. Figure 3.16 illustrates the results of their model along with the real traffic and with other approaches.

The real traffic has been measured every 5 minutes. In figure 3.16, the GARCH result is shown in grey. We can observe that it has more ability to track the real traffic matrix, in comparison with 1-Inverse and Tomogravity models which are shown in purple and pale black respectively.

To mathematically evaluate the model, they defined Spatial Relative Errors as:

$$err_{sp}(n) = \frac{||\hat{x}_T(n) - x_T(n)||_2}{||x_T(n)||_2}$$
$$n = 0, 1, \ldots, N;\ t = 1, 2, \ldots, T;$$

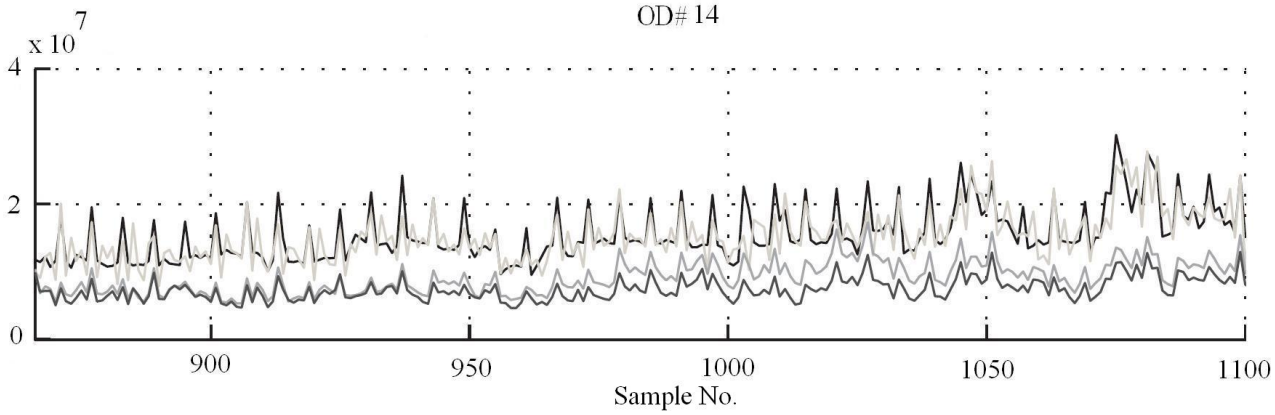

Figure 3.16: Comparing Garch (gray) , 1-inverse (dotted) , Tomogravity (pale black) and Real traffic (black), [25]

where $||.||$ stands for the L_2 norm, N is the number of nodes and T is the number of measurements. They compared the Cumulative Distribution Function (CDF) of these three methods in figure 3.17.

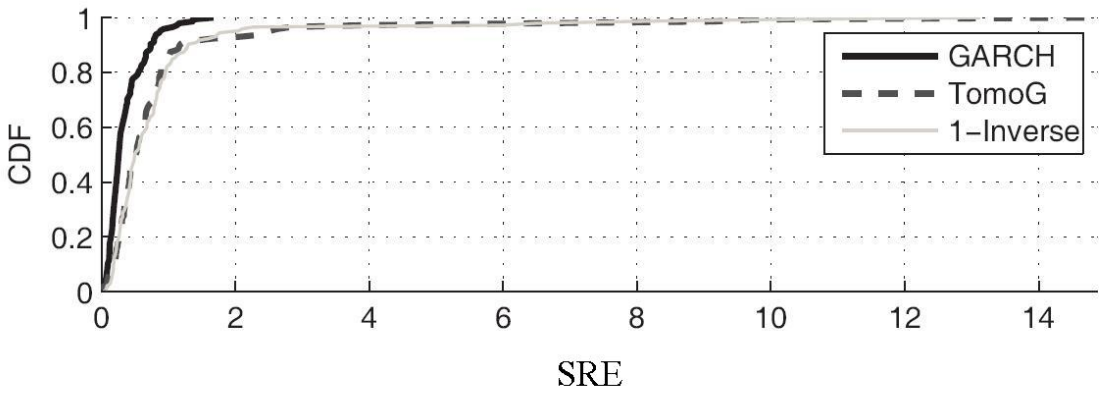

Figure 3.17: CDF of Spatial Relative Error of three methods, [25]

Basically figure 3.17 demonstrates the probability of having a spatial relative error less than or equal to a specific value. For example, in [25] they declare that based on

this figure, 92.8% of the OD flows can be tracked with SREs less than 0.8, while this number for the Tomogravity and 1-Inverse model is 72% and 69%, respectively.

3.5 ARIMA-Based Traffic Prediction

We will discuss ARIMA model in detail in the next chapter where we describe the recently proposed prediction and provisioning algorithm. In [26] the authors use the ARIMA model to capture the traffic volume in order to forecast the Traffic Matrix for the next time slot. An ARIMA Filter can be thought of as a transformation tool with a general extended form of:

$$\begin{aligned} x_t = \ & \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \cdots + \varphi_{p+d} x_{t-p-d} \\ & \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \end{aligned} \tag{3.35}$$

where x_t is random variable representing network traffic in our work. ε_t as explained in details in the previous sections, represents the innovation process. φ's and θ's are the autoregressive and moving average coefficients, respectively.

To gain a stationary series they generate $w_t = x_t - x_{t-1}$, (d would be equal to 1 for w). They use the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to determine the order of auto-regressive and moving average. They consider two values for autoregressive order, $p = 1$ & 2. They also found $q = 1$ would be the best for the moving average order.

Autoregressive and moving average parameters are estimated by a maximum likelihood approach.

To forecast the traffic volume, x_t for the next l step ahead they follow a recursive

approach. Being at time t, for predicting traffic at time $t+l$ we first need to have a prediction of all traffic before that time. Based on the equation 3.35 the forecast of the traffic in l step ahead would be:

$$\begin{aligned}\hat{x}_t(l) =&\varphi_1[x_{t+l-1}] + \cdots + \varphi_{p+d}[x_{t+l-p-d}] + \varepsilon[t+l] \\ &- \theta_1[\varepsilon_{t+l-1}] - \cdots - \theta_q[\varepsilon_{t+l-q}]\end{aligned} \tag{3.36}$$

where x's are either the already observed real values or predicted values, and ε's are either measured real values or 0, i.e. assigning 0 to unknown ε's. As time advances, the unknown ε can be computed by:

$$\varepsilon_{t+1} = x_{t+1} - \hat{x}_t(1)$$

where x_{t+1} is the real measured traffic at time $t+1$, and $\hat{x}_t(1)$ is its predicted value at time t.

Also to have a better prediction, they recursively update the estimates until the real data can be measured:

$$\hat{x}_{t+1}(l) = \hat{x}_t(l+1) + \Psi_l \varepsilon_{t+1}$$

Ψ's are the parameters computed based on φ's and θ's:

$$\begin{aligned}&\Psi_0 = 1; \\ &\Psi_j = \varphi_1 \Psi_{j-1} + \cdots + \varphi_{p+d}\Psi_{j-p-d} - \theta_j\end{aligned}$$

The calculated estimation of the traffic in equation 3.36 can be considered as the

Minimum Mean Square Error (MMSE) for the exact forecast. The authors in [26] take this as the Exact Forecast Curve (EFC). To have a boundary for their prediction they used a method called Probability-Hop Forecasting method. They explain that by considering ε's to be normally distributed, the conditional probability distribution of the future value x_{t+l}, $p(x_{t+1}|x_t, x_{t-1}, \dots)$, will be normal with the mean $\hat{x}_t(l)$ and standard deviation of $1 + (\sum_{j=1}^{l-1} \Psi_j^2)^{1/2}\sigma_\varepsilon$; where l is the number of step forecasts ahead. σ_ε is the variance of ε_t. Therefore the two curves (The Upper and Lower Probability Curves) representing the probability limits of the forecast with desired level of probability h can be written as:

$$x_{t+1}^{\pm} = \hat{x}_t(l) \pm u_{h/2}(1 + \sum_{j=1}^{l-1} \Psi_J^2)^{1/2}\sigma_\varepsilon \tag{3.37}$$

where $u_{h/2}$ is the deviate exceeded by a portion $h/2$ of the standard normal distribution. The positive and negative signs determine the Upper Probability Curve (UPC) and Lower Probability Curve (LPC), respectively. $\hat{x}_t(l)$ is the exact prediction of $t + l$ when we are at time t. In their work, however, they considered $l = 1$, i.e., they are only interested in predicting the next time slot traffic estimate.

In Figure 3.18 they show the results of their simulations for a probability limit of 50%. The total time of the experiment is 18 hours, and they used the first 8 hours of the observations to do the time series model fitting. The next 10 hoursof observations is split into 4 equal parts of 2.5 hours each. Figure 3.18 shows the real traffic demands along with the EFC, UPC and LPC curves for the second 2.5-hour period. The samples are taken every 5 minutes. Therefore there are 30 samples in a 2.5-hour period of the experiment.

In figure 3.18, we can see that the model can track the traffic narrowly often with

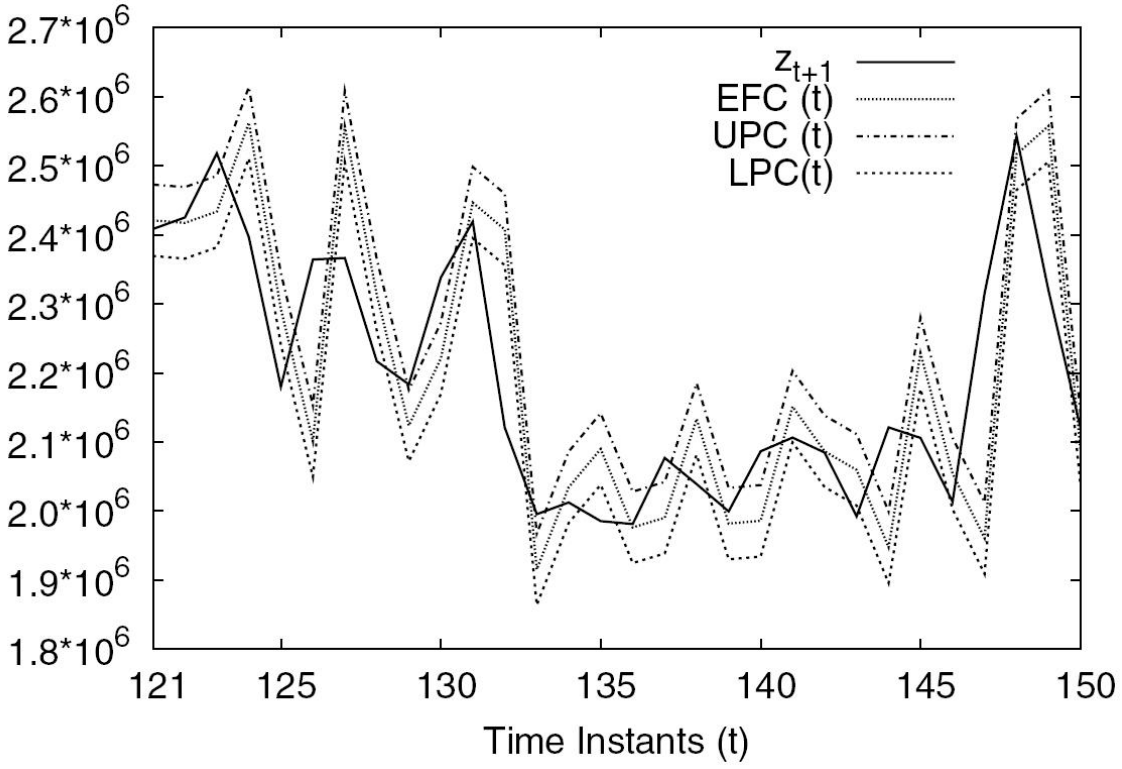

Figure 3.18: UPC , LPC , Exact Forecast Curve along with the Real Data, [26]

a lag time. However it fails to follow the traffic when some sudden changes happen in the system. Especially around time instance 125 this lack of traceability is more obvious.

3.6 Traffic Prediction Application

Todays 'Best-Effort' INTERNET is based on the *Best-Effort* delivery approach. However, Best-Effort delivery has relatively poor performance. Therefore, a significant Over-Provisioning of bandwidth is used in order to achieve better QoS guarantees

and to reduce the congestion o f the Best-Effort Internet[27]. Poor resource utilization and poor energy efficiency are the results of significant over-provisioning, and they establish the ineffectiveness of the *Best-Effort* INTERNET platform. The poor utilization of the available resources and bandwidth in the Best-Effort Internet cost Internet Service Providers excess capital costs annually.

To address these problems, the research community is exploring new network architectures called the *Future Internet Network* [30] [31] [32]. Recently a new framework for *Autonomic Future INTERNET* has been proposed that supports multiple service classes; the existing *Best-Effort* service class and a newly proposed *Essentially-Perfect QoS* service class [29]. The traffic flows in the Future Internet can be split to two different classes. The traffic flows categorized into the new QoS service class will never experience delays or congestion. These flows will receive Essentially-Perfect end-to-end QoS guarantees with negligible queuing delays within the routers.

The design of the existing *Best-Effort* routers need to change only incrementally to support the new QoS servuce class. In [29], the author shows how to alter the router design to achieve the propose dFuture Internet network. Figure 3.19 illustrates the router design.

Figure 3.19 shows an $M \times M$ Input-Queued switch used in a Best-Effort Internet router. A DeMux logic block is for directing the traffic to the appropriate VOQ buffer. A Best-Effort Scheduler is used to schedule each VOQ buffer for service at each time slot. A heuristic BE scheduler can achieve low efficiency and as a result Best-Effort Internet routers need very large buffer sizes, which contributes to excessive queuing delays and excessive energy consumption.

On the other hand, when a new QoS service class is added to the Internet routers,

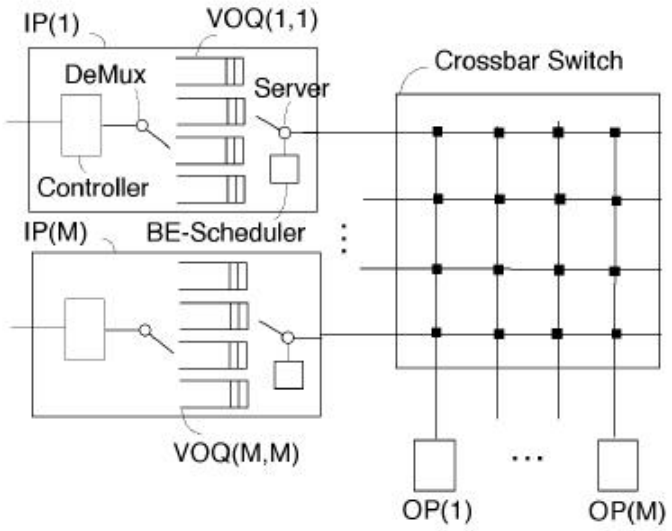

Figure 3.19: Current Best Effort-Router design. [29]

one goal is to provision enough bandwidth for this QoS class. In [29], the author proposed a new Future-Internet router design to support both the BE class and the new QoS service class. The existing VOQs in the BE router shown in 3.19 are logically partitioned into the two service classes, the QoS and BE classes. Incoming traffic can be directed to the appropriate class, BE or QoS, by examining the packet headers or the traffic flow's traffic specification.

3.7 Conclusion of Chapter 3

In this chapter different methods of the Traffic Matrix prediction have been reviewed. First, the use of the famous Kalman filter for predicting the future Traffic Matrix was explored. To predict the future Traffic Matrix given the past Link Loads observations, the authors in [21] first built a state-space model for the Traffic Matrix and Link Loads. To have an optimal prediction of the Traffic Matrix, they follow the prediction and update steps, recursively. In the prediction step, they use the updated TM to predict the future TM. In the update step, based on the new observation, they update the current TM for the future predictions.

The explanation of two recursive filters came next. The authors in [23] took advantage of two EWMA and TSW filters to design an estimator that can be adjusted to be both agile and stable. The main contribution in that paper was the design of a 'flip-flop' filter, containing two TSW filters. One of those filters is agile and the other is stable. The system can switch between these two based on the prediction accuracy.

In [22] the authors characterized the OD flows using ARCH filters. They used ARCH to consider the dependency of innovation term to the past information. The innovation itself is considered as a function of the past innovations. The authors in [22] picked the Student-t distribution for the innovation process. Another way of modelling the OD traffic flows is discussed in [25]. This method is a generalized form of ARCH and is called GARCH. In this method the variance of the innovation process varies over time.

In [26] the authors used ARIMA filters to model the traffic behavior. In ARIMA filters, the innovation process is considered as an independent and identically distributed random variable with mean 0 and standard deviation of σ.

