Packet-Loss Prediction Model Based on Historical Symbolic Time-Series Forecasting

by

Hooman Homayounfard

A dissertation submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the Faculty of Engineering and Information Technology

UNIVERSITY OF TECHNOLOGY, SYDNEY

October 2013

CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

Acknowledgements

During my PhD program, I have received support and encouragement from a great number of individuals. I express my deep gratitude to my principal supervisor, Dr Paul Kennedy, for his excellent guidance in writing this thesis and the papers which preceded chapters. Dr Paul's professional advice and unyielding support led me to accomplishing this task.

I would like to thank my co-supervisors Prof John Debenham, Prof Robin Braun and Prof Simeon Simoff for advising me on the work of this thesis. I should also acknowledge Prof Barry Jay, Prof Didar Zowghi, Prof Massimo Piccardi, Prof Longbing Cao, Prof Jie Lu, Dr Ante Prodan, Dr Vinod Mirchandani, Dr Priyadarsi Nanda, Dr Masoud Talebian, Andrew Litchfield, Jose Vergara, Nima Ramzani, Debi Taylor and Vahid Behbood. These are a few of many people who had a positive impact on my achievements.

I am in debt to the whole UTS community for challenging my mind over the past years. I do thank Mrs Eilagh Rurenga for polishing chapters in my thesis. I am beyond grateful for the help from Craig Shuard and Phyllis Agius, the FEIT research officers.

Dear family and friends, thank you for your incredible support. I would have been lost without you in this process. My special regard to my wife and mother, Narges and Mahin, for their sincere love, patience and support. The thesis is dedicated to my dear Leila.

Publications

1. **Hooman Homayounfard**, Paul Kennedy, and Robbin Braun, *NARGES: Prediction Model for Informed Routing in a Communications Network*, In J. Pei et al., editor, *LNAI*, volume 7818, pages 327-338. Springer Berlin Heidelberg, 2013.

2. **Hooman Homayounfard** and Paul Kennedy, *HDAX: Historical Symbolic Modelling of Delay Time Series in a Communications Network*, In P. J. Kennedy, K. Ong, and P. Christen,editors, *AusDM09*, volume 101 of CRPIT, pages 129-138, Melbourne, Australia, 2009.

Abstract

Rapid growth of Internet users and services has prompted researchers to contemplate smart models of supporting applications with the required Quality of Service (QoS). By prioritising Internet traffic and the core network more efficiently, QoS and Traffic Engineering (TE) functions can address performance issues related to emerging Internet applications. Consequently, software agents are expected to become key tools for the development of future software in distributed telecommunication environments. A major problem with the current routing mechanisms is that they generate routing tables that do not reflect the real-time state of the network and ignore factors like local congestion.

The uncertainty in making routing decisions may be reduced by using information extracted from the knowledge base for packet transmissions. Many parameters have an impact on routing decision-making such as link transmission rate, data throughput, number of hops between two communicating peer end nodes, and time of day. There are also other certain performance parameters like delay, jitter and packet-loss, which are decision factors for online QoS traffic routing.

The work of this thesis addresses the issue of defining a Data Mining (DM) model for packet switching in the communications network. In particular, the focus is on decision-making for smart routing management, which is based on the knowledge provided by DM informed agents. The main idea behind this work and related research projects is that time-series of network performance parameters, with periodical patterns, can be used as anomaly and failure detectors in the network. This project finds frequent patterns on delay and jitter time-series, which are useful in real-time packet-loss predictions.

The thesis proposes two models for approximation of delay and jitter time-series, and prediction of packet-loss time-series – namely the Historical Symbolic Delay Approximation Model (HDAX) and the Data Mining Model for Smart Routing in Communications Networks (NARGES). The models are evaluated using two kinds of datasets. The datasets for the experiments are generated using: (i) the Distributed Internet Traffic Generator (D-ITG) and (ii) the OPNET Modeller (OPNET) datasets.

HDAX forecasting module approximates current delay and jitter values based on the previous values and trends of the corresponding delay and jitter time-series. The prediction module, a Multilayer Perceptron (MLP), within the NARGES model uses the inputs obtained from HDAX. That is, the HDAX forecasted delay and jitter values are used by NARGES to estimate the future packet-loss value.

The contributions of this thesis are (i) a real time Data Mining (DM) model called HDAX; (ii) a hybrid DM model called NARGES; (iii) model evaluation with D-ITG datasets; and (iv) model evaluation with OPNET datasets.

In terms of the model results, NARGES and HDAX are evaluated with offline heterogeneous QoS traces. The results are compared to Autoregressive Moving Average (ARMA) model. HDAX model shows better speed and accuracy compared to ARMA and its forecasts are more correlated with target values than ARMA. NARGES demonstrates better correlation with target values than ARMA and more accuracy of the results, but it is slower than ARMA.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The key to growth is the introduction of higher dimensions of consciousness into our awareness,"
— Lao Tzu

Routing as the core task in all control systems, together with the admission, flow, and congestion control systems, is determining the overall network performance based on the quality and quantity of provided service. It refers to the distributed tasks of creating and utilising routing tables - one for each node within the network - that switch ingressing data packets to the optimum link in their path to the destination.