In the last section, the application of traffic prediction in a network was reviewed. The prediction method can be used in a recently proposed Future-Internet router structure [29], to provide enough bandwidth to satisfy the future QoS demands.

Chapter 4

Model Description

In this chapter we will describe the model to forecast and predict the future traffic demands in a network. The algorithm can be used to predict traffic demands in a Future Internet network that supports 2 traffic classes, a Best-Effort class and a QoS class. An Autonomic Controller can use the predicted traffic demands for the QoS traffic class to configure each router, to provide enough bandwidth on each link for the future QoS traffic.

Here we first introduce an ARIMA filter that is the basis of the model. The novel idea of the model is to build multiple ARIMA-like filters which operate in parallel, and at different time-scales, to fully capture the traffic history.

4.1 ARIMA Filters and Traffic Cycles

Traffic prediction has been an interesting area of research. Different models and approaches have been proposed to forecast traffic. However many of these approaches

do not consider the cyclical behavior of traffic. On the other hand it has been observed that traffic demands show properties of self-similarity over different time scales [9]. These long range dependencies inherent in self-similar processes can be used for further prediction and traffic forecasting.

The data rate in a network is influenced by many different factors such as routing policy, user usage patterns, and even management related packets flowing in a network. The underlying nature of network traffic is stochastic. AutoRegressive Integrated Moving Average (ARIMA) filters can be used as a powerful utility to model the stochastic traffic in a network.

A sequence of observed data that are ordered in time is called a time series. The main characteristic of time series is the importance of the order of observations [1]. While in other statistical analyses the order of observed data is not crucial, in time series analysis the order in which the observations are made is explicitly recognized. Another difference between time series analysis and other statistical analysis is that in time series analysis, the observations are considered as dependent [3].

In time series analysis the collected samples are seen as a realization of a stochastic process. The stochastic process properties can be captured using ARIMA filters. As stated in [26], ARIMA filters have the general form of:

$$\Phi(B) \bigtriangledown^d x_t = \theta(B)\epsilon_t \tag{4.38}$$

ϵ represents shock (innovation) to the system at different steps. These shocks or innovations are considered as Independent and Identically Distributed (IID) random variables.

In equation 4.38, $\Phi(B)$ is the autoregressive operator of order p with the form:

$$\Phi(B) = 1 - \Phi_1 B - \cdots - \Phi_p B^p \tag{4.39}$$

In this equation, B is the backward shift operator, i.e. $Bx_t = x_{t-1}$ to include the earlier time series samples. Also $\bigtriangledown^d$ is difference operator of order d such that $\bigtriangledown^d = (1-B)^d$, with the role of making the series stationary. On the other hand, function $\theta(B)$ is the moving average operator of order q:

$$\theta(B) = 1 - \theta_1 B - \cdots - \theta_q B^q \tag{4.40}$$

This filter can be considered as a transformation function that transforms the stochastic process into a sequence of uncorrelated random variables.

The linear format of equation 4.38 can be written as:

$$\begin{aligned} x_t =& \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \varphi_3 x_{t-3} + \cdots + \\ & \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \cdots - \theta_q \epsilon_{t-q} \\ & or \\ & x_t = \sum_{i=1}^{p+d} \varphi_i x_{t-i} + \sum_{k=1}^{q} \theta_k \epsilon_{t-k} + \epsilon_t \end{aligned} \tag{4.41}$$

where $\varphi(B) = \Phi(B)\bigtriangledown^d$ is the stationary autoregressive operator:

$$\varphi(B) = 1 - \varphi_1 B - \cdots - \varphi_{p+d} B^{p+d}$$

and $\theta(B)$ is the moving average operator of order q, defined in 4.40.

In both types of equations 4.41 and 4.38, the value of the series at time t, x_t is composed of two parts; the prediction and the error part. The error part here is the ϵ_t and the other part is the prediction part. Except for the error value, ϵ_t, all other values can be observed and calculated. Traffic prediction can be accomplished based on these values; i.e., after observing traffic rate in each step, the value of the traffic rate in the next step is being predicted.

In the model we explore in this chapter, several ARIMA-like filters are created, each operating on a different time horizons. The logic behind this approach is that traffic flows show high auto-correlation properties. The time-axis is divided into time-intervals, also called provisioning-intervals, which are assumed to equal 15 minutes in this discussion. The similarities in traffic behavior at the same hour in preceding days, and preceding weeks can be used to predict the traffic rate between two nodes in the next provisioning interval.

We use the real inter-city traffic measurements taken from in the 'Geant' backbone network [33]. The network topology is shown in figure 4.20. There are 22 nodes in the network. Traffic flows over each link were collected every 15 minutes. Therefore four samples are taken per hour, and 96 samples are taken per day. (The results reported in this thesis use the measurements for a period of 1 month.)

As an example in figure 4.21 the real traffic volume between a pair of nodes for a period of 14 days in the network is shown. As time advances, the traffic rate changes. This change in a 15 minute interval is closely related to the traffic change in the last 15-minute interval. However, we can confirm the similarity of changes in the traffic rates in the same intervals in the last hour, or last day or last week.

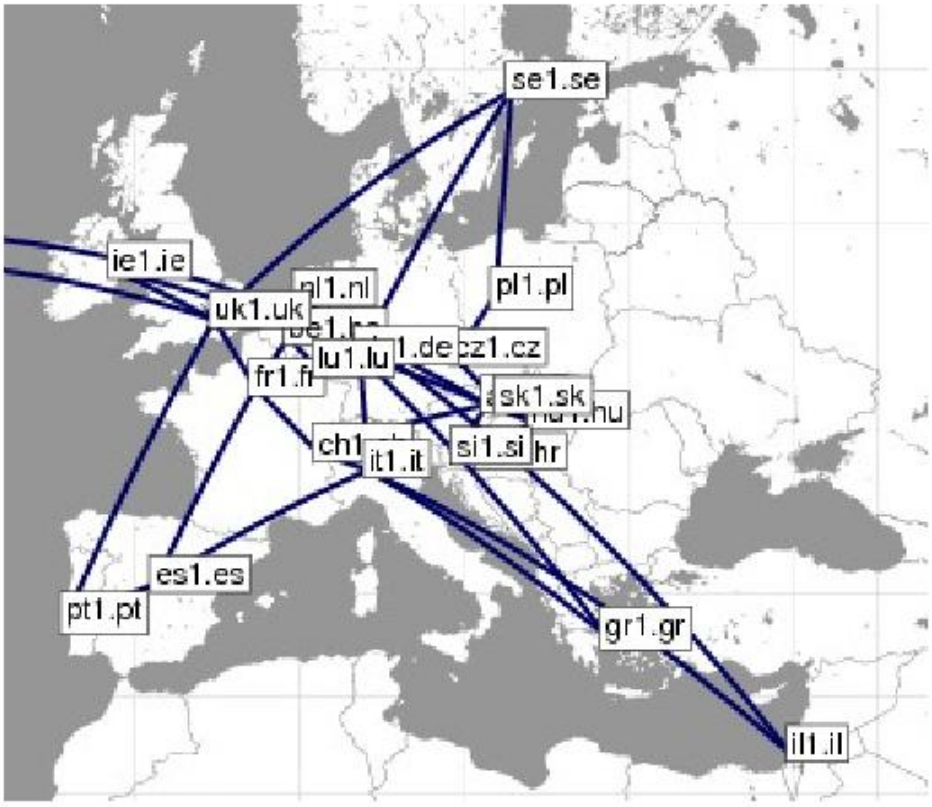

Figure 4.20: The 'Geant' European Backbone Network

Autocorrelation of Traffic

Mathematically, autocorrelation of a random process describes the correlation of values of process at different times [3]. If we consider the traffic demand in a network as a repeatable random process, we can define the autocorrelation of two different times in equation 4.43:

$$auto\_correlation(t,s) = \frac{E[(x_t - \mu_t)(x_s - \mu_s)]}{\sigma_t \sigma_s} \tag{4.42}$$

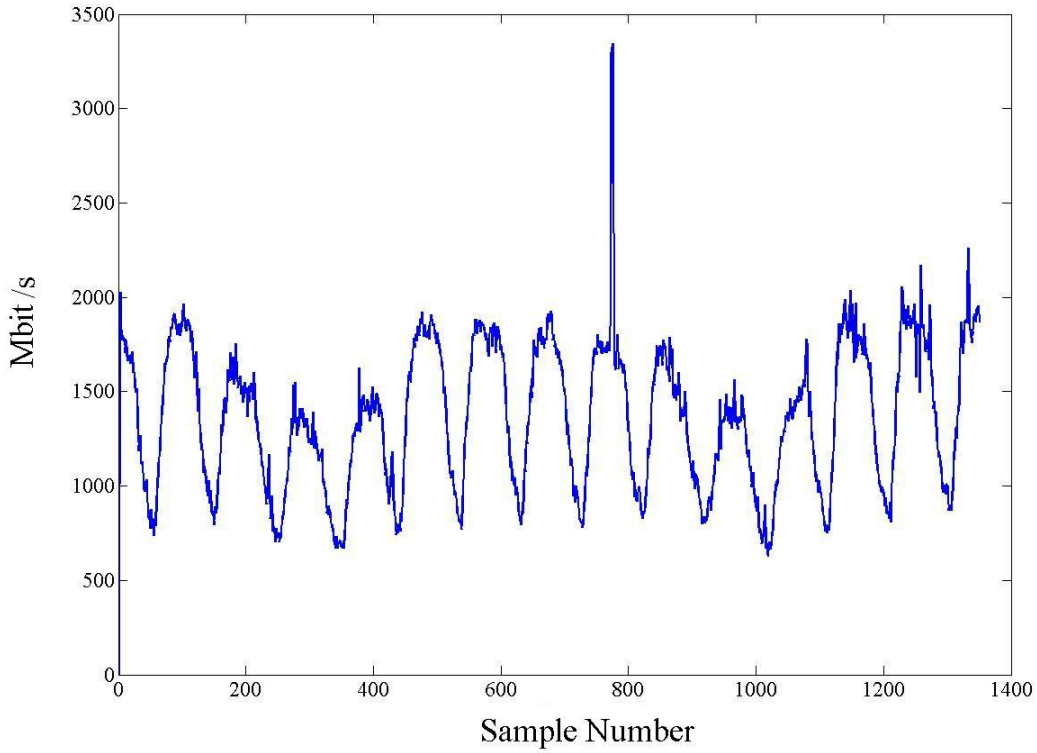

Figure 4.21: Traffic Flows between two nodes in the network

where x_s and x_t are the traffic value of two time scales. For example x_t can represent traffic demands of a week starting today, while x_s can show traffic demands of a week with lag of l relative to the x_t. μ's and σ's represent the mean and the standard deviation of traffic demands at different time scales s and t. This equation can also be a function of l where $l = t - s$, that shows the autocorrelation of two times, with l time slots difference. With $l = 0$ and $s = t$ *auto_correlation* is equal to 1 that shows the maximum of the correlation.

In figure 4.22 the autocorrelation of a traffic flow between one pair of nodes in the 'Geant' network is illustrated. The experimental time is 7 days (There are 96 samples in each day). The maximum correlation is scaled to 1. This figure shows

some interesting properties. For example, for the small l's, close time scales, the autocorrelation function is about 0.7 to 0.8, which shows that the traffic is highly correlated to the near past.

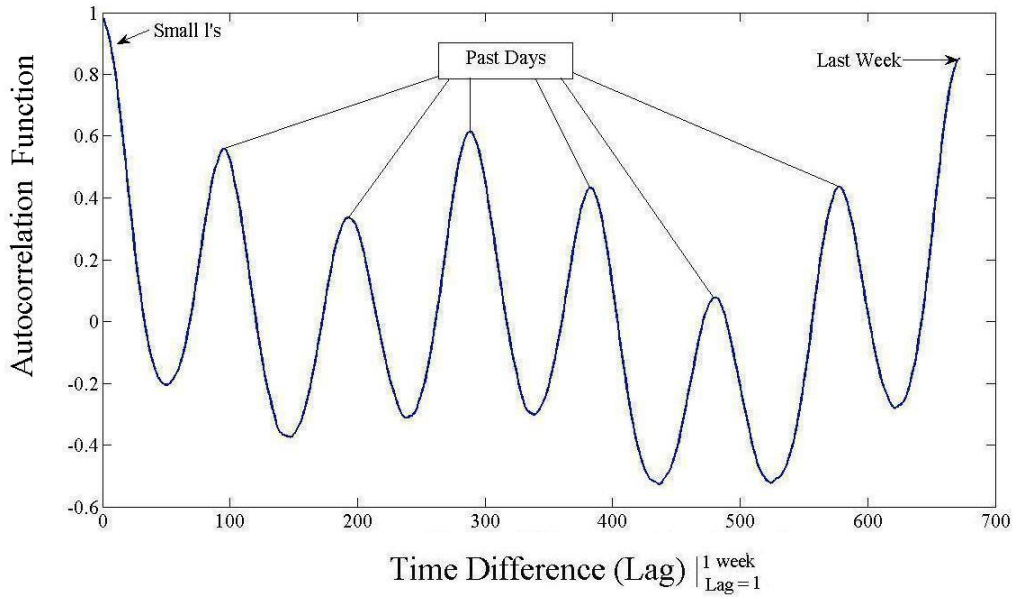

Figure 4.22: Autocorrelation of a traffic flow in the 'Geant' IP Backbone Network

Also, peak values in the figure, show the correlation of the traffic to the same time in previous days. We can infer that the traffic is heavily correlated to the same time in previous days. Another interesting point is the correlation value of the traffic to the same time last week. As each day contains 96 samples, the last week is roughly 700 samples back. We can see for this flow, the traffic is also heavily correlated to the same time last week. The autocorrelation value for the last week, is about 0.9 that proves this thought.

Autocorrelation can also be computed in a similar way computed in [37]. In the

signal processing context [38] [39], the autocorrelation is defined as:

$$auto\_correlation(l) = \sum_{t=T}^{T+N} x(t) \times x(t-l) \tag{4.43}$$

with the l ranging from 0 to the period of measurement, N.

In figure 4.23 the traffic samples of 3 hours between two nodes, along with the same samples shifted by 25 time-slots, is illustrated. Multiplying these two series, we will have the autocorrelation for $l = 25$.

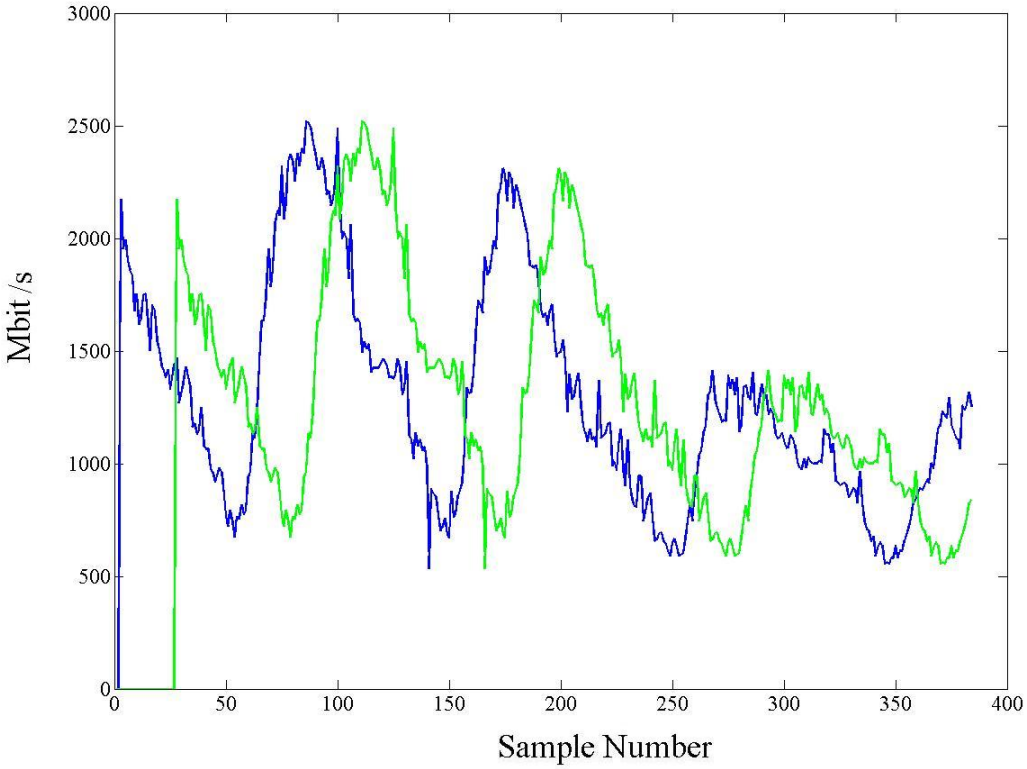

Figure 4.23: Real Traffic volume and 25 time-slot shifted traffic volume

In figure 4.24 the autocorrelation of traffic flows between some pair of nodes in the 'Geant' network is illustrated (before normalization). The time of measurement

is 14 days. The spikes in the figures show high correlations. This figures also confirms the traffic is highly correlated over several different time scales. (The autocorrelation of an uncorrelated signal smoothly decreases to zero). In figure 4.24, the changes in traffic rates are also correlated daily.

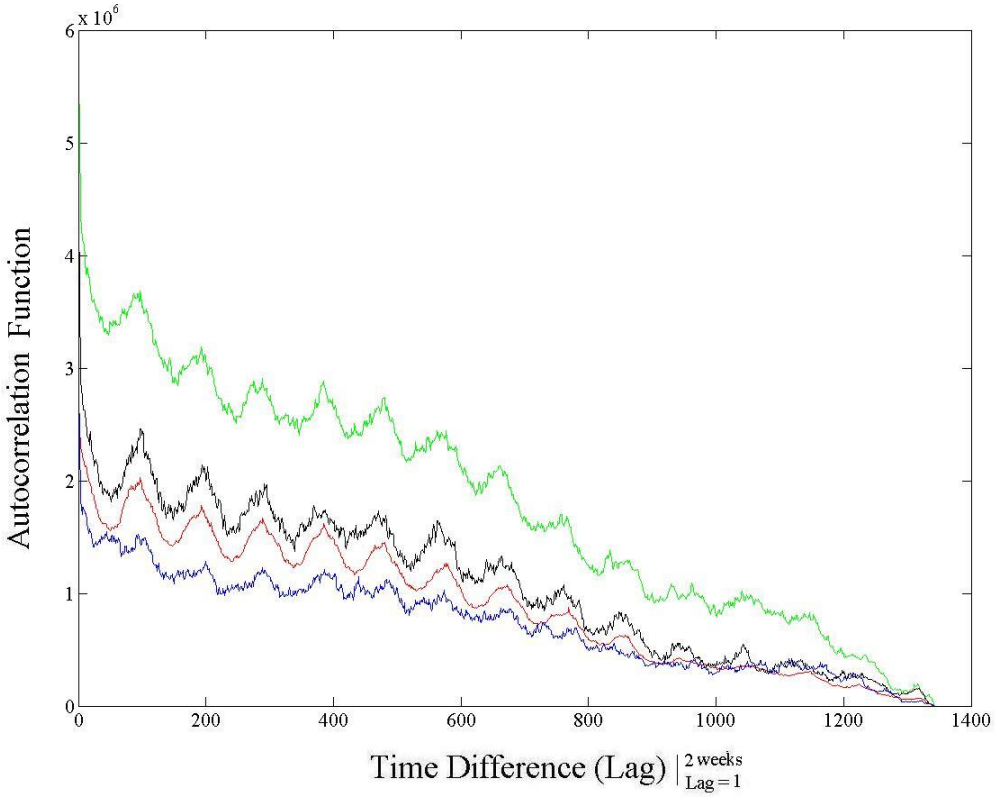

Figure 4.24: Autocorrelation of 4 traffic flows in the 'Geant' backbone network

To have a good prediction for the next time slot, we have to capture all the similarities of the past traffic history, over short-term (immediate) and long-term trends. In this model, multiple ARIMA filters are used to capture the traffic flow over different time scales. The general model consists of a combination of four ARIMA filters operating over four different time scales. These four filters are used separately to

find four separate traffic predictions, which are then combined to create a 'lightweight' (i.e., computationally-efficient) prediction model.

These four time scales are:

1. Quarter-Hour : Exploiting last quarter-hour trend of traffic
2. Hour : Exploiting last hour trend of traffic
3. Day : Exploiting last day same time trend of traffic
4. Week : Exploiting last week same day same time trend of traffic

Network traffic measurements may show transient behaviour that is not related to any long term pattern. Therefore, it may be desirable to eliminate these transient changes in the network, by low-pass filtering the traffic measurements. By doing so, the transient high-frequency changes can be eliminated. The resulting signal may be more stable and more accurate.

A simple digital filter can be applied to filter traffic rates. The general form of a digital filter is shown in equation 4.44:

$$\begin{aligned} a(1) \times \bar{x}_t =& b(1) \times x_t + b(2) \times x_{t-1} + \cdots + b(nb+1) \times x_{t-nb} \\ & - a(2) \times \bar{x}_{t-1} - \cdots - a(na+1) \times \bar{x}_{t-na} \end{aligned} \tag{4.44}$$

where nb is the order of the filter.

In our work we only use the polynomial b as the active coefficient to build a filtered data based on the past traffic history, and we don't take the past filtered data into account. That is $a(n) = 1$ and all other a are zero. More complex filtering schemes will be defined and tested in chapter 5.

We also tested different window lengths for the polynomial b in our model. Figure 4.25 shows different filter lengths chosen for the $b(n)$. Filters with larger nb will eliminate high frequency elements more; as a result some changes which are not transient may be eliminated. However, a small window size is not desirable in a sense it cannot efficiently eliminate the high frequency terms. We found that a window size of 4 is reasonable in our experimental results reported in chapter 5 (if a filter is used at all).

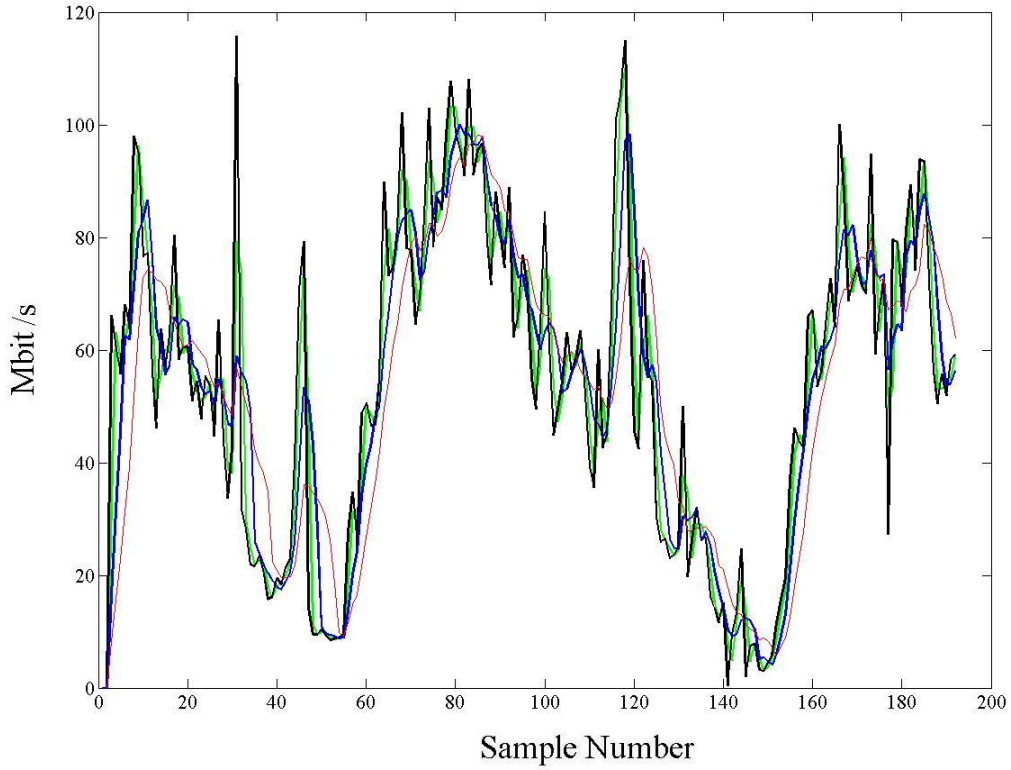

Figure 4.25: Different Window sizes of Low Pass Filters. Black = Real traffic, Green = window size 2, Blue = window size 4, Red = window size 8

4.2 Model Formulation

In this part we explain the model to predict the traffic rate. We use real traffic measurements from the 'Geant' European Backbone network to verify our model, and as mentioned before, the traffic rates are collected every 15 minutes.

The real traffic volume in the network between a pair of nodes i and j at the time-slot t, is shown as $x(i,j)^t$. Here t represents the discrete number of samples gathered from the 'Geant' network, starting from 0. The variable t represents the current time-interval, in multiples of 15-minutes. For example, $x(i,j)^t$ with $t = 3$ represents the traffic volume between node i and j at time t = 45 minutes.

We denote the filtered traffic data, as $\bar{x}(i,j)^t$. As mentioned before, we may use low-pass filters to remove transients. The ARIMA-like filters will then be used to estimate and predict the traffic rates in the next time interval between each pair of nodes. We denote the estimated traffic rate for time t as $\hat{x}(i,j)^t$.

After applying four ARIMA filters, the estimates can be combined to yield the prediction for the next 15 minutes. If an ISP has a good estimate of the bandwidth required for a traffic class on a specific link for a specific time slot, they can provision the bandwidth on that link for that time slot to meet the estimated needs. The model also estimates the provisioning bandwidth for the next time slot, by adding some 'excess' bandwidth to account for transients. The provisioning traffic rate estimate is denoted as $\dot{x}(i,j)^t$.

ARIMA filters exploit the past history of traffic flows. The four ARIMA filters in the model work on different time scales. Therefore, it is necessary to work with different samples of traffic from different time scales. Define the window function w in equation 4.45:

$$v = w(t_s, t_f, s) \tag{4.45}$$

This window function collects the relevant traffic samples over different time scales. This function operates on the traffic flow vectors $[x(i,j)^t]$. It returns a vector v which consists of a sequence of traffic samples, where $v(1)$ here is the value of $x(i,j)$ at the smallest time $t \geq t_s$. The last element $v(k)$ is the value of $x(i,j)$ at the largest time $t \leq t_f$. Elements of v are separated with the step size s in the vector$[x(i,j)^t]$. There are four different ARIMA filters. We describe each of them in the next sections.

4.2.1 Traffic Estimation exploiting Quarter-Hourly history

The first ARIMA filter works on the traffic change in the current 15-minute interval. It predicts the next traffic demand based on the current traffic rate and the current change, which happened in the last quarter-hour. To reduce transient effects, the traffic samples may be low-pass filtered initially. The change in traffic in the current 15-minute interval is shown in equation 4.46:

$$\triangle_q = \bar{x}(i,j)^t - \bar{x}(i,j)^{t-1} \tag{4.46}$$

Based on the predicted traffic change, we can estimate the traffic rate for the next time slot as in equation 4.47:

$$\hat{x}(i,j)_q^{t+1} = x(i,j)^t + \triangle_q \tag{4.47}$$

This equation assumes the same traffic change in the last 15-minutes interval will happen again in the next 15-minute time interval. We can intuitively observe the

logic of this filter by looking at figure 4.21.

Equation 4.47 can also be rewritten as 4.48:

$$\hat{x}(i,j)_q^{t+1} = x(i,j)^t + \sum_{\tau=0}^{\tau=3} w(\tau)x(i,j)^t - \sum_{\tau=1}^{4} w(\tau)x(i,j)^{t-1} \tag{4.48}$$

where w explicitly represents the initial low-pass filtering. Until now, we have a prediction of the traffic for the next 15-minute time slot. Each router or node can use this estimate to provision enough bandwidth for the QoS traffic demands. To compute how much extra traffic is needed to provide for the QoS traffic class, the variance of the traffic over a short-term window is computed. In our experiments the variance over the last 8 samples of traffic flows is determined:

$$\sigma_q^2 = Var(w(t-8,t,1)) \tag{4.49}$$

where the w is the window function that returns the last 8 samples (2 hours). A node may wish to add some excess bandwidth to the estimate, to account for transients. A node can determine the standard deviation of the traffic over the recent short term (i.e., 2 hours), and add some multiple of the standard deviation as 'excess bandwidth', as follows:

$$\dot{x}(i,j)^{t+1} = \hat{x}(i,j)^{t+1} + k_q\sigma_q$$

$$That\ \ is \tag{4.50}$$

$$\dot{x}(i,j)^{t+1} = x(i,j)^t + \triangle_q + k_q\sigma_q$$

It is obvious that by increasing the k_q the provisioning bandwidth provided for the next time slot can cover more transient changes in the QoS traffic.

4.2.2 Traffic Estimation Exploiting Hourly history

The previous ARIMA filter exploiting the last quarter-hour information cannot capture the traffic changes that happened over the last hour. Therefore another filter can be used to get another estimate. Here a second ARIMA filter is introduced to capture the traffic information over the last hour. Based on this information, a second estimate can be made.

The process is similar to the previous ARIMA filter design. First step is to compute several recent changes in traffic history. The second step is to predict the next change by computing a weighted-moving-average of these recent traffic changes. Denote the predicted change based on the history of the last hour as $\triangle_h$, and it can be obtained as follows:

$$\triangle_h = \sum_{\tau=0}^{3} z_h(\tau)(\bar{x}(i,j)^{t-\tau} - \bar{x}(i,j)^{t-\tau-1}) \tag{4.51}$$

In equation 4.51 the traffic changes that occurred over the last hour are computed. The final prediction of the next change in traffic, $\triangle_h$ is the moving average of the these changes. Therefore the estimate of the next traffic volume at time $t+1$ can be computed as:

$$\hat{x}(i,j)_h^{t+1} = x(i,j)^t + \triangle_h \tag{4.52}$$

Equation 4.52 delivers the prediction of the traffic rate, based on the observations

of the last hour. To combine the quarter-hourly estimate and hourly estimates, a weighted average of these two can be used to compute a final traffic rate estimate. The weighted average of these two models is determined as follows:

$$\hat{x}(i,j)^{t+1} = w_q^t \times \hat{x}(i,j)_q^{t+1} + w_h^t \times \hat{x}(i,j)_h^{t+1} \tag{4.53}$$

To provision enough QoS bandwidth for the next time slot, the variance of the traffic over the last 4 samples (last hour) is computed, as follows:

$$\sigma_h^2 = Var(w(t-4,t,1)) \tag{4.54}$$

The amount of excess bandwidth provisioned for the QoS traffic can also be calculated by a weighted average of the standard deviations of these two models.

$$\dot{x}(i,j)^{t+1} = \hat{x}(i,j)^{t+1} + w_q^t k_q \sigma_q + w_h^t k_h \sigma_h \tag{4.55}$$

The weights play an important role in the equation 4.53 to find the final estimate. It is sensible that the prediction model with more accuracy should have a larger weight and larger influence on the future prediction. As time advances the real traffic demand will be measured, and the accuracy of each model can be computed. The weights can then be adjusted based on the accuracy of each model. The accuracy of each model can be determined by computing the difference between the estimate and the real traffic demand:

$$\begin{aligned} \epsilon_q &= |\hat{x}_q^{t+1} - x(i,j)^{t+1}| \\ \epsilon_h &= |\hat{x}_h^{t+1} - x(i,j)^{t+1}| \end{aligned} \tag{4.56}$$

The error term computed in 4.56 can be used to update the weights after observing the real traffic demand. The weights are updated as follows:

$$\begin{aligned} w_q^{t+1} &= \epsilon_h/(\epsilon_q + \epsilon_h) \\ w_h^{t+1} &= \epsilon_q/(\epsilon_q + \epsilon_h) \end{aligned} \tag{4.57}$$

The sum of the these weights is unity. The weights add more emphasis to the better estimate.

4.3 Traffic Estimation Exploiting Daily history

The first two ARIMA filters compute the traffic changes over the short term, i.e., the last quarter-hour and the last hour. But the traffic in figure 4.21 confirms intuitively that similarities exist with the same time in the last day (96 samples ago) or days. This similarity can also be used in the prediction process. Here, a third ARIMA filter is introduced which exploits the traffic changes in the same hour in the last day(s). The moving average of changes over several day(s) helps to predict the future change of traffic demand.