Routing tables are local databases that model and store the global network states within each node. A routing table is a feature, common to all routing protocols and algorithms. It encompasses all the required information for a node to make packet-switching decisions.

Routing tables - as the maps in packet delivery throughout the network - are dynamic and get updated by network state-based events such as node failure, link failure and congestion. However, the major issue with current routing mechanisms is that they generate routing tables that might not show the real-time state of a network and ignore factors like local congestion and packet-loss (Borges et al., 2011).

Moving data from one place to another, on the other hand, is a fundamental issue in many areas of science (Foster, 1991; Caro, 2004; Dutta, 2006). From recent developments in wireless networks to compiling weak signals from remote transmitters, various applications rely on a telecommunications network in today's world. The demand and supply of telecommunication services is a worldwide phenomenon with an exponential growth (Foster, 1991).

Rapid increases in the number of Internet users and services have prompted researchers within academia and industry to contemplate smart ways of supporting applications with the required Quality of Service (QoS) (Rankin et al., 2005). By prioritising data packets and the backbone network more dynamically, QoS and packet-switching functions can resolve performance issues with contemporary Internet applications such as real-time voice and video conferencing.

Technologies such as those based on software agents are expected to become key tools for the development of future software (Debenham and Simoff, 2007). Debenham et al. (2008) state that an effective routing mechanism and its management will be crucial to satisfactorily support diverse services in communications networks. Consequently, this type of technology may be used in distributed telecommunication environments such as mobile computing, e-commerce and routing management.

Mobiles, Internet, Local Area Networks (LAN), satellite services, and now Voice over IP (VoIP), just to mention a few significant examples, already have been and are changing the way of communicating, receiving, discovering and storing information and controlling external devices. Business associated with telecommunications is huge and the industry is incredibly active in the production of more and more robust and flexible communication models and devices (Rankin et al., 2005). The development of novel techniques and tools for agent-based autonomous routing management, such as in this thesis, offers the widest possible set of services to customers.

The main aspects of the world of contemporary telecommunications are the various types of devices, networks, and services. Modern telecommunications has a long list of network technologies and/or protocols - such as Multi-Protocol Label Switching (MPLS), Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Intermediate System To Intermediate System (IS-IS) and Interior Router Protocol (IRP) - for traffic routing throughout the network. Moreover, on top of these network and basic protocols, several different services can be offered, like best effort traffic, packet datagram, virtual circuit, guaranteed quality and fair-share. Routing policies and protocols must adapt to conflicting goals and limits raised by heterogeneous network applications and user-defined demands (Sandick and Crawley, 1997).

As discussed in Weiss (2000), building a network control and management system with the flexibility for decision-making in any context is challenging. Nevertheless, it is possible to devise open and modular architectures that can be used over a wide range of situations. Given that communications networks often present with several interplaying physical and software components and layers, it seems natural to think about the networks (and about their systems for management and control) in terms of distributed systems of multiple interacting agents (Weiss, 2000; Stone and Veloso, 2000).

Agent technology has received ever-increasing attention from telecommunications domain experts, academia, and practitioners in recent years (Caro, 2004). Agent modelling explicitly accounts for the modularity of network elements. It bypasses the client-server communication model once an agent carries its own description codes and can be used as an independent part of a globally distributed and fault-tolerant system (Hayzelden and

Bigham, 1999; Kotz and Gray, 1999; Gray et al., 2000; Tintin and Lee, 1999; Küpper and Park, 1998). Future networks will ideally be managed and controlled by societies of informed agents that will make decisions on behalf of human experts (Shoham and Tennenholtz, 1995; Moulin, 1998; Debenham et al., 2008). This thesis takes a step towards building such an informed agent.

Timely and informed decision-making based on information in the routing tables has been used to route data in a network (Miloucheva et al., 2003). The uncertainty may be decreased in making routing decisions for packet transmissions by using information extracted from a knowledge base of the routing history. Caro (2004) state that many parameters have an impact on routing decision-making. They mention parameters such as link transmission rate, data throughput, number of hops between two communicating peer end nodes, and time of day. Miloucheva et al. (2003) also list certain performance parameters like delay, jitter and packet-loss as their decision factors for online Quality of Service (QoS) traffic routing.