The real traffic of the 'Geant' backbone network is gathered every 15 minutes. Consequently there are 96 samples of traffic rates in a day. A weighted-moving-average of the changes that happened in the same hour in the last day can be computed as follows:

$$\triangle_d = \sum_{\tau=0}^{3} z_d(\tau)(\bar{x}(i,j)^{t-96-\tau+1} - \bar{x}(i,j)^{t-96-\tau}) \tag{4.58}$$

Here again the filtered data is used. Equation 4.58 calculates the change in traffic in the same time interval in last day as a prediction for the future change. Similarly, the estimate of the third ARIMA filter is computed using:

$$\hat{x}(i,j)_d^{t+1} = x(i,j)^t + \triangle_d \tag{4.59}$$

After computing the estimation for the future change of traffic volume, based on the observation in the last day, the final traffic estimate can be computed using a weighted average of all these three ARIMA filters, as shown in equation 4.60:

$$\begin{aligned} \hat{x}(i,j)^{t+1} = w_q^t \times \hat{x}(i,j)_q^{t+1} \\ + w_h^t \times \hat{x}(i,j)_h^{t+1} \\ + w_d^t \times \hat{x}(i,j)_d^{t+1} \end{aligned} \tag{4.60}$$

The weighted average of the different estimations' models creates a combined model that puts more emphasis on the model with the best accuracy. The last step is to provide excess bandwidth for the QoS traffic class. The variance of the traffic over a 1 hour interval in the last day is computed. We call it σ_d and it can be computed as:

$$\sigma_d^2 = Var(w(t-96-3, t-96, 1)) \tag{4.61}$$

$$\begin{aligned} \dot{x}(i,j)^{t+1} = \hat{x}(i,j)^{t+1} + w_q^t k_q \sigma_q \\ + w_h^t k_h \sigma_h \\ + w_d^t k_d \sigma_d \end{aligned} \tag{4.62}$$

The weights used when combining these 3 filters should be adjusted based on their accuracy. As time advances, the real traffic is measured and compared with the estimated value of each model. We can the determine the accuracy of each model.

$$\begin{aligned} \epsilon_q &= |\hat{x}_q^{t+1} - x(i,j)^{t+1}| \\ \epsilon_h &= |\hat{x}_h^{t+1} - x(i,j)^{t+1}| \\ \epsilon_d &= |\hat{x}_d^{t+1} - x(i,j)^{t+1}| \end{aligned} \tag{4.63}$$

Equation 4.63 measures the accuracy of each model. The weights are updated as shown in equation 4.64, letting $\theta \in (q, d, h)$:

$$w_\theta^{t+1} = (1/\epsilon_\theta)/(1/\epsilon_q + 1/\epsilon_h + 1/\epsilon_d) \tag{4.64}$$

Again the sum of all weights is unity. This weight updating method puts more emphasis on the more accurate model. We can also extend the filter length in this daily ARIMA filter, and call it the Multi-Day ARIMA filter (see chapter 5).

4.4 Traffic Estimation Exploiting weekly history

The fourth and final ARIMA filter is created to capture changes that happened in the traffic in the same hour over the last week(s). Comparing the traffic rate of the present time and the same time in the last week(s), in figure 4.21, we can observe the long-term correlation between traffic behavior on the weekly time scale.

Similar to the daily model, ARIMA filter here computes the traffic changes in an hour interval at the same time in the last week(s). The future change can be predicted by a weighted-moving-average of the changes over the last week(s):

$$\triangle_w = \sum_{\tau=0}^{3} z_w(\tau)(\bar{x}(i,j)^{t-96\times7-\tau+1} - \bar{x}(i,j)^{t-96\times7-\tau}) \tag{4.65}$$

The estimate of this filter is computed as follows:

$$\hat{x}(i,j)_w^{t+1} = x(i,j)^t + \triangle_w \tag{4.66}$$

The final estimation is the combined estimation of all ARIMA filters, and it can be computed using a weighted average of the 4 filter estimates. This is shown in equation 4.67.

$$\begin{aligned} \hat{x}(i,j)^{t+1} = & \, w_q^t \times \hat{x}(i,j)_q^{t+1} \\ & + w_h^t \times \hat{x}(i,j)_h^{t+1} \\ & + w_d^t \times \hat{x}(i,j)_d^{t+1} \\ & + w_w^t \times \hat{x}(i,j)_w^{t+1} \end{aligned} \tag{4.67}$$

To provision excess bandwidth for the next QoS traffic demand based on the traffic behavior in the same hour in the last week, the variance of the traffic in the last week is computed:

$$\sigma_w^2 = Var(w(t - 96 \times 7 - 3, t - 96 \times 7, 1)) \tag{4.68}$$

The provisioning bandwidth based on all four filters has two terms. The final term is the combination of all estimations. The second term is the excess bandwidth that is added to account for transients. It is a combination of standard deviations of traffic

trends in the four different time scales. It is shown in equation 4.69.

$$\begin{aligned} \dot{x}(i,j)^{t+1} = \hat{x}(i,j)^{t+1} &+ w_q^t k_q \sigma_q \\ &+ w_h^t k_h \sigma_h \\ &+ w_d^t k_d \sigma_d \\ &+ w_w^t k_w \sigma_w \end{aligned} \tag{4.69}$$

When combining all four filter models, the weights put more emphasis on the most accurate model. As time advances, the accuracy of each model is computed and the weights are updated for the next prediction interval. To update the weights, we first calculate the accuracy of each filter. By letting $\theta \in (q, d, h)$ we have:

$$\epsilon_\theta = |\hat{x}_\theta^{t+1} - x(i,j)^{t+1}| \tag{4.70}$$

To update the weights we have:

$$w_\theta^{t+1} = (1/\epsilon_\theta)/(1/\epsilon_q + 1/\epsilon_h + 1/\epsilon_d + 1/\epsilon_w) \tag{4.71}$$

In the next chapter, the experimental results of this model will be presented.

4.5 Conclusion of Chapter 4

This chapter has summarized a traffic provisioning model based upon several ARIMA-like filters operating in parallel. Each filter operates in a different time scale, i.e., quarter-hour, hour, day or week. The model is very general, as the number of measurements considered in each filter can be changed by adjusting the filter lengths, and the filter weights can also be adjusted. The estimates of each filter are then combined using a weighted average to yield a single estimate. When computing the combined estimate, the filter with the highest accuracy has the highest weight, and the weights change dynamically. Several experiments of the model are presented in the next chapter.

Chapter 5

Algorithm Evaluation

5.1 Chapter Organization

In this chapter we explore the robustness of the model described in chapter 4, to changes in filter parameters. Several different metrics to evaluate the model are reviewed. As described in chapter 4, the model consists of several individual ARIMA filters, each operating at a different time-scale. Each of these individual filters can be evaluated separately for its prediction ability. Alternatively, all the models can be combined as described in chapter 4, in order to present a weighted average prediction. To evaluate these models, we use real traffic demand matrices measured every 15 minutes in the Geant European backbone network, as the input for our model.

Two types of filters are used in the model, and each filter can be adjusted to measure the robustness of the model. First, an *'Input Filter'* can be used to low-pass filter all the measured traffic demand matrices, to remove short-term transients. Second, the filter parameters for each individual ARIMA filter can be adjusted. In this chapter, several different types of digital low-pass filters are tested as the Input

Filter. In addition, several different filter parameters can be used in each ARIMA filter. In particular, the length and the weights of the individual ARIMA filters can be adjusted.

In the next section, several different metrics of evaluation are reviewed.

5.2 Metrics of Evaluation

Several different metrics are used to evaluate the algorithm. These metrics are reviewed.

5.2.1 Mean Satisfied Bandwidth

The first metric is the *Mean Satisfied Bandwidth* (MSB). This metric quantifies the ability of the prediction algorithm to meet the future bandwidth demands of the QoS traffic class. The goal of the prediction algorithm is to provision enough bandwidth to meet the future traffic demands of the QoS traffic class, while minimizing the amount of 'excess bandwidth' in the provisioned bandwidth. The *Mean Satisfied Bandwidth* is defined as the fraction of the traffic demand in the next time slot, that is satisfied by the traffic estimate, as shown in Eq. 5.72:

$$\alpha = \frac{1}{t}\sum_{\tau=1}^{t}[min(\dot{x}(i,j)^{\tau}, x(i,j)^{\tau})/(x(i,j)^{\tau}] \tag{5.72}$$

In Eq. 5.72, $\dot{x}(i,j)$ is the provisioning bandwidth for flow (i,j) and $x(i,j)$ is the real traffic demand for flow (i,j), at time t. The equation yields a fraction between 0 and 1. In the tables to be presented later in this chapter, we report the *Mean Satisfied Bandwidth* as a percentage. If the prediction algorithm can provision enough

bandwidth in the next time-slot to completely satisfy the QoS traffic demand, i.e. $\dot{x}(i,j)^{t+1} \geq x(i,j^{t+1})$, then 100% of the QoS traffic is satisfied and the MSB is 100%. However, if the provisioned bandwidth is less than the QoS bandwidth demand, then the fraction of the QoS demand that is satisfied is given by $\dot{x}(i,j)^{t+1}/x(i,j)^{t+1}$.

5.2.2 Mean Excess Bandwidth

The *Mean Excess Bandwidth* (MEB) is defined as the unused excess bandwidth provisioned for the next time slot, in excess of the actual traffic demand in the next time slot. It is expressed as a fraction of the actual traffic demand. This excess bandwidth is not used by the QoS-class traffic, and can be used by the Best-Effort traffic class. This metric is defined in equation 5.73:

$$\gamma = \frac{1}{t}\sum_{\tau=1}^{t}[max(\dot{x}(i,j)^{\tau} - x(i,j)^{\tau}, 0))/(x(i,j)^{\tau}] \tag{5.73}$$

The MEB is a fraction between 0 and 1. In the following tables, it will be reported as a percent of the actual traffic demand in each time-slot.

5.2.3 Error

The previous metrics evaluate the ability of the prediction algorithm to provide enough bandwidth to meet the future demands. The error metric presented in this section aims to quantify the accuracy of the prediction algorithm. First the Absolute Error can be expressed as:

$$E = \sum_{\tau=1}^{t} |\hat{x}(i,j)^{\tau} - x(i,j)^{\tau}| \tag{5.74}$$

where $\hat{x}$ is the predicted value. This metric yields the sum of the absolute errors, and its unit is Megabits per second. The absolute error of one prediction equals the absolute value of the difference between the predicted value from the real value. The Relative Error can be defined in equation 5.75, with respect to the mean value of real traffic rates:

$$E_{R.E.} = \frac{\frac{1}{t}\sum_{\tau=1}^{t} |\hat{x}(i,j)^{\tau} - x(i,j)^{\tau}|}{\frac{1}{t}\sum_{\tau=1}^{t} x(i,j)^{\tau}} \tag{5.75}$$

The relative error is the absolute error, normalized by the value of the actual traffic demand at time t. The value of this metric is a fraction between 0 and 1. In the following tables, it will be reported as a percent of the actual traffic demand in each time-slot.

Another error term that is useful in evaluating the prediction accuracy is the *Mean-Squared-Prediction-Error* (MSPE). It is a well-known metric in evaluating prediction algorithms. It can be defined in equation 5.76:

$$M.S.P.E. = \frac{1}{t}\sum_{\tau=1}^{t} [\hat{x}(i,j)^{\tau} - x(i,j)^{\tau}]^2 \tag{5.76}$$

It is desirable to normalize the MSPE, relative to the actual traffic demand at time t, as shown in equation 5.77. Equation 5.77, yields the Normalized-Mean-Squared-Prediction-Error, E_{NMSE}:

$$E_{N.M.S.P.E.} = \frac{\sqrt{(1/t)\sum_{\tau=1}^{t} (\hat{x}(i,j)^{\tau} - x(i,j)^{\tau})^2}}{1/t\sum_{\tau=1}^{t} x(i,j)^{\tau}} \tag{5.77}$$

The value of this metric is usually between 0 and 1.

5.3 Input Filter

As described in the introduction to Chapter 5, there are 2 types of filters in the model. The *Input Filters* can be used to low-pass filter the traffic demand matrices, to remove transient changes. In this section, we evaluate several different input filters, to explore the robustness of the model. In the next section, we will evaluate several different ARIMA filters.

5.3.1 A Simple Digital Input Filter

At the first step, we will experiment with very simple Input Filters with equal filter weights. These filters are being used to smoothen the traffic demands and decrease the transient changes. The general form is as follows:

$$\begin{aligned} &\bar{x}(n) = w(1) * x(n) + w(2) * x(n-1) + \cdots + w(n_w)x(n-n_w) \\ &\bar{X} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n] \\ &X = [x_1, x_2, \ldots, x_n] \\ &W = [w_1, w_2, \ldots, w_{n_w}] \end{aligned} \tag{5.78}$$

In equation 5.78, $\bar{X}$ is the vector of filtered traffic demands of flow (i, j), X is the vector of real traffic demands of that flow, and W is the vector of Input Filter weights, where n_w is the order of the filter. It is obvious that by increasing the n_w, we can have better smoothing low-pass filters. It is obvious that $n_w = 1$ implies no filtering. Here the weight vector plays a key role in the filter performance. We can run the experiment for the case of no Input Filtering, and for 3 different sets of Input

Filters with different orders; $n_w = 4, 8, 16$;

$$W = ones(1, n_w)/n_w \tag{5.79}$$

For example, $n_w = 8$, implies an 8-element vector with equal weights $W = [1/8, 1/8, \ldots, 1/8]$. The results of the experiments with these three vectors is shown next.

Mean Satisfied Bandwidth

The MSB with different length choices of Input Filters is shown in table 5.1. It can be inferred from the table 5.1 that, in general the individual ARIMA filters perform slightly better when Input Filtering is applied. However, when all these individual ARIMA filters are combined to produce a weighted prediction in the model ARIMA (Q,H,D,W), the Mean Satisfied Bandwidth is maximized when no Input Filter is applied. These results suggest that the combined ARIMA model is the best model and that it is relatively robust to changes in the Input Filters.

Mean Excess Bandwidth

Table 5.2 shows the MEB for different length choices of Input Filters. It can be inferred from the table 5.2 that, in general the individual ARIMA filters perform slightly better when Input Filtering is applied. However, when all these individual ARIMA filters are combined to produce a weighted prediction, the Mean Excess Bandwidth is minimized when no Input Filter is applied. These results suggest that the combined ARIMA model is also the best model and that it is relatively robust to changes in the Input Filters.

MSB	ARIMA Filters	Unfiltered Data	Filtered W_L=4	Filtered W_L=8	Filtered W_L=16
K=0	ARIMA(Q)	83.72	86.44	87.19	87.58
	ARIMA(H)	85.62	86.9	87.4	87.69
	ARIMA(D)	86.28	87.25	87.61	87.8
	ARIMA(W)	86.45	87.24	87.58	87.76
	ARIMA(Q,H,D,W)	89.1	88.55	88.32	88.16
K=1	ARIMA(Q)	91.57	94.15	94.44	94.39
	ARIMA(H)	93.77	94.36	94.43	94.42
	ARIMA(D)	93.49	94.14	94.37	94.49
	ARIMA(W)	93.57	94.12	94.33	94.43
	ARIMA(Q,H,D,W)	94.82	94.66	94.62	94.57
K=2	ARIMA(Q)	95.3	96.67	96.76	96.7
	ARIMA(H)	96.52	96.73	96.73	96.71
	ARIMA(D)	96.15	96.54	96.67	96.74
	ARIMA(W)	96.19	96.52	96.64	96.69
	ARIMA(Q,H,D,W)	96.9	96.84	96.81	96.78

Table 5.1: Mean Satisfied Bandwidth (PERCENT); Filtering with Different Weight Length

MEB	ARIMA Filters	Unfiltered Data	Filtered W_L=4	Filtered W_L=8	Filtered W_L=16
K=0	ARIMA(Q)	19.53	14.35	13.12	12.54
	ARIMA(H)	15.46	13.61	12.82	12.4
	ARIMA(D)	14.79	13.2	12.56	12.23
	ARIMA(W)	14.85	13.44	12.84	12.52
	ARIMA(Q,H,D,W)	11.48	11.81	11.87	11.93
K=1	ARIMA(Q)	40.1	36.34	36.05	36.18
	ARIMA(H)	36.73	36.13	36.08	36.14
	ARIMA(D)	37.63	36.62	36.25	36.06
	ARIMA(W)	37.73	36.83	36.49	36.32
	ARIMA(Q,H,D,W)	35.98	36.09	36.04	36.04
K=2	ARIMA(Q)	66.02	64.39	64.31	64.43
	ARIMA(H)	64.54	64.33	64.34	64.41
	ARIMA(D)	65.42	64.75	64.49	64.37
	ARIMA(W)	65.51	64.92	64.7	64.59
	ARIMA(Q,H,D,W)	64.41	64.48	64.42	64.4

Table 5.2: Mean Excess Bandwidth (PERCENT); Filtering with Different Weight Length

Relative Error

Table 5.3 shows the relative error (RE) for different choices of Input Filters. Based on the results of this table, the same conclusion can be drawn; that using Input Filters can decrease the prediction error of each individual ARIMA filter, while the combination of ARIMA filters has a better accuracy when no Input Filter applied.

RE	ARIMA Filters	Unfiltered Data	Filtered W_L=4	Filtered W_L=8	Filtered W_L=16
K=0,1,2	ARIMA(Q)	28.23	20.67	18.91	18.08
	ARIMA(H)	22.31	19.59	18.47	17.87
	ARIMA(D)	21.29	19	18.1	17.64
	ARIMA(W)	20.67	18.85	18.04	17.63
	ARIMA(Q,H,D,W)	16.31	16.84	16.99	17.13

Table 5.3: Relative Error (PERCENT); Filtering with Different Weight Length

Normalized Mean Squared Prediction Error

In table 5.4, the NMSPE for different choices of Input Filters is shown. Based on the results of this table, using Input Filters can slightly reduce the NMSPE, in the individual ARIMA filters and in the combined ARIMA filter. The differences however, are limited to 3 percent, which is not too large. For example, comparing different columns of this table one can notice that using an Input Filter with length 16 yields a slightly lower NMSPE. Unfortunately, the most important metrics are the Mean Satisfied Bandwidth and Mean Excess Bandwidth, and the use of Input Filters slightly decreases the performance of those metrics.

NMSPE	ARIMA Filters	Unfiltered Data	Filtered W_L=4	Filtered W_L=8	Filtered W_L=16
K=0,1,2	ARIMA(Q)	79.59	57.01	52.99	51.19
	ARIMA(H)	62.09	54.02	51.74	50.58
	ARIMA(D)	54.54	50.91	50.03	49.68
	ARIMA(W)	52.87	50.67	49.96	49.69
	ARIMA(Q,H,D,W)	52.83	51.16	50.44	50.02

Table 5.4: Normalized Mean Squared Prediction Error (PERCENT);Filtering with Different Weight Length

5.3.2 Using More Complex Input Filters

To decrease the effects of transients, we can take advantage of more complex digital Input Filters with different filter weights. In this section we describe the design and use of more complex digital filters. Here we take advantage of low-pass Butterworth digital filters. By changing the order and cutoff frequency of the filter we can get different results. The use of simple digital Input Filters in the previous section showed mixed results. The use of Input Filters results in a slight improvement in the NMSPE metric, while it results in a slight reduction in the MSB and MEB metrics. In this section we explore more complex digital filters to see how they influence the overall performance.

Cutoff frequency is the frequency where the magnitude response of the filter is $\sqrt{1/2}$ of the nominal passband value. The Order of the filter is the highest term of the numerator and denominator. Comparing two Butterworth filters, the one with higher order will result in more attenuation of the terms with higher frequencies.

As mentioned in [35], Butterworth digital filters have a flat frequency response in the passband, and its response magnitude decays monotonically as the frequency increases. Other filters could also be used such as Chebyshev filters type one and two.

However those filters have more ripples in the passband than Butterworth filters. The frequency response of two Butterworth filter is shown in figures 5.26 and 5.27.

The general transfer function form of the Butterworth digital filter is as follows:

$$H(z) = \frac{a(1)z^n + \cdots + a(n+1)}{z^n + b(2)z^{n-1} + \cdots + b(n+1)} \tag{5.80}$$

Where n is the order of the filter, $A = [a(1), \ldots, a(n+1)]$ is the numerator's coefficients, $B = [1, b(2), \ldots, b(n+1)]$ is the denominator's coefficients.

We can use the general form of the Butterworth digital filter to design a low-pass digital filter with a desired order and cutoff frequency. Frequency responses of two low-pass filters with same cutoff frequency but different orders are shown in figures 5.26 and 5.27. Comparing the magnitude response of these two filters, its is noticeable that the magnitude decreases more rapidly.

We performed experiments on Input Filters with different orders and cutoff frequencies, with orders being $n = 1, 2, 3$ and cutoff frequencies being $f_s = 0.7, 0.8, 0.9$ relative to the Nyquist frequency. Nyquist frequency equals half of the sampling frequency. We consider traffic demands as a constitutions function, and we sampled this function every 15 minutes. As a result the Nyquist frequency is the highest frequency of the samples. By designing different filters in terms of order and cutoff frequency, we can apply different Butterworth digital filters as the Input Filters. The results are shown next.

Mean Satisfied Bandwidth, using Butterworth Input Filters

The Mean Satisfied Bandwidth for different types of Butterworth Digital Input Filters, with different orders and cutoff frequencies, is shown in table 5.5.

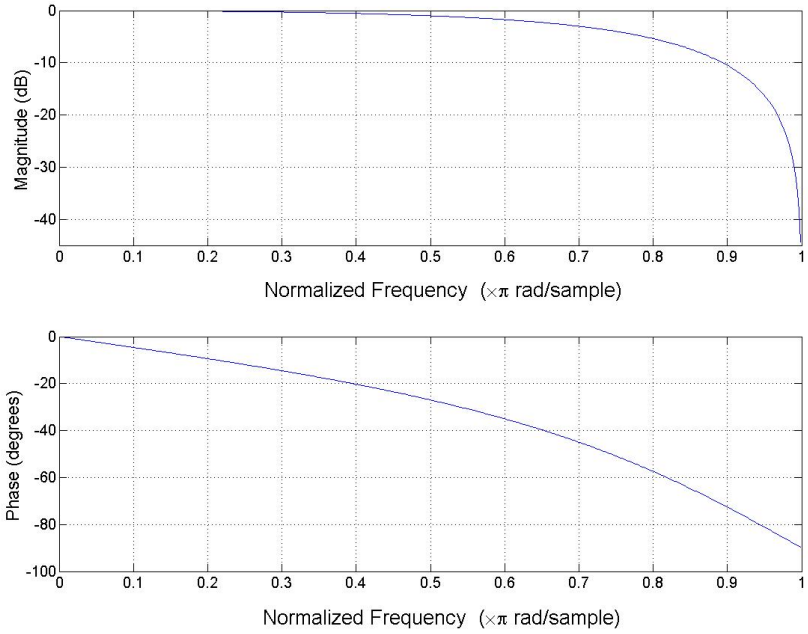

Figure 5.26: Frequency Response of Butterworth low-pass filter with order =1 and Cutoff freq = 0.7 relative to Nyquist Frequency

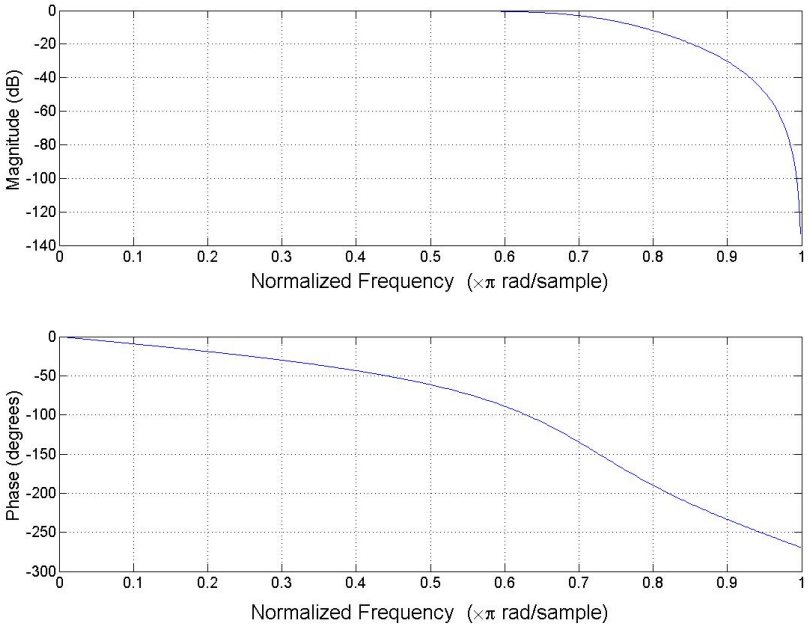

Figure 5.27: Frequency Response of Butterworth low-pass filter with order =3 and Cutoff freq = 0.7 relative to Nyquist Frequency

		Filter Order = 1			Filter Order = 2			Filter Order = 3			Unfiltered
		Cutoff_freQ			Cutoff_freQ			Cutoff_freQ			
MSB	ARIMA Filters	0.7	0.8	0.9	0.7	0.8	0.9	0.7	0.8	0.9	
K=0	ARIMA(Q)	84.64	84.26	83.87	84.71	84.28	83.75	84.57	84.29	83.72	83.72
	ARIMA(H)	86	85.88	85.75	86.03	85.89	85.75	86.08	85.9	85.75	85.62
	ARIMA(D)	86.61	86.51	86.4	86.58	86.49	86.38	86.56	86.46	86.37	86.28
	ARIMA(W)	86.68	86.6	86.5	86.61	86.55	86.48	86.56	86.5	86.45	86.45
	ARIMA(Q,H,D,W)	88.95	89.04	89.13	89.07	89.17	89.23	89.13	89.25	89.28	89.1
K=1	ARIMA(Q)	92.53	92.22	91.92	92.39	92.02	91.7	92.44	91.95	91.57	91.57
	ARIMA(H)	93.95	93.88	93.82	93.95	93.82	93.75	94	93.84	93.7	93.77
	ARIMA(D)	93.71	93.63	93.55	93.66	93.58	93.51	93.63	93.54	93.48	93.49
	ARIMA(W)	93.71	93.64	93.58	93.63	93.57	93.53	93.59	93.51	93.5	93.57
	ARIMA(Q,H,D,W)	94.73	94.76	94.79	94.74	94.77	94.8	94.75	94.78	94.8	94.82
K=2	ARIMA(Q)	96.05	95.89	95.7	95.96	95.69	95.58	96.06	95.66	95.43	95.3
	ARIMA(H)	96.59	96.56	96.53	96.6	96.55	96.51	96.62	96.56	96.49	96.52
	ARIMA(D)	96.28	96.23	96.19	96.25	96.19	96.16	96.22	96.17	96.13	96.15
	ARIMA(W)	96.27	96.23	96.19	96.22	96.18	96.15	96.18	96.14	96.13	96.19
	ARIMA(Q,H,D,W)	96.86	96.87	96.89	96.86	96.86	96.88	96.86	96.86	96.87	96.9

Table 5.5: Mean Satisfied Bandwidth with different order and cutoff frequencies (PERCENT)

It can be inferred from the table 5.5 that, in general the individual ARIMA filters perform slightly better when Input Filtering is applied. However, when all these individual ARIMA filters are combined to produce a weighted prediction, the MSB is maximized when no Input Filter is applied (for parameters k= (1,2)). For parameter k=0, the MSB is slightly higher with Input Filtering, but the Mean Excess Bandwidth is also slightly higher, so there is no net improvement due to Input Filtering. These results tend to confirm the results in the previous section, and suggest that the combined model is also the best model and that it is relatively robust to changes in the Input Filters.

Mean Excess Bandwidth, using Butterworth Input Filters

Table 5.6 shows the MEB for different choices of Input Filters.

		Filter Order = 1			Filter Order = 2			Filter Order = 3			Unfiltered
		Cutoff_freQ			Cutoff_freQ			Cutoff_freQ			
MEB	ARIMA Filters	0.7	0.8	0.9	0.7	0.8	0.9	0.7	0.8	0.9	
K=0	ARIMA(Q)	17.83	18.56	19.35	18.04	18.87	19.78	18.29	19.1	20	19.53
	ARIMA(H)	14.96	15.16	15.35	15.01	15.26	15.45	15	15.3	15.52	15.46
	ARIMA(D)	14.29	14.47	14.67	14.41	14.61	14.77	14.49	14.69	14.84	14.79
	ARIMA(W)	14.45	14.61	14.78	14.6	14.77	14.89	14.7	14.87	14.96	14.85
	ARIMA(Q,H,D,W)	11.73	11.69	11.6	11.73	11.73	11.64	11.64	11.71	11.67	11.48
K=1	ARIMA(Q)	38.38	38.78	39.25	38.7	39.28	39.59	38.56	39.42	39.93	40.1
	ARIMA(H)	36.54	36.61	36.68	36.53	36.67	36.75	36.49	36.66	36.81	36.73
	ARIMA(D)	37.32	37.44	37.57	37.43	37.57	37.67	37.48	37.64	37.74	37.63
	ARIMA(W)	37.47	37.59	37.71	37.6	37.73	37.81	37.68	37.82	37.87	37.73
	ARIMA(Q,H,D,W)	36.17	36.16	36.09	36.27	36.29	36.19	36.2	36.33	36.27	35.98
K=2	ARIMA(Q)	65.01	65.18	65.41	65.11	65.44	65.57	65	65.51	65.82	66.02
	ARIMA(H)	64.47	64.5	64.53	64.46	64.51	64.56	64.43	64.5	64.58	64.54
	ARIMA(D)	65.21	65.3	65.38	65.3	65.4	65.46	65.34	65.45	65.52	65.42
	ARIMA(W)	65.33	65.41	65.49	65.43	65.52	65.57	65.49	65.58	65.61	65.51
	ARIMA(Q,H,D,W)	64.58	64.58	64.52	64.68	64.72	64.63	64.63	64.76	64.71	64.41

Table 5.6: Mean Excess Bandwidth with different orders and cutoff frequencies (PERCENT)

It can be inferred from table 5.6 that, in general the individual ARIMA filters yield a slightly better MEB when no Input Filtering is used. When all these individual ARIMA filters are combined to produce a weighted prediction, the MEB is also minimized when no Input Filter is used. These results tend to confirm the results in the previous section, and suggest that the Input Filters do not improve the predictions, according to the MEB metric.

Relative Error, using Butterworth Input Filters

Table 5.7 shows results of the Relative Error metric, when we use different digital filters with different orders and cutoff frequencies. In terms of relative error, table 5.7 reports interesting results. The use of Input Filters yields a lower Relative Error metric for the combined ARIMA filter. However, for each individual ARIMA filter alone, the Relative Error is reduced when no Input Filter is applied.

		Filter Order = 1			Filter Order = 2			Filter Order = 3			Unfiltered
		Cutoff_freQ			Cutoff_freQ			Cutoff_freQ			
RE	ARIMA Filters	0.7	0.8	0.9	0.7	0.8	0.9	0.7	0.8	0.9	
K=0,	ARIMA(Q)	25.74	26.81	27.96	26.04	27.25	28.58	26.39	27.58	28.9	18.08
1,2	ARIMA(H)	21.57	21.85	22.14	21.63	22.01	22.28	21.62	22.05	22.39	17.87
	ARIMA(D)	20.57	20.82	21.1	20.74	21.02	21.25	20.86	21.15	21.36	17.64
	ARIMA(W)	20.19	20.4	20.61	20.4	20.61	20.76	20.53	20.75	20.86	17.63
	ARIMA(Q,H,D,W)	16.59	16.5	16.39	16.5	16.45	16.35	16.4	16.37	16.33	17.13

Table 5.7: Relative Error for filters with different orders and cutoff frequencies (PERCENT)

Normalized Mean Squared Prediction Error, using Butterworth Input Filters

Table 5.8 shows results of metric NMSPE when we use different Input Filters. Based on the results represented in table 5.8, in terms of NMSPE, the use of more complex Butterworth Input Filters does not result in any improvement.

5.4 Testing the Individual ARIMA Filters

In this section, we will experiment with different filter coefficients, for the individual ARIMA filters. First we will summarize each of these individual ARIMA filters, and

NMSPE	ARIMA Filters	Filter Order = 1 Cutoff_freQ 0.7	0.8	0.9	Filter Order = 2 Cutoff_freQ 0.7	0.8	0.9	Filter Order = 3 Cutoff_freQ 0.7	0.8	0.9	Unfiltered
K=0, 1,2	ARIMA(Q)	70.27	72.93	75.89	69.56	73.02	76.22	68.43	72.46	76.06	51.19
	ARIMA(H)	59.35	60.26	61.19	58.71	60.01	61.14	57.91	59.47	60.92	50.58
	ARIMA(D)	53.16	53.59	54.05	53.33	53.79	54.19	53.4	53.86	54.22	49.68
	ARIMA(W)	52.17	52.44	52.71	52.41	52.68	52.88	52.54	52.8	52.96	49.69
	ARIMA(Q,H, D,W)	52.54	52.67	52.76	52.17	52.48	52.67	51.7	52.19	52.58	50.02

Table 5.8: Normalized Mean Squared Prediction Error for filters with different orders and cutoff frequencies (PERCENT)

then we will present the results of the experiments. Here we give a short explanation on each individual ARIMA filter. Also, we are going to examine a variation of the ARIMA(D) filter which exploits more daily history. It is similar to ARIMA(D) filter, but instead of using just one prior day in the filter, the filter weights have been adjusted so that it uses several previous days in the filter.

Quarter-Hourly ARIMA Filter

A simple version of the ARIMA(Q) model is based on the last 15-min traffic change. Assuming something similar is going to happen at the present time, we can model the traffic rate as follows.

$$x_{t+1} = x_t + \triangle_q$$

Or:

$$x_{t+1} = x_t + \bar{x} - \bar{x}_{t-1}$$

where $\bar{x}$ is the filtered data, which by default is the low-pass filtered traffic demands, using the last traffic demands samples.

$$\bar{x} = \sum_{\tau=1}^{n} w(\tau) x_{t-(\tau-1)}$$

This filter as shown above estimates the next change in the traffic rate based on the observed change in the last quarter-hour. By changing the ARIMA(Q) filter length and weights, the changes in traffic rates observed over several previous quarter-hour intervals can be included into the estimated change in the next quarter-hour interval.