Data Mining (DM) techniques are being used in industrial applications and replaced the manual knowledge acquisition processes with automated knowledge discovery (KD) (Weiss et al., 1998). As Tennenhouse et al. (1997) and Caro and Dorigo (1998) discuss, applying Data Mining to telecommunications will be more practical when networks transform to really become active networks - in which a packet may carry its own built-in execution code and description information. In this regard, all the network future smart routers will be able to do (as normal status of operations) customised computations on the ingressing packets and update their routing table according to the obtained knowledge (Fayyad, 1996; Schoonderwoerd et al., 1997; Caro and Dorigo, 1998). At the moment, routers in a communications network act more like high speed packet-switching boxes (Caro and Dorigo, 1998), but in the future those boxes will be ousted by network processors. Consequently, the whole network will appear as a multiprocessor supercomputer where societies of intelligent agents will accomplish all the actions - including routing management - in virtual organisations and over the future Grid (Foster, 1991; Caro, 2004).

This thesis addresses the issue of defining a DM model for packet-loss prediction on the Internet. In particular, it focuses on decision-making for informed Internet routing management based on the knowledge provided by a data mining agent (Debenham et al., 2008; Jennings et al., 2007). The attention is restricted to real-time routing decision-making and the way data mining knowledge can be used for this purpose.

1.1 Motivation

Current routing strategies use routing tables for evaluation of the network routes. These are relying on the information provided by heuristic models such as shortest-path, least-cost, random-assignment, and round-robin (Lau and Woo, 2007). Depending on the static or dynamic nature of the network strategy, the information used for routing is reflected within routing tables that are created when the network state is updated. The issue with these strategies is that they create routing tables that might not reflect the real-time status of the network, and consequently, neglect factors like local congestion.

Adoption of contemporary routing strategies may result in a system that operates under sub-optimal network conditions such as partial network congestion. Consequently, from a network perspective, the unbalanced use of network links causes an increase in cycle time and will probably create traffic bottlenecks. Based on the issues outlined above, it is believed that the enhancement of routing management motivates a need for a smart decision-making process.

Ash (1997) states that routes stored in a routing table are not dynamic. Dynamic routing approaches are needed for enhancing resource network utilisation and reducing costs (Lau and Woo, 2007). Observing that networks are becoming more dynamic, *more heterogeneous*, *less reliable*, and *larger in scale,*" Miloucheva et al. (2003) and Jennings et al. (2007) affirm that there is a gap in the literature regarding pattern engineering and autonomous routing management tools of the communications network. To overcome the drawbacks of traditional network management methods based on centralised control of a relatively small number of managed entities, a data mining model is proposed in this thesis. The DM model will help to provide routing knowledge to a routing management module in making informed decision for data transfer over the network. Consequently, the overall benefit that will be obtained by using this approach is curtailment of flooding the network with data mining information.

1.2 Objectives and Key Tasks

This thesis aims to present a predictive model that can make informed decisions for routing in a communications network. It investigates the enhancement of routing management by taking into account the data mining knowledge of past history of routing decisions as well as QoS patterns derived from network QoS trace datasets. A smart edge router can make informed decisions for inter-domain data transfer, using the output of the proposed predictive DM model in this thesis.

The model will eventually cover multiple interconnected autonomous systems. This will require consideration of several QoS performance parameters for an effective routing management. The major tasks in the thesis are as:

- Literature review and proposal of alternative models for demonstrating the knowledge, and justifying the reasons for the research,

- Model Design to find a model for packet-loss prediction,
- Software Model Development for running experiments and performance evaluation,
- Performance Evaluation - Simulation for evaluating the model results and validating model robustness.

1.3 Research Design and Methodology

The thesis defines a model for recognising the structures and patterns in the Time-Series (TS) data sequences (QoS and/or traffic) within a communications network based on the DM models. The model uses a pattern database for further prediction of network parameters and behaviours, which are important for routing decision-making. This research encompasses studying informed decision-making for inter-domain routing in a communications network based on the knowledge obtained from the proposed predictive models.

There are certain evaluation parameters for QoS routing that define network performance such as throughput, packet delay, packet-loss, jitter, minimum hop-count and maximum reservable bandwidth. Here, it is aimed to predict link packet-loss parameter based on delay and jitter traces — in order to detect and avoid link overload in advance of occurring.

1.3.1 Data Network Scenarios

This thesis creates datasets using D-ITG traffic generator and OPNET Modeler. The network performance traces may be obtained either from real network data logs by means of a software agent or can be generated using a network modeller. The sequences of network QoS measurements represent time-series of data network statistics. As discussed in Rocha-Mier et al. (2007), OPNET modeler simulates network traffic and provides various kinds of services and applications for performance evaluation of the proposed model.

Network QoS datasets will also be used for the experiments. The datasets are adopted from several archives provided in the University of Naples Federico II (UNINA) website. Each archive contains delay, jitter, or packet-loss samples in the text format. In each text file, the first column indicates a reference time (i.e. the time passed from the first packet) and the second (which is also equal to the third) is the sample value in that time interval.

1.3.2 Proposed Data Mining Model

The conceptual framework will be a set of integrated software agents that are responsible for: (i) analysing time series of stochastic QoS data network, and (ii) predicting link parameters. The thesis defines a theoretical DM framework and implements a DM hybrid model based on it. The model uses the pattern definition, pattern matching and forecasting components both self-made and off-the-shelf implemented as a software agent with the ability to encapsulate various decision-making techniques.

In the proposed model, the DM theoretical model with a Pattern Definition (PD) module is augmented – based on the Pattern Definition Language (PDL) concept described in Miloucheva et al. (2003) – to help in pattern matching and pattern classification modules. As stated in Miloucheva et al. (2003), the number of consecutive outliers in QoS values defines the length of outlier value pattern. The method for calculation of outlier values may be applied for detection and prevention of the unexpected values of QoS performance parameters due to anomalies, such as node failure, link failure and link congestion.

The Time-series Data Mining (TSDM) model includes the following components:

1. QoS TSDB (Database of QoS Measured Time Series) - This is a data network database populated either by the OPNET Modeler generated data network, the University of Naples Federico II (UNINA) QoS archives.

2. HDAX module (Pattern Classification, Matching and Forecasting Module) - It is responsible for pattern classification, pattern matching and forecasting QoS TS values for the current period of the network lifecycle (window). Firstly, it analyses the TS sequences of QoS performance traces for past periods to construct classes of patterns. Next, it matches the measured TS subsequence(s) with the most frequently observed patterns to approximate the current pattern of the slope and approximate the current TS value. Finally, it extracts the forecasted values out of the most similar patterns to the goal pattern, and stores forecasted QoS time series into the QoS Forecasted Time-Series Database (FTSDB).

3. PDB (Pattern Database) - This is a database of composite patterns and their frequencies. PDB is used by HDAX to store patterns and their corresponding frequencies.

4. QoS FTSDB (Quality of Service Forecasted Time Series Database) - This is the database of network QoS TS forecasted by the HDAX module for current missing values.

5. MLP ANN - This is a Multilayer Perceptrons Artificial Neural Network used for predicting a TS future value. The MLP is using the current forecasted time-series values (of performance parameters) assigned to each link. In this case study, MLP architecture is used with two QoS performance parameters as the inputs, namely delay and jitter.

As suggested by Rocha-Mier et al. (2007), the MLP uses linear combination function and sigmoid activation function for the hidden and output layers. The output of the MLP ANN will be the packet-loss assigned to the link. The measured (target) QoS packet-loss TS as for training dataset of the MLP ANN.

Whilst the conceptual framework envisages a set of agents, one for each router, the thesis evaluates only one agent.

1.4 Contributions of the Thesis

In contrast with datagram packet-forwarding, virtual circuit routing technologies use TE techniques to find backup paths and to redirect the ingress packets after a resource failure. The main contribution of this thesis is to offer a data mining model for mapping delay and jitter traces to the packet-loss assigned to a link in the network. The hybrid model presented in this thesis will predict the number of the lost packets assigned to a link. The knowledge provided by the data mining model is to be used within the network router as a decision factor in redirecting the QoS packets toward a backup link (and/or path) over the Internet Protocol (IP) network scenario designed based on a test plan.

Consider an IP network that uses Border Gateway Protocol (BGP) routing scenario in which the packets come directly to the network from the attached peers of BGP routers, or from networks which pass VoIP streams from the other peers. When a resource failure occurs within the BGP routing scenario, a number of peers that are communicating directly or via other networks are dropped from accessing the network. This thesis presents a hybrid model which aims at predicting packet-loss – to improve the resilience of the network for a single link failure. The original contributions made in this thesis are

1. A real time Data Mining (DM) model called HDAX, for approximation of delay

and jitter time-series;

2. Design and Development of a hybrid DM model called NARGES, for prediction of the future packet-loss based on the approximation of current delay and jitter assigned to a link - obtained from the above DM model;

3. Evaluation of the proposed DM models with D-ITG data generated with varieties of End-to-End Paths (e2eP) and congestion levels;

4. Evaluation of the proposed DM models with OPNET data generated with three queuing policies, namely First In First Out (FIFO), Weighted Fair Queuing (WFQ), and Priority Queueing (PQ) network queuing policies.

1.5 Structure of the Thesis

The thesis is structured as follows. In Chapter 2, a literature review has been done on the key concepts needed for the accomplishment of this research. The chapter encompasses a brief review of QoS routing, introduces DM models related to the work of this thesis, and identifies a research gap. In Chapter 3, the definition and formal descriptions of the proposed models and algorithms are presented, specifically for the HDAX and NARGES models. In Chapter 4, the evaluation of the models and the results are presented. Three performance factors, namely accuracy, speed and the cross-correlation, of the algorithms are used for the evaluation of HDAX and NARGES models. Then, Friedman test is used to compute the Holm's $p - value$ and to test similarity of the algorithms performance on each of the above performance factors. The results of the models are compared to Autoregressive Moving Average (ARMA) model. Chapter 5, concludes the thesis; placing research into the broader context and highlighting several potential areas of further research.

Chapter 2

A Review of QoS Time-Series Prediction Models

"To strive, to seek, to find, and not to yield."