Hourly ARIMA Filter

This filter computes an estimated change in the traffic rate in the next quarter hour, based on the 4 samples in the last hour. A Weighted-Moving-Average of the changes of traffic rates over the last hour is used to predict the traffic change in the next time slot. As mentioned before, the estimated traffic change can be computed using the following equation :

$$x_{t+1} = x_t + \triangle_h$$

$$\triangle_h = \sum_{\tau=0}^{3} z_h(\tau)(\bar{x}(i,j)^{t-\tau} - \bar{x}(i,j)^{t-\tau-1})$$

where z_h is a vector of ARIMA filter weights that assigns weighs to the changes of traffic rates in the last hour. By changing the ARIMA(H) filter length and weights, the changes observed over several previous hours can be included into the estimated change in the next quarter-hour interval. To better understand the Hourly ARIMA filter, please refer to figure 5.28.

Figure 5.28 shows where the ARIMA(H) filter focuses its attention, when the

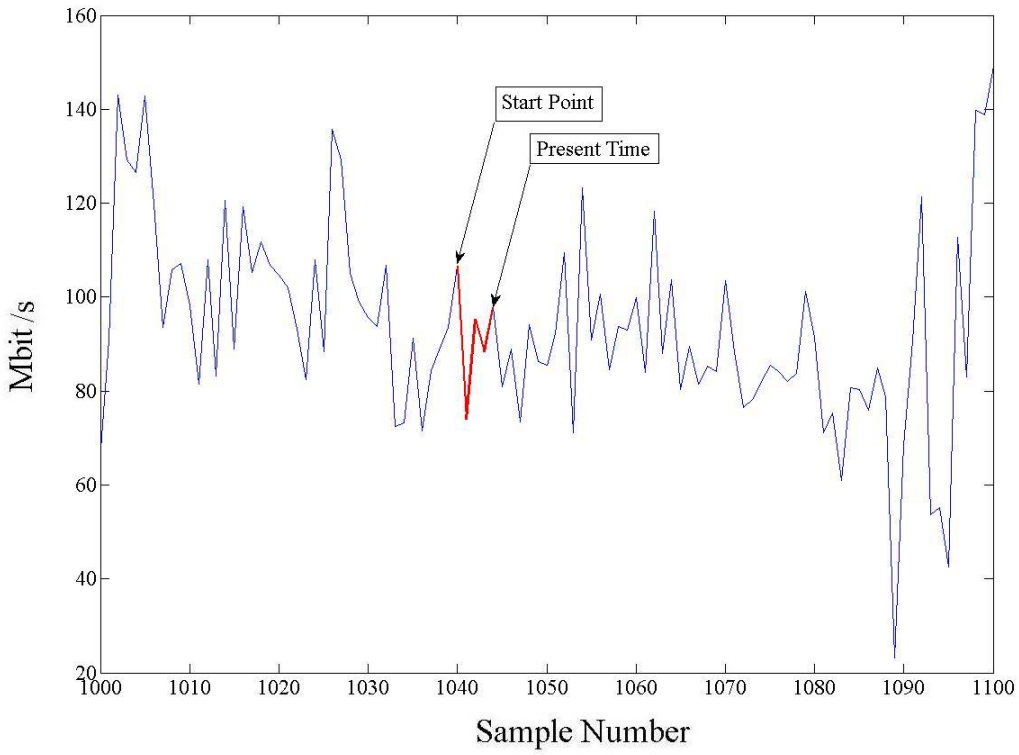

Figure 5.28: Hourly ARIMA Filter Exploits traffic trend over the last hour;in red

length of the filter vector is 4. For example, letting the arrows point to the present time, then the red line shows where this filter works on.

The Daily ARIMA Filter

The Daily ARIMA(D) filter will predict the future traffic change based on the traffic changes occurring in the same time interval in the last day. A Weighted Moving Average of the traffic differences in the same hour in the last day is used to model the traffic. By changing the ARIMA(D) filter length and weights, observations over multiple days can be included into the estimate of the change in the next quarter-hour

interval.

$$x_{t+1} = x_t + \triangle_d$$

$$\triangle_d = \sum_{\tau=0}^{3} z_d(\tau)(\bar{x}(i,j)^{t-96-\tau+1} - \bar{x}(i,j)^{t-96-\tau})$$

where z_d is the filter for averaging the past changes of the traffic in the last day.

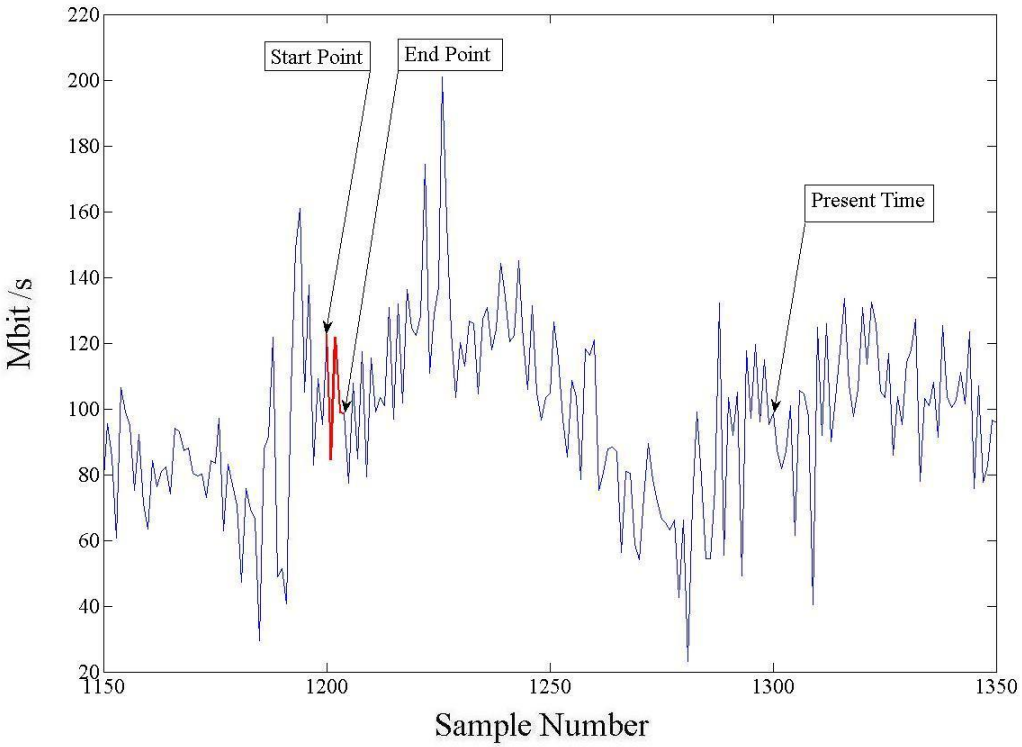

Figure 5.29: Daily ARIMA Filter Exploits traffic trend over the same time last day; in red

Figure 5.29 shows the times where the ARIMA(D) focuses on, when the filter length is 4. As an example, if the arrow points to the present time-slot t = 1300, and because in each day there are 96 samples, then the same hour in the last day will occur at time-slots 1201 to 1205 (where start point and end point arrows show).

Using a ARIMA(D) filter of length 4, the traffic observed in this period is being used to predict the change of traffic in the future interval.

The Multi-Day ARIMA(D) Filter

In this sub-section we change the single day ARIMA(D) filter length and weights, to examine the traffic changes that occur over several previous days. We call this the Multi-Day ARIMA(MD) filter, as it takes observations over multiple days to predict the next time slot traffic rate. Here we use observations taken over the last four days. The traffic rate of the next time slot can be predicted based on observations in the same hour over the last 4 days. The form of the Multi-Day ARIMA(MD) filter that considers the last 4 days is :

$$\begin{aligned} &\hat{x}_{t+1} = x_t + \triangle_{Md} \\ &\triangle_{md} = \sum_{md=1}^{4} H(md) \sum_{\tau=0}^{3} z_{md}(\tau)(\bar{x}(i,j)^{t-24*4*md-\tau+1} - \bar{x}(i,j)^{t-24*4*md-\tau}) \end{aligned} \tag{5.81}$$

where $md = 4$ implies that this filter exploits the information of the last 4 days. z_{md} is the vector of filter weights similar to the previous ARIMA filters. Basically this filter first exploits the information of the traffic changes of the same hour in the last 4 days. It then computes a final estimate by weighted averaging of the results of each individual day. The weighted averaging is performed with $H(md)$ filter. $H(md)$ is a 4 element vector to merge the results of the 4 days.

After predicting the next traffic bandwidth change, we can provide bandwidth

with the following equations:

$$\sigma_{md}^2 = Var(w(t-8,t,1)) \tag{5.82}$$

$$\dot{x}^{t+1} = x^{t+1} + k_{md}\sigma_{md} \tag{5.83}$$

Figure 5.30 shows the intervals where this Multi-day ARIMA(MD) filter is working on. In this figure, let the present time be time-slot 900. The red parts of the traffic trend are used in this ARIMA(MD) filter. The previous equations show a 2 step computation, to aid in the explanation. This filter first computes an estimated change due to the traffic trend observed in the same hour over the last 4 days separately (each of the red lines in the picture). Then, it merges the 4 estimated changes together to find the ultimate prediction. This two-step computation can be rearranged to yield a single ARIMA(MD) filter with the appropriate filter length and filter coefficients. Observations of many different hours over many different days can be included in the estimate by changing the ARIMA(MD) filter length and weights.

The Weekly ARIMA Filter

The weekly ARIMA filter exploits the information observed in the traffic trend over the last week. As explained in chapter 4 the weighted moving average of the traffic differences of the same time last week can be used for prediction. By changing the ARIMA(W) filter length and weights, observations over multiple days over multiple weeks can be included into the estimate of the change in the next quarter-hour interval.

$$x_{t+1} = x_t + \triangle_w$$

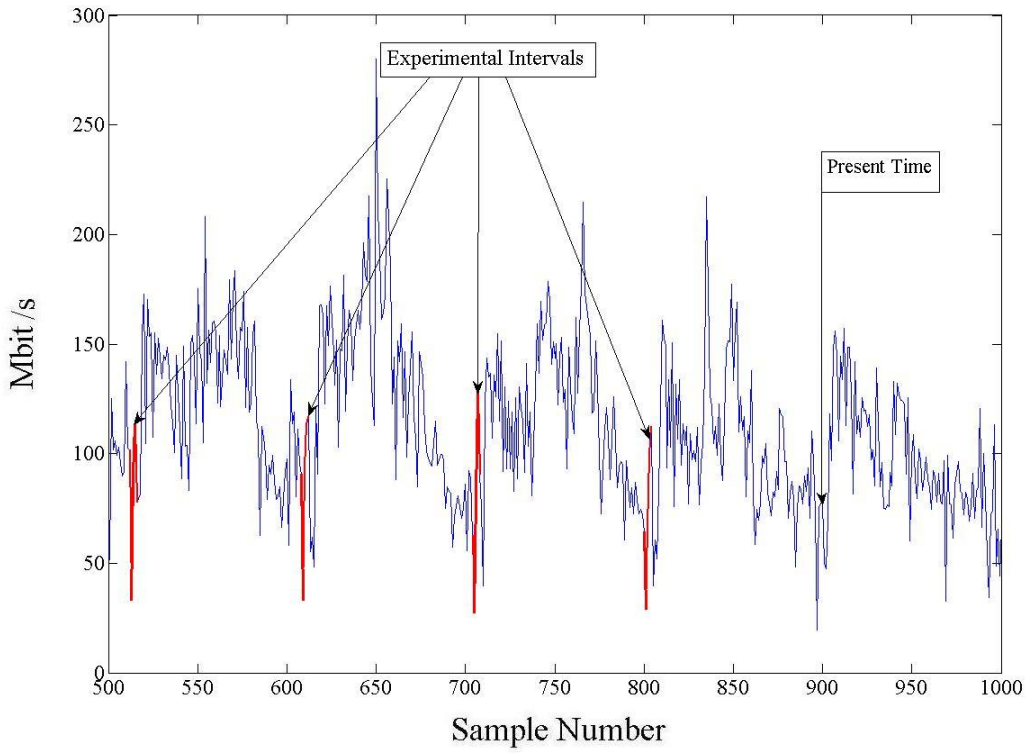

Figure 5.30: Multi-Daily ARIMA Filter Exploits traffic trend over the same time last days; in red

$$\triangle_w = \sum_{\tau=0}^{3} z_w(\tau)(\bar{x}(i,j)^{t-96\times7-\tau+1} - \bar{x}(i,j)^{t-96\times7-\tau})$$

where z_w is a vector of filter weights.

Figure 5.31 shows the window that the ARIMA(W) filter focuses on to predict the future change of the traffic, using a filter of length 4. However, by changing the ARIMA(W) filter length and weights, we can run the experiment for different situations.

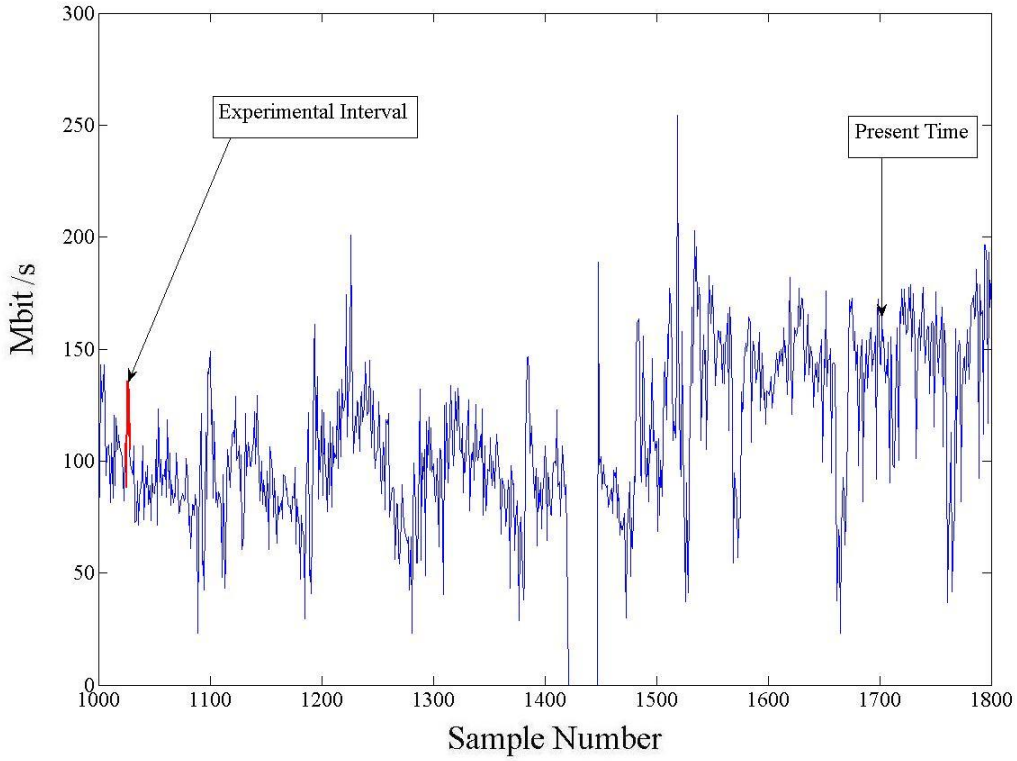

Figure 5.31: Weekly ARIMA Filter Exploits traffic trend over the same time last week;in red

5.4.1 Applying Different ARIMA filters

In the last section, we described the terminology of the ARIMA filters. We observe that the ARIMA filter vector z is a vector that tells the algorithm what samples of the traffic rates will influence the next prediction. This vector also specifies the different weights to be associated with those samples.

In this sub-section, we will change the weights and length of the individual ARIMA filters to see the result. We will test two situations: In case (a) no Input Filtering is used. In case(b) simple Input Filter is used.

In our previous experiments, the ARIMA filter we used, was a simple 4-element

filter as follows:

$$z_h = [1/2, 1/4, 1/8, 1/8] \tag{5.84}$$

For the individual ARIMA filters in our experiments, we can use a technique called Exponential Smoothing. The ARIMA filters can use exponentially decreasing weights to compute a moving average of the traffic changes. This exponential smoothing is summarized next.