— Alfred, Lord Tennyson

The literature review in this thesis consists of four sections. Section 2.1 is a general overview. Section 2.2 is introductory and problem oriented, focusing on introduction of routing concepts in communications networks. It provides a brief overview of diverse routing strategies, algorithms, techniques and the issues associated with decision-making in routing. Section 2.3 introduces knowledge discovery in telecommunications and the data network scenarios. It studies certain datasets used in network data mining and the theoretical framework employed for this purpose. Section 2.4 reviews prediction methods and models for QoS Routing. Summations of the two recent and related works that focus on using intelligent agents and QoS pattern analysis for real-time data mining are also presented. Section2.5 reviews related qualitative and quantitative methods in time-series analysis and forecasting. Section 2.6 concludes the chapter.

2.1 Overview

Interactive network multimedia applications including video and audio streaming are increasingly gaining popularity on the Internet. Although the Best Effort packet switching approach is still used in current network services, the introduction of Next Generation Networks and establishment of IP networks make IP technology the most dominant protocol for packet-delivery in future networks. Diverse network resources throughout end-to-end network paths pose a challenge for multimedia audio or video applications in ensuring Service Level Agreement (SLA). Other applications that play a role in this challenge include mobile Internet access, distance learning, e-commerce, entertainment and so on (Zheng and Boyce, 2001).

Major on-line multimedia services on the Internet set User Datagram Protocol (UDP) as the transmission protocol (Postel, 1980, 1981). Unlike TCP/IP, UDP does not yield any retransmission delay for packet loss, congestion control or rate control. It is vital to design robust control mechanisms to compensate for the lack of guarantee and to improve the quality of multimedia services, under certain limitations such as restricted bandwidth, packet-loss (loss), packet-delay (delay) and delay variations (jitter) that happen in transmission time (Roychoudhuri and Al-Shaer, 2005). This thesis regards certain metrics for measurement of QoS in the network, namely delay, jitter and packet-loss.

Telecommunications' multimedia QoS is enormously affected by packet-loss within the links (Cole and Rosenbluth, 2001; Markopoulou et al., 2002; Zhang et al., 2009). Packet-loss happens when buffers overflow on a router within the end-to-end path. While UDP is not reliable in network congestion and packet-loss occurrence, the TCP/IP protocol ensures retransmission of the lost packets. However, packet retransmission is not conducive to the best real-time quality for the multimedia output. The retransmission adds a delay and this might exceed the Mouth-to-Ear (M2E) on-line audio delay of 400ms

(Roychoudhuri and Al-Shaer, 2005; ITU-T, 1993, 2003).

Although Forwarded Error Correction rectifies packet-loss, it puts bandwidth and processing overhead on the network. Therefore, high use of Forwarded Error Correction (FEC) may cause excessive bandwidth consumption when there is a challenge to balance bandwidth usage and efficient packet-loss recovery. This is counted as a motivation for research on Internet delay, jitter and packet-loss (Roychoudhuri and Al-Shaer, 2005). They also reports various Internet packet-loss factors such as packet-loss ratio, degrees of burstiness and inter-loss gap, delay and jitter patterns. Roychoudhuri and Al-Shaer (2005) states that despite the existence of comprehensive knowledge about these factors together with their correlation, there are still gaps for dynamic FEC, and models, to be able to predict and avoid packet-loss.

A substantial amount of investigation asserts the correlation between delay, jitter and the packet-loss on the Internet (Brakmo et al., 1994; Moon, 2000; Jain and Dovrolis, 2002; Paxson, 1997; Roychoudhuri and Al-Shaer, 2005; Hermanns and Schuba, 1996). Moon (2000) analyses the delay and loss measurements for TCP and UDP traffic using the loss-conditioned average delay and observing oscillatory behaviour. Moon (2000) reports that the utilisation level of the congested link fluctuates severely and when the link is congested, packets are dropped, and when the link is under-utilised, bandwidth is wasted. Moon (2000) runs simulations of the Random-Early Detection (RED) queue management to test if it breaks down the synchronisation, and if correlation is no longer detectable by the measure of loss-conditioned average delay. It notes that the traces can be used in a trace-driven IP voice simulator. According to Moon (2000), another approach is to develop and validate an analytical model for the delay and loss, and use the model in simulations. Paxson (1997), however, examines delay-loss correlation, but he reports weak, but not negligible, correlation between jitter and loss. In contrast with Paxson's observations, this thesis will focus on predicting packet loss based upon the observed patterns of delay and jitter, rather than depending merely on the overall amount of jitter as indicator of congestion.