Exponential Smoothing

Authors in [34] show how to use the exponential smoothing for predicting. The general form of the exponential smoothing is shown as:

$$\begin{aligned} s_1 &= q_0 \\ s_t &= \alpha q_{t-1} + (1-\alpha) s_{t-1} \end{aligned} \tag{5.85}$$

Where q's are the past traffic changes, s_t is the predicted change of the traffic for the next time slot, and α is called the smoothing factor, which $0 < \alpha < 1$. s_{t-1} also, can be written as a function of last traffic changes. Therefore the predicted value of the traffic change, for the time t based on the last k observations would be:

$$s_k = \alpha q_{k-1} + \alpha(1-\alpha) q_{k-2} + \alpha(1-\alpha)^2 q_{k-3} + \cdots + (1-\alpha)^{k-1} q_0 \tag{5.86}$$

That is the reason this approach is called Exponential. q_0 is the first observed traffic change. Here the length of the filter is k. A small value of k will have a less smoothing effect, and be more responsive to the recent changes of the traffic demands, while a

large value of k will bring in more samples and consequently have a greater smoothing effect.

Another parameter that we need to determine is the smoothing factor, α. Values of α close to 1 have less smoothing effect and put more emphasis on the recent changes. On the other hand, values of α close to zero have more smoothing effect by assigning larger weights to the distant past samples.

In this section, we run experiments on three different smoothing factors α, for two different filter lengths k. Table 5.9 summarizes these wights. The first set is the default set.

	Weights	α	K
Z_1	[1/4, 1/4, 1/4, 1/4]	-	-
Z_2	[3/4, 3/16, 3/64, 1/64]	3/4	4
Z_3	[1/2, 1/4, 1/8, 1/8]	1/2	4
Z_4	[1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/128]	1/2	8
Z_5	[1/4, 1/4*(3/4), 1/4*(3/4)^2, 1/4*(3/4)^3, 1/4*(3/4)^4, 1/4*(3/4)^5, 1/4*(3/4)^6, (3/4)^7]	1/4	8

Table 5.9: Different ARIMA Filtering Weights

In the following, the results of different choices of ARIMA filters are shown in 2 sets of tables. In the first table, the traffic demands use Input Filters. The Input Filter we used, is a Butterworth low-pass filter with order of 1 and relative cutoff frequency of 0.7. The second table shows the results when traffic demands do not use an Input Filter.

We first examine with the Daily ARIMA(D) filter and then we show the results for the Multi-Day ARIMA(D) filters.

Mean Satisfied Bandwidth

Tables 5.10 and 5.11 show the MSB for different sets of ARIMA filters. In 5.10 the Input Filter is a digital Butterworth filter with relative cutoff frequency 0.7 and filter order of 1.

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
MSB	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	84.64	84.64	84.64	84.64	84.64
	ARIMA(H)	86.59	85.33	86	85.98	86.83
	ARIMA(D)	87.03	86	86.61	86.59	87.26
	ARIMA(W)	87.01	86.14	86.68	86.64	87.24
	ARIMA(Q,H,D,W)	89.09	89.12	88.95	88.94	88.81
K=1	ARIMA(Q)	92.53	92.53	92.53	92.53	92.53
	ARIMA(H)	94.23	93.28	93.95	93.96	94.3
	ARIMA(D)	93.96	93.15	93.71	93.71	94.18
	ARIMA(W)	93.93	93.19	93.71	93.7	94.15
	ARIMA(Q,H,D,W)	94.77	94.79	94.73	94.73	94.7
K=2	ARIMA(Q)	96.05	96.05	96.05	96.05	96.05
	ARIMA(H)	96.7	96.37	96.59	96.59	96.7
	ARIMA(D)	96.43	95.92	96.28	96.28	96.57
	ARIMA(W)	96.4	95.91	96.27	96.27	96.54
	ARIMA(Q,H,D,W)	96.87	96.89	96.86	96.86	96.84

Table 5.10: Different ARIMA filters, with Input Filter of order 1 and relative cutoff frequency of 0.7 (PERCENT)

Referring to both tables, it appears that the MSB is slightly maximized when no Input Filters are used. This result is consistent with the previous sections, which showed no improvement when Input Filtering was used. Referring to the second table (when no Input Filtering is used), the MSB is slightly maximized when the 1st or 2nd ARIMA filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are

MSB	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	83.72	83.72	83.72	83.72	83.72
	ARIMA(H)	86.44	84.67	85.62	85.59	86.65
	ARIMA(D)	86.96	85.36	86.28	86.25	87.13
	ARIMA(W)	86.97	85.68	86.45	86.41	87.15
	ARIMA(Q,H,D,W)	89.22	89.29	89.1	89.1	88.96
K=1	ARIMA(Q)	91.57	91.57	91.57	91.57	91.57
	ARIMA(H)	94.15	92.79	93.77	93.77	94.3
	ARIMA(D)	93.91	92.67	93.49	93.48	94.1
	ARIMA(W)	93.89	92.85	93.57	93.56	94.1
	ARIMA(Q,H,D,W)	94.86	94.89	94.82	94.82	94.8
K=2	ARIMA(Q)	95.3	95.3	95.3	95.3	95.3
	ARIMA(H)	96.67	96.16	96.52	96.52	96.72
	ARIMA(D)	96.4	95.6	96.15	96.15	96.52
	ARIMA(W)	96.38	95.67	96.19	96.18	96.51
	ARIMA(Q,H,D,W)	96.91	96.93	96.9	96.9	96.88

Table 5.11: Different ARIMA filters, with Unfiltered Traffic Demands(PERCENT)

necessary, and that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good performance.

Mean Excess Bandwidth

Tables 5.12 and 5.13 show the Mean Excess Bandwidth (MEB), for cases where Input Filters are used and not used. Referring to both tables, it appears that the MEB is slightly minimized when no Input Filters are used. This result is consistent with the previous sections, which showed no improvement when Input Filtering was used. Referring to the second table (when no Input Filtering is used), the MEB is slightly minimized when the 2nd, 3rd and 4th ARIMA filters are used. The differences

however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good performance.

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
MEB	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	17.83	17.83	17.83	17.83	17.83
	ARIMA(H)	14.13	16.29	14.96	14.98	13.59
	ARIMA(D)	13.65	15.59	14.29	14.29	13.1
	ARIMA(W)	13.91	15.71	14.45	14.42	13.26
	ARIMA(Q,H,D,W)	11.62	11.78	11.73	11.73	11.74
K=1	ARIMA(Q)	38.38	38.38	38.38	38.38	38.38
	ARIMA(H)	36.26	37.3	36.54	36.53	36.21
	ARIMA(D)	36.94	38.33	37.32	37.31	36.53
	ARIMA(W)	37.14	38.51	37.47	37.42	36.63
	ARIMA(Q,H,D,W)	36.24	36.23	36.17	36.15	36.19
K=2	ARIMA(Q)	65.01	65.01	65.01	65.01	65.01
	ARIMA(H)	64.37	64.68	64.47	64.47	64.37
	ARIMA(D)	64.97	65.93	65.21	65.2	64.68
	ARIMA(W)	65.13	66.09	65.33	65.27	64.73
	ARIMA(Q,H,D,W)	64.69	64.65	64.58	64.56	64.6

Table 5.12: Different ARIMA filters, with Input Filter of order 1 and relative cutoff frequency of 0.7(PERCENT)

Relative Error

Tables 5.14 and 5.15 show the result for the relative error for different sets of ARIMA filters. Referring to both tables, it appears that the Relative Error (RE) is slightly minimized when no Input Filters are used. Referring to the second table (when no

MEB	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	19.53	19.53	19.53	19.53	19.53
	ARIMA(H)	14.35	17.28	15.46	15.5	13.82
	ARIMA(D)	13.79	16.71	14.79	14.81	13.31
	ARIMA(W)	14.02	16.65	14.85	14.84	13.42
	ARIMA(Q,H,D,W)	11.52	11.48	11.48	11.48	11.55
K=1	ARIMA(Q)	40.1	40.1	40.1	40.1	40.1
	ARIMA(H)	36.34	37.99	36.73	36.73	36.19
	ARIMA(D)	37.03	39.14	37.63	37.63	36.65
	ARIMA(W)	37.22	39.21	37.73	37.69	36.73
	ARIMA(Q,H,D,W)	36.2	35.96	35.98	35.96	36.05
K=2	ARIMA(Q)	66.02	66.02	66.02	66.02	66.02
	ARIMA(H)	64.39	64.9	64.54	64.53	64.34
	ARIMA(D)	65.03	66.5	65.42	65.41	64.75
	ARIMA(W)	65.18	66.62	65.51	65.46	64.8
	ARIMA(Q,H,D,W)	64.68	64.39	64.41	64.39	64.49

Table 5.13: Different ARIMA filters, with Unfiltered Traffic Demands(PERCENT)

Input Filtering is used), the RE is slightly minimized when the 1st ARIMA filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

Normalized Mean Square Error

Tables 5.16 and 5.17 show the result for the Normalized Mean Squared Prediction Error for different weight sets of ARIMA filters.

Referring to both tables, it appears that the NMSPE is slightly minimized when

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
RE	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	25.74	25.74	25.74	25.74	25.74
	ARIMA(H)	20.35	23.5	21.57	21.6	19.59
	ARIMA(D)	19.65	22.43	20.57	20.57	18.87
	ARIMA(W)	19.46	21.84	20.19	20.2	18.72
	ARIMA(Q,H,D,W)	16.34	16.54	16.59	16.6	16.63

Table 5.14: Relative Error for Different ARIMA filters, with Input Filter of order 1 and relative cutoff frequency of 0.7(PERCENT)

		Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
RE	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	28.23	28.23	28.23	28.23	28.23
	ARIMA(H)	20.67	24.96	22.31	22.36	19.93
	ARIMA(D)	19.83	24.05	21.29	21.31	19.17
	ARIMA(W)	19.6	23.02	20.67	20.71	18.92
	ARIMA(Q,H,D,W)	16.16	16.21	16.31	16.3	16.39

Table 5.15: Relative Error for Different ARIMA filters, with Unfiltered Traffic Demands(PERCENT)

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
NMSPE	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	70.27	70.27	70.27	70.27	70.27
	ARIMA(H)	55.66	64.57	59.35	59.39	54.37
	ARIMA(D)	51.72	56.58	53.16	53.14	50.77
	ARIMA(W)	51.33	54.65	52.17	52.18	50.51
	ARIMA(Q,H,D,W)	51.76	52.93	52.54	52.56	51.88

Table 5.16: Different ARIMA filters, with Input Filter of order 1 and relative cutoff frequency of 0.7(PERCENT)

NMSPE	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	79.59	79.59	79.59	79.59	79.59
	ARIMA(H)	57.01	69.71	62.09	62.15	55.56
	ARIMA(D)	52.12	60.32	54.54	54.54	51.18
	ARIMA(W)	51.5	56.82	52.87	52.91	50.68
	ARIMA(Q,H,D,W)	51.94	53.22	52.83	52.84	52.03

Table 5.17: Different ARIMA filters, with Unfiltered Traffic Demands(PERCENT)

Input Filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

The Multi-Day ARIMA(MD) Filter

In this section, we evaluate the Multi-Day ARIMA(MD) filter, that exploits the traffic trend over last 4 days. The traffic rate of the next time slot can be predicted based on the observations in the same hour in the last 4 days. Therefore, the length and weights of the original ARIMA(MD) filter are modified, to implement this model. We run the experiments for traffic demands with 2 cases, (a) no Input Filtering, and (b) with Butterworth Input Filters.

Mean Satisfied Bandwidth, using the Multi-day ARIMA(MD) filter

Table 5.19 shows the MSB for the Multi-day ARIMA(MD) filter, when no Input Filtering is used. The table 5.18 shows the MSB for the Multi-day ARIMA(MD)

filter, when Input Filtering is used. The Input Filter is a Butterworth digital filter with relative cutoff frequency of 0.7 and order of 1.

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
MSB	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	84.64	84.64	84.64	84.64	84.64
	ARIMA(H)	86.59	85.33	86	85.98	86.83
	ARIMA(MD)	87.44	86.83	87.21	87.2	87.58
	ARIMA(W)	87.01	86.14	86.68	86.64	87.24
	ARIMA(Q,H,D,W)	88.99	89.01	88.86	88.84	88.75
K=1	ARIMA(Q)	92.53	92.53	92.53	92.53	92.53
	ARIMA(H)	94.23	93.28	93.95	93.96	94.3
	ARIMA(MD)	94.27	93.85	94.14	94.14	94.38
	ARIMA(W)	93.93	93.19	93.71	93.7	94.15
	ARIMA(Q,H,D,W)	94.74	96.05	94.7	94.7	94.68
K=2	ARIMA(Q)	96.05	96.05	96.05	96.05	96.05
	ARIMA(H)	96.7	96.37	96.59	96.59	96.7
	ARIMA(MD)	96.61	95.91	96.53	96.53	96.68
	ARIMA(W)	96.4	95.91	96.27	96.27	96.54
	ARIMA(Q,H,D,W)	96.86	96.05	96.85	96.85	96.83

Table 5.18: Using Multiday ARIMA filter, with Input Filter of order 1 and relative cutoff frequency of 0.7

Referring to both tables, it appears that the MSB is slightly maximized when no Input Filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary.

We can now compare the combined ARIMA filters, using the original single-day ARIMA(D) model, tables 5.10 and 5.11, and the multi-day ARIMA(MD) model, tables 5.18 and 5.19. The Multi-day ARIMA(MD) filter works better than the single-day ARIMA filter, but the combined ARIMA filters using the single-day ARIMA(D)

MSB	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	83.72	83.72	83.72	83.72	83.72
	ARIMA(H)	86.44	84.67	85.62	85.59	86.65
	ARIMA(MD)	87.4	86.43	87.4	87	87.51
	ARIMA(W)	86.97	85.68	86.45	86.41	87.15
	ARIMA(Q,H,D,W)	89.13	89.2	89.02	89.01	88.9
K=1	ARIMA(Q)	91.57	91.57	91.57	91.57	91.57
	ARIMA(H)	94.15	92.79	93.77	93.77	94.3
	ARIMA(MD)	94.25	93.57	94.25	94.03	94.34
	ARIMA(W)	93.89	92.85	93.57	93.56	94.1
	ARIMA(Q,H,D,W)	94.83	94.85	94.79	94.79	94.78
K=2	ARIMA(Q)	95.3	95.3	95.3	95.3	95.3
	ARIMA(H)	96.67	96.16	96.52	96.52	96.72
	ARIMA(MD)	96.6	96.19	96.47	96.47	96.65
	ARIMA(W)	96.38	95.67	96.19	96.18	96.51
	ARIMA(Q,H,D,W)	96.9	96.92	96.89	96.89	96.88

Table 5.19: Using Multiday ARIMA filter, with Unfiltered Traffic Demands

model has a slightly better MSB. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

Mean Excess Bandwidth, using the Multi-day ARIMA(MD) filter

Tables 5.20 and 5.21 show the Mean Excess Bandwidth when we use Multi-daily ARIMA(MD) filter.

Referring to both tables, it appears that the MEB is slightly minimized when no Input Filters are used. The differences however are typically a few hundreds of

MEB	ARIMA Filters	Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	17.83	17.83	17.83	17.83	17.83
	ARIMA(H)	14.13	16.29	14.96	14.98	13.59
	ARIMA(MD)	12.86	13.94	13.2	13.21	12.55
	ARIMA(W)	13.91	15.71	14.45	14.42	13.26
	ARIMA(Q,H,D,W)	11.72	11.91	11.86	11.86	11.83
K=1	ARIMA(Q)	38.38	38.38	38.38	38.38	38.38
	ARIMA(H)	36.26	37.3	36.54	36.53	36.21
	ARIMA(MD)	36.37	37.03	36.54	36.54	36.17
	ARIMA(W)	37.14	38.51	37.47	37.42	36.63
	ARIMA(Q,H,D,W)	36.28	36.31	36.25	36.23	36.24
K=2	ARIMA(Q)	65.01	65.01	65.01	65.01	65.01
	ARIMA(H)	64.37	64.68	64.47	64.47	64.37
	ARIMA(MD)	64.56	65.93	64.67	64.67	64.42
	ARIMA(W)	65.13	66.09	65.33	65.27	64.73
	ARIMA(Q,H,D,W)	64.71	64.65	64.64	64.63	64.64

Table 5.20: Using Multiday ARIMA filter, with Input Filter of order 1 and relative cutoff frequency of 0.7

a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

We can now compare the combined ARIMA filters, using the original single-day ARIMA(D) model, tables 5.12 and 5.13, and the Multi-day ARIMA(MD) model, tables 5.20 and 5.21. The Multi-Day ARIMA(MD) filter works slightly better than the single-day ARIMA filter, but the combined ARIMA filters using the single-day ARIMA(D) model has a slightly better MEB.

MEB	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0	ARIMA(Q)	19.53	19.53	19.53	19.53	19.53
	ARIMA(H)	14.35	17.28	15.46	15.5	13.82
	ARIMA(MD)	12.94	14.58	12.94	13.49	12.66
	ARIMA(W)	14.02	16.65	14.85	14.84	13.42
	ARIMA(Q,H,D,W)	11.61	11.61	11.67	11.62	11.66
K=1	ARIMA(Q)	40.1	40.1	40.1	40.1	40.1
	ARIMA(H)	36.34	37.99	36.73	36.73	36.19
	ARIMA(MD)	36.41	37.48	36.41	36.7	36.23
	ARIMA(W)	37.22	39.21	37.73	37.69	36.73
	ARIMA(Q,H,D,W)	36.23	36.07	36.17	36.06	36.12
K=2	ARIMA(Q)	66.02	66.02	66.02	66.02	66.02
	ARIMA(H)	64.39	64.9	64.54	64.53	64.34
	ARIMA(MD)	64.59	65.29	64.77	64.76	64.46
	ARIMA(W)	65.18	66.62	65.51	65.46	64.8
	ARIMA(Q,H,D,W)	64.7	64.47	64.49	64.48	64.55

Table 5.21: Using Multiday ARIMA filter, with unfiltered traffic demands

Relative Error, using the Multi-day ARIMA(MD) filter

Tables 5.22 and 5.23 represent the result of the Relative Error for using Multi-Daily ARIMA(D) filter.

Referring to both tables, it appears that the RE is slightly minimized when no Input Filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters.

We can now compare the combined ARIMA filters, using the original single-day ARIMA(D) model, table 5.14 and 5.15, and the Multi-day ARIMA(MD) model, tables 5.22 and 5.23. The Multi-day ARIMA(MD) works better in terms of prediction error, however the combined ARIMA filters using the single-day ARIMA(D) model has a

RE	ARIMA Filters	Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	25.74	25.74	25.74	25.74	25.74
	ARIMA(H)	20.35	23.5	21.57	21.6	19.59
	ARIMA(MD)	18.49	20.02	18.98	18.99	18.07
	ARIMA(W)	19.46	21.84	20.19	20.2	18.72
	ARIMA(Q,H,D,W)	16.49	16.7	16.74	16.75	16.74

Table 5.22: Using Multiday ARIMA filter, with Input Filter of order 1 and relative cutoff frequency of 0.7

RE	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	28.23	28.23	28.23	28.23	28.23
	ARIMA(H)	20.67	24.96	22.31	22.36	19.93
	ARIMA(MD)	18.59	20.92	19.38	19.4	18.22
	ARIMA(W)	19.6	23.02	20.67	20.71	18.92
	ARIMA(Q,H,D,W)	16.3	16.35	16.47	16.47	16.51

Table 5.23: Different ARIMA filters, with Unfiltered Traffic Demands

slightly better RE. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

Normalized Mean Square Prediction Error, using the Multi-Day ARIMA(MD) filter

Tables 5.24 and 5.25 show the results of the NMSPE metric when we use Multi-Day ARIMA(D) filter.

		Butterworth Filter Used Filter_order = 1 ; Cutoff FreQ = 0.7				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
NMSPE	ARIMA Filters	Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	70.27	70.27	70.27	70.27	70.27
	ARIMA(H)	55.66	64.57	59.35	59.39	54.37
	ARIMA(MD)	50.23	51.62	50.62	50.62	49.91
	ARIMA(W)	51.33	54.65	52.17	52.18	50.51
	ARIMA(Q,H,D,W)	51.85	53.06	52.65	52.67	51.94

Table 5.24: Using Multiday ARIMA filter, with Unfiltered Traffic Demands

Referring to both tables, it appears that the NMSPE is slightly minimized when Input Filters are used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters.

We can now compare the combined ARIMA filters, using the single-day ARIMA(D) model, tables 5.16 and 5.17 and the Multi-Day ARIMA(MD) model, tables 5.24

NMSPE	ARIMA Filters	Unfiltered Data				
		ARIMA Filter Size = 4			ARIMA Filter Size = 8	
		Z_set(1)	Z_set(2)	Z_set(3)	Z_set(4)	Z_set(5)
K=0,1,2	ARIMA(Q)	79.59	79.59	79.59	79.59	79.59
	ARIMA(H)	57.01	69.71	62.09	62.15	55.56
	ARIMA(MD)	50.32	52.78	50.32	51.01	50.01
	ARIMA(W)	51.5	56.82	52.87	52.91	50.68
	ARIMA(Q,H,D,W)	52.03	53.33	52.92	52.94	52.09

Table 5.25: Different ARIMA filters, with Unfiltered Traffic Demands

and 5.25. Multi-Day ARIMA filter individually performs better than the single-day ARIMA filter, however the combined ARIMA filters using the single-day ARIMA(D) model has a slightly better NMSPE. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input Filters are necessary, and once again that fairly simple filters can be used for the individual ARIMA filters. The combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

5.5 Conclusion of Chapter 5

In chapter 5, we explored the robustness of the model described in chapter 4, to changes in filter parameters. Several different metrics to evaluate the model were reviewed. As described in chapter 4, the model consists of several individual ARIMA filters, each operating at a different time-scale. All the models can be combined as described in chapter 4, to present a combined prediction. To evaluate the models, we use real Traffic Demand matrices measured every 15 minutes in the 'Giant' European IP Backbone Network as the input for our model.

Two types of filters are used in the combined model, and each type of filter was adjusted to measure the robustness of the model. First, several different *'Input Filter'* can be used to low-pass filter all the measured traffic demand matrices, to remove short-term transients. Second, the filter parameters for each individual ARIMA filter can be adjusted.

In a first set of experiments, several variations of Input Filters were used, including no Input Filtering, some very simple equal-weight Input Filters, and more complex Butterworth low-pass filters. It is concluded that the combined ARIMA filter consistently outperforms the individual ARIMA filters, over several different performance metrics. It is also concluded that there is no noticeable improvement when Input Filtering was used. The differences however are typically a few hundreds of a percent, which is insignificant. These results suggest that no Input filters are necessary, and that the combined ARIMA filter is consistently better than the individual ARIMA filters. The combined ARIMA filter consistently yields very good estimates.

In fact in traditional ARIMA models, innovations or shocks to the systems, are considered as noise. In our work, these innovations stand for the traffic changes over time. In our model these innovation are all traffic measurements and they are precise and exact. Therefore we cannot count them as noise. By filtering out the high frequencies, the combined ARIMA(Q,H,D,W) model becomes less accurate, which proves we are losing valid information.

In a second set of experiments, the filter lengths and weights used in the individual ARIMA filters ere varied. It is concluded once again that the combined ARIMA filter consistently outperforms the individual ARIMA filters, over several different performance metrics. It is also concluded that there is no noticeable improvement when

the individual ARIMA filters were varied. The differences between various individual ARIMA filters are typically a few hundreds of a percent, which is insignificant. The combined ARIMA filter consistently yields very good estimates.

In summary, the experiments in Chapter 5 demonstrate that the combined ARIMA filter seems to be very robust with respect to the choice of the individual ARIMA filter lengths and weights, and it consistently yields very good estimates.

Chapter 6

Conclusion

In this thesis, a recently proposed algorithm for predicting traffic demands in a real IP network has been studied [37]. This algorithm uses multiple ARIMA filters, each working on different time scales, to estimate future traffic demands. Network traffic has a periodic characteristic which can be viewed on multiple time-scales, i.e., quarter hours, hours, days and weeks. Each ARMIA filter has an associated weight, and the weights change dynamically according to the accuracy of the filter in predicting the traffic. The final traffic estimate is the weighted average of the individual estimates of each of these filters. The weights are updated dynamically, based on the most recent estimate accuracy of each filter.

One of the main goals of this thesis is to evaluate the robustness of the proposed model. To evaluate the robustness, the filter coefficients for each ARIMA filter were varied, and the accuracy of the model was evaluated. Extensive experimental results were collected to evaluate the robustness of the model. To test the algorithm, real traffic measurements reported every 15 minutes for Geant European IP Backbone Network over a period of one month is used. To eliminate the short-term transient

changes of the traffic over time, different low-pass filters have been implemented to pre-process all the traffic measurements. The output of these low-pass filters was then processed by the proposed algorithm, using multiple ARIMA filters. Our extensive experimental results indicate that the prediction model is very robust. The use of different low-pass filters to pre-process the traffic measurements did not have a significant effect on accuracy. Variations of each ARIMA filter's coefficients did not have a significant effect on accuracy. In all our experiments, the proposed algorithm can predict short-term future traffic demands with accuracies of $\approx 90\%$ or more.

We summarize a recently proposed *Autonomic Future Internet* network model, which can use the proposed traffic estimation model [36] and [29]. We summarize how the final traffic estimate can be used in an autonomic controller of a Future-Internet router to provision bandwidth for the future traffic demands. In the Future-Internet model proposed in [36] and [29], two different traffic classes are defined, the traditional *Best-Effort* traffic class, and a new *Smooth-QoS* traffic class. The Best-Effort traffic class consists of traditional bursty Internet traffic. The Smooth-QoS traffic class consists of relatively low-jitter traffic flows, which typically have been smoothened using a token-bucket based traffic shaper at the source.

In the proposed Future Internet network, it is desirable to provision bandwidth for Smooth-QoS traffic class, to allow for deterministic end-to-end QoS guarantees [36] and [29]. The proposed traffic estimation algorithm can be used to estimate future traffic demands for the Smooth-QoS traffic class in the next 15 minute interval, based upon the past history. The traffic estimates can then be used to provision bandwidth for the Smooth-QoS traffic class, in anticipation of their arrival. Our extensive experimental results illustrate that the proposed traffic estimation algorithm

can typically satisfy up to 95% of the future Smooth-QoS traffic demands. Bandwidth that is provisioned for the Smooth-QoS class, but remains unused, can be used by the Best-Effort class, so there is no penalty for provisioning a small excess bandwidth for the Smooth-QoS class.

Bibliography

[1] S. M. Pandit, S. Wu (1983). Time series and system analysis, with applications. New York: Wiley (1983).

[2] Ruey S. Tsay (2005). Analaysis of Financial Time Series. A JOHN WILEY and SONS, INC., PUBLICATION.

[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel (1994). Time Series Analysis: Forecasting and Control. 3rd ed. Upper Saddle River, NJ: Prentice-Hall.

[4] T. Bollerslev, R. F. Engle, D. B. Nelson (1994). ARCH models. Handbook of Econometrics. Amsterdam, Netherlands: North-Holland,vol. 4, pp. 29613038.

[5] A. K. Bera, M. L. Higgins (1993). ARCH models: Properties, estimation and testing. J. Economic Surveys, vol. 7, no. 4, pp. 305362.

[6] I. Juva (2005). Traffic Matrix Estimation . HELSINKI UNIVERSITY OF TECHNOLOGY, Department of Electrical and Communications Engineering, Networking Laboratory.

[7] B. Jennings, S. Van der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, J. Strassner (2007). Towards autonomic management of communications networks. Communications Magazine, IEEE.

[8] A. JAstorga, J. Serrat, W.K. Chai, L. Mamatas, A. Galis, S. Clayman, A. Cheniour, L. Lefevre, O. Mornard, A. Fischer, A. Paler, H. de Meer (2010). Platforms and Software Systems for an Autonomic Internet. IEEE Global Telecommunications Conference (GLOBECOM).

[9] W. E. , M. S. Taqq, W. Willinger, D. V. Wilson (1993). On the Self-Similar Nature of Ethernet Traffic. Proc. of ACM SIGCOMM 93, pp. 183-193.

[10] J. Kowalski, B. Warfield (1995).Modeling traffic demand between nodes in a telecommunications network. ATNAC 95.

[11] Y. Zhang, M. Roughan, N. Duffield, A. Greenberg (2003). Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads. ACM Sigmetrics.

[12] Y. Vardi (1996). Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data. the American Statistical Association., pages 365377.

[13] S. Vaton, J.S. Bedo, A. Gravey (2005). Advanced methods for the estimation of the Origin Destination traffic matrix. Revue du 25me anniversaire du GERAD.

[14] O. Goldschmidt (2000). ISP Backbone Traffic Inference Methods to Support Traffic Engineering. In Internet Statistics and Metrics Analysis Workshop, San Diego.

[15] J. Cao, D. Davis, S. Vander Weil, B. Yu (2000). Time-Varying Network Tomography. J. of the American Statistical Association.

[16] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, C. Diot (2002). Traffic matrix estimation: Existing techniques and new solutions. in SIGCOMM02, Pittsburg, USA.

[17] C. Tebaldi, M. West (1998). Bayesian Inference of Network Traffic Using Link Count Data. Journal of the American Statistical Association., pages 557573.

[18] A. Medina, K. Salamatian, N. Taft, I. Matta, Y. Tsang, C. Diot (2003). On the Convergence of Statistical Techniques for Inferring Network Traffic Demands. Tecnhical Report, BUCS-2003-003

[19] A. Medina, K. Salamatian, N. Taft, I. Matta, Y. Tsang, C. Diot (2004). A Two-step Statistical Approach for Inferring Network Traffic Demands. Technical Report BUCS-2004-011 Revises BUCS-TR-2003-003

[20] A. Soule, A. Nucci, R. Cruz, E. Leonardi, N. Taft (2004). How to identify and estimate the largest traffic matrix elements in a dynamic environment. In ACM Sigmetrics, New York.

[21] A. Soule , K. Salamatian , A. Nucci , N. Taft (2005). Traffic matrix tracking using Kalman filters. ACM SIGMETRICS Performance Evaluation Review, v.33 n.3.

[22] B. Krithikaivasan, Y. Zeng, K. Deka, D. Medhi (2007). ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic. IEEE/ACM Trans. Networking, vol. 15, no. 3, pp. 683-696.

[23] F. Agharebparast,V.C.M Leung (2003). A new traffic rate estimation and monitoring algorithm for the QoS-enabled Internet. Global Telecommunications Conference, GLOBECOM '03. IEEE ,Pages 3883- 3887 vol.7

[24] Cisco. NetFlow Services Solutions Guide, July 2001.

[25] D. Jiang ,G. Hu (2009). GARCH model-based large-scale IP traffic matrix estimation. IEEE COMMUNICATIONS LETTERS, VOL. 13, NO. 1.

[26] B. Krithikaivasan, K. Deka, D. Medhi (2004). Adaptive bandwidth provisioning based on discrete temporal network measurements. IEEE INFOCOM04, Hong Kong, pp. 17861796.

[27] L.G. Roberts (2009). A Radical New Router: The Internet is Broken - Lets Fix It. IEEE Spectrum.

[28] T.H. Szymanski, D. Gilbert (2010). Provisioning Mission-Critical Telerobotic Control Systems over Internet Backbone Networks with Essentially-Perfect QoS. IEEE JSAC, Vol. 28, No. 5.

[29] T.H. Szymanski (2011). Video Multicasting in an Autonomic Future Internet with Essentially-Perfect Throughput and QoS Guarantees. 11th Int. Conf. on ITS Telecom. (NEW2AN 2011) , St. Petersburg, Russia, (and Springer Lecture Notes in Computer Science, Vol. 6869)

[30] S. Shenker (1995). Fundamental Design Issues for the Future Internet. IEEE JSAC, 11761188.

[31] S. Giordano,S. Salsano,S. Van den Berghe,G. Ventre,D. Giannakopoulos (2003). Advanced QoS Provisioning in IP networks: The European Premium IP Projects. IEEE Comm. Mag., 3036.

[32] J. Carapinha, R. Bless, C. Werle, K. Miller, V. Dobrota, A.B. Rusm, H. Grob-Lipski,H. Roessler (2010). Quality of Service in the Future Internet. In: Proc. ITU-K Kaleidoscope, India.

[33] S. Orlowski, R. Wessaly, M. Pioro, A. Tomaszewski (2009). SNDlib 1.0 - Survivable Network Design Library. Networks, Vol. 55, Issue 3, pp. 276-286.

[34] R. Hyndman, A. Koehler, J.K. Ord, R. Snyder, SpringerLink (Online service) (2008). Forecasting with Exponential Smoothing: The State Space Approach. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

[35] G. Bianchi, R. Sorrentino (2007). Electronic Filter Simulation and Design. McGraw-Hill Professional. pp. 1720. ISBN 978-0-07-149467-0.

[36] T.H. Szymanski (2011). Future Internet Video Multicasting with Essentially Perfect Resource Utilization and QoS Guarantees. IEEE/ACM International Workshop on Quality of Service (iWQoS), San Jose, USA.

[37] T.H. Szymanski, S. Behdin (2012). Traffic Provisioning in a Future Internet. IEEE ICC Workshop on the Future Internet (FutureNet V), Ottawa, Canada.

[38] A.V. Oppenheim, A.S. Willsky, I.T. Young (1983). Signals and systems. Prentice-Hall, Englewood Cliffs, New Jersey.

[39] P. Prandoni, M. Vetterli (2008). Signal Processing for Communications. EPFL Press.(Free online Textbook)