The network delay assigned to a path encompasses the propagation delay across the transmission lines for each individual link in the path together with the accumulated service delay and queuing delay at the hops (routers) of the path. It may be calculated as the Round Trip Time (RTT) and is roughly estimated as twice as the one-way propagation delay (OWD). According to Roychoudhuri and Al-Shaer (2005), the disparity between the above-mentioned M2E delay and One Way Delay (OWD) is dependent on the delay inherent to the employed codecs. A codec is a device or computer program used for encoding and/or decoding a digital signal (Naoum and Maswady, 2012). The IP Packet Delay Variation (IPDV) or jitter, as another QoS factor, can degrade a multimedia application's quality including real-time voice/video stream. According to Gulliver and Ghinea (2007), low amounts of jitter and packet loss can also result in critical debasement of the multimedia perceptual quality. In spite of the presence of receiver-based delay buffer adjustment techniques and complex QoS management schemes, the impact of the delay and jitter cannot be removed completely in broadcast environments (Gulliver and Ghinea, 2007; Roccetti et al., 2001; Moon, 2000). The International Telecommunications Union (ITU) also report investigations about correlations between QoS parameters (Roychoudhuri and Al-Shaer, 2005).

2.2 Routing in Communications Networks

Routing is the function in communications networks that manages transferring data between an end-to-end pair of nodes located in a network. Rapid improvement of storage management control systems in the 1990s provided an opportunity to extend the network routing rules beyond the conventional fixed hierarchical routing to dynamic non-hierarchical routing. The term dynamic refers to time-variant routing methods, which are real-time state-dependent as opposed to static methods (Ash, 1997).

In static routing strategies and algorithms, the routing tables store the expected network parameters like congestion, latency, and bandwidth. The static approach does not update routing tables except, for example, when the topology has been changed (Stallings, 2007). The introduction of dynamic routing into several telecommunications networks has resulted in significant improvement in network connection reliability and availability while decreasing costs.

As stated in Stallings (2007) and Caro (2004), current routing rules and protocols used in IP networks are typically transparent to any QoS/SLA that different packets/traffic flow may have. As a result, and although certain protocols have hooks, routing decisions are presently made regardless of unavailability of the resources and/or lack of requirements. That is, flows are often routed over paths that are unable to support the requirements, while alternate network resources are available on alternate paths.

QoS routing tries to use routing algorithms capable of identifying reliable paths in order to avoid deterioration in performance and to maximise the possible number of flows with regard to their requirements. Moreover, this QoS enhancement should be "*as synergetic as possible with the routing protocols, so as to facilitate their introduction*" (Caro, 2004).

Distributed systems can be characterised as a directed weighted graph (Sobrinho, 2002; Caro and Dorigo, 1998; Schwartz and Stern, 1980). Each node in the set V represents a processing and/or forwarding unit, and each edge in E can be mapped to a transition link in the network. The main task of routing algorithms is making decisions to direct network traffic flow from a sender node to the receiver while maximising network performance to meet QoS requirements. In the work of this thesis, the data flow is not statically assigned, and it follows a stochastic profile that is challenging to model.

As mentioned in the literature, routing algorithms are responsible for interacting with the congestion, admission control algorithms, links' queuing policy, and user-generated traffic (Caro and Dorigo, 1998; Steenstrup, 1995; Bertsekas and Gallager, 1992). In this regard, the routing algorithm used in communications networks is responsible for managing a set of elemental functionalities as follows: (i) aggregating, organising, and publishing information about the traffic and network state; (ii) using the information to generate feasible routes to maximise performance objectives; (iii) passing packets to the selected routes.

In principle, adaptive routing strategies are more attractive, because they adapt the policy of routing decisions based on various network states; namely time, failure, and congestion (Stallings, 2007; Ash, 1997; Maxemchuk and El Zarki, 1990; Schwartz and Stern, 1980). The main disadvantage of these strategies is that they may cause loops in routes taken. Moreover, they are dependent on state information of the network and generate extra congestion for this reason. Bertsekas and Gallager (1992) mention the inconsistency of this category of routing protocols in the case of node or link failures or local topological changes. However, adaptive strategies are appealing to end users for their performance and to network experts for their built-in congestion control mechanism (Stalling, 2007). These benefits depend on the soundness of the design and the nature of the load as Stallings reports. Table 2.1 summaries the major routing strategies based on Stallings (2007).

Table 2.1: Comparison between routing strategies

Routing Strategy	**Advantages**			**Disadvantages**			**Route Selection**
	Simplicity	**Reliability**	**Robustness**	**Congestion risk**	**Oscillations in route**	**Dependency on state information**	
Fixed	*High*	*High**	*High**	*Medium*	*Low*	*Low*	*Based on the expected fixed routing table entries*
Flooding	*High*	*High*	*High*	*High*	*High*	*Low*	*Flooding to all nodes*
Random	*High*	*Medium*	*High*	*Medium***	*Medium*	*Low*	*Least-Cost or based on the probability:* $P_i = \frac{R_i}{\sum_j R_j}$
Adaptive	*Low*	*Medium*	*High*	*Medium / Low*	*Medium*	*High*	*Based on the algorithms*

** in stable and small networks*
*** in comparison small networks*

2.2.1 Problems in Decision-Making for Routing

The principle problems of decision-making for routing reported in the literature are summarised as follows:

- Routing systems and databases are distributed network-wide with strict real-time constraints on them. Failure and state information as well as user generated traffic patterns are also propagated over the network. Foster (1991) states that it seems unlikely to get a thorough and online knowledge about the distributed state while nodes require information from other domains to make routing decisions. Currently, the routing algorithm assigned to each router can only make real-time decisions based on local domain information because information from other domains might be transmitted through other nodes after a delay (Caro and Dorigo, 1998) and the incompleteness of inter-domain information required by DM tools makes real-time decision-making very hard. However, a data mining module can play a critical role in predicting and managing the network decisions. In a real-time scenario, an informed agent (Debenham and Simoff, 2007) may tap the network data to learn online, and eventually, "*determine the best routing table values with respect to the network performance criteria*" (Caro and Dorigo, 1998).

- Decision-making in routing is stochastic and time-varying. The time-variant and stochastic nature of the data generation process makes decision-making complex. To analyse telecommunications data, network data should be transformed into a time-series (Rocha-Mier et al., 2007). Thus, it is hard if not infeasible to depict an image of the whole system as stated by Caro and Dorigo (1998).

- Decision-making in routing is multi-objective. That is, there are various conflicting parameters that influence decisions in routing. In this regard, the most common performance parameters are throughput, packet loss and average delay. For instance, Bertsekas and Gallager (1992, p. 367) state that:

  > "*The effect of good routing is to increase throughput for the same value of average delay per packet under offered load conditions and to decrease average delay per packet under low and moderate offered load conditions.*"

  Other performance metrics measure the results of the routing algorithm's simplicity, flexibility, etc. Moreover, in the special case of QoS network routing, the number of decision factors accrues rapidly, making the situation much harder to be efficiently managed.

- Decision-making in routing is multi-constrained. Heterogeneous underlying network technologies, provided network services, and requested user services are three main sources of constraints (Caro and Dorigo, 1998). Users demand the best quality, most reliability, and lowest costs across the heterogeneous networks. Commercial factors and service providers, on the other hand, try to respond to

> these requests while optimising their profit. Moreover, in a high-speed QoS network, there are growing demands for fault-isolation and reliability. There should be a guarantee that the user sessions get the necessary resources in the proper time and under manageable fault events.

Caro and Dorigo (1998) assert that the problem of decision-making for routing in a communications network belongs to the category of reinforcement learning. They define reinforcement learning as a sub-area of machine learning concerned with how an agent converts situations into actions, so as to maximise a numerical 'reward' signal. Caro and Dorigo (1998) report the need for a precise metric to evaluate forwarding decisions. The measure should be able to deal with incomplete state information that may also be hidden from each agent. In this thesis a hybrid DM model is presented that uses the previous patterns of delay and jitter time-series to approximate the trend and the value of these time-series in the future.

2.2.2 Online QoS Routing

Routing algorithms investigate available routes (with enough bandwidth) to achieve efficient resource utilisation. Moreover, in QoS routing the selected routes ought to have enough resources to meet the requirements of QoS (Marzo et al., 2003). There are two different objective functions used by the QoS routing strategies to maximise efficiency of the network. These are the shortest path and the least loaded path which are used for minimising cost and load balancing, respectively. Since these routing strategies may conflict with one another, they are not simultaneously achievable by mere facilitation of a single routing algorithm.

Guerin, Orda, and Williams (1997) propose the Widest-Shortest Path Routing Strategy (WSP), which uses the Minimum Hop-Count Algorithm (MHA) to choose a path amid the feasible paths. Then, the path with the Maximum Reservable Bandwidth (MRB) is selected, in case more than one path is available. In this regard, the MRB is defined as the minimum of the reservable bandwidth of all links within a path (Katz et al., 2003). Wang and Crowcroft (1991), on the other hand, introduce the Shortest-Widest Path Routing Strategy (SWP). In contrast with the selection criteria of the WSP, SWP select paths based on the MRB, and thereafter based on the MHA over the MRB results. All together, WSP and SWP select a path to fulfill a primary criterion, respectively, amid resource utilisation and load balancing. Table 2.2 summarises a qualitative comparison between two major categories of QoS online routing strategies based on Marzo et al. (2003).

Table 2.2: A qualitative comparison between WSP and SWP routing strategies (adapted from Marzo et al. (2003))

Online Quality of Service Routing				
Strategy	**Primary Criterion**	**Information**	**Selection Procedure**	**Disadvantages**
WSP	*Resource Utilisation*	*MRB*	*Using MHA algorithm and then MRB over the MHA results*	*Selecting a path with larger number of hops – for WSP only).* *No Established Limit*
SWP	*Load Balancing*		*Using MRB algorithm and then MHA over the MRB results.*	*Selecting a path with a congestion point.* *No Request Rejection nor Path Recovery treatment are considered.*

2.2.3 QoS Routing Parameters

This thesis focuses on the IP network layer introduced in Rose and Cass (1987) and considers of the time-series of QoS (performance) parameters needed for informed decision-making in routing (Feng et al., 2011). Rocha-Mier et al. (2007) and Weiss et al. (1998) define certain criteria in their works including Ethernet traffic, DB load and CPU utilisation. However, there are certain performance measures and QoS parameters taken into account in the literature that are more suitable for our work - defined in Green (2007) and Katz et al. (2003) - including: Throughput, Packet Delay, Packet Loss, Jitter, Maximum hops and MRB. In the following, the QoS parameters that mainly used in this thesis are introduced.

2.2.3.1 Packet Delay

The (average) delay per packet (or latency) is the end-to-end traveling time for a packet between the sender and receiver nodes. It is calculated by the sum of a series of network delays as follows:

$$D_{Total} = \sum_{i=1}^{n} D_i \tag{2.1}$$

where D_{Total} is the total delay across the end-to-end path through n networks, and D_i is the delay of a packet through the i_{th} network in the series.

2.2.3.2 IPDV

This parameter, also called jitter, is the deviation in the arrival time of the packets from the expected time. The maximum jitter over a point-to-point link and a series of networks is the sum of the IPDVs for each network in the series. The actual jitter for the point-to-point link, however, may be less than this maximum number because of the positive and/or negative direction of the variations. Cumulative jitter is expressed as follows:

$$J_{Total} = \sum_{i=1}^{n} j_i \tag{2.2}$$

where J_{Total} is the total delay across the end-to-end path through n networks, and j_i is the jitter of a packet through the i_{th} network in the series.

2.2.3.3 Packet Loss

Packet loss is the disposal of a data packet during transmission before it arrives at the destination. Formally, the total percentage of packet loss in multiple networks is defined as the product of all packet loss in each individual network:

$$L_{Total} = 1 - \left(\Pi_{i=1}^{n} (1 - L_i) \right) \tag{2.3}$$

where L_{Total} is the total loss across the end-to-end path through n networks and L_i is the packet loss percentage for the i_{th} network.

2.3 Telecommunications Data Analysis

As the world is becoming more data-driven, the uncertainty and complexity of decision-making is rapidly increasing in manufacturing, business, medical and telecommunications industries (Han and Kamber, 2006). To cope with this growing dilemma in chaotic environments, novel modelling and computing paradigms such as data mining are needed.

The salient characteristics of these new models (Kusiak, 2002) can be described as follows:

- Adaptability of decision-making models to their environment;
- Capability of manipulating qualitative and/or quantitative, varying data;
- Optimality of response time;
- Interpretability of the decisions and outcomes.

Han and Kamber (2006) lists several applications for the telecommunications industry such as novel network design, mobility and microbilling, mobile services, and homeland security (2006). Moreover, as affirmed by Sasisekharan et al. (1996), data mining can be used in forecasting for large-scale communications networks. In programs such as AT&T's Teresa, for instance, the rules and knowledge in its database are used for conducting remote tests and measures. Diagnostic software that attempts to resolve (or predict) chronic and transient faults must rely on large amounts of historical information and analytical mining results of behavioural and also discord patterns (Keogh et al., 2005; Sasisekharan et al., 1996). In another approach, the AT&T's Scout system examines data corresponding to a brief period of time (minutes or hours) and detects patterns among diagnostic data from the global network (Kusiak, 2002; Sasisekharan et al., 1996).

As mentioned before, in this section a general overview of data mining process in a telecommunications network is presented. Later, the theoretical framework and various data sets mentioned, which is used in network data mining. Finally, summations of the two recent and related works that focus on using intelligent agents and QoS pattern analysis for real-time data mining are discussed.

2.3.1 Knowledge Discovery in Telecommunications

Data Mining (DM), or Knowledge Discovery in Databases (KDD), emerged as a modern field from diverse disciplines — including, but not limited to, statistics, artificial intelligence, machine learning, and databases. It identifies valid and up-to-date knowledge together with potentially functional and eventually interpretable patterns out of stored raw data (Han and Kamber, 2006; Fayyad et al., 1996).

The goal of Data Mining is to build models useful for decision-making. In this regard, Data Mining models must provide useful, interpretable knowledge because it is not likely that companies decide based on difficult-to-understand black-box models (Fayyad et al., 1996). Objectives such as model accuracy and interpretability of the knowledge are contradictory to some extent. That is, naive models are typically more interpretable, although they are usually less accurate.

Modern DM techniques should provide highly precise results using high-dimensional models. These techniques also require validation of models and interpretation of results for successful decision-making. Regardless of how powerful is the DM technique used in the modelling stage, the implemented model may not be valid if data are not correctly collected and cleaned, or the modelling is not formulated well.

Data Mining, as a recognised topic within the process of Knowledge Discovery in Databases, finds beneficial patterns within the data using data analysis and knowledge discovery techniques. As an example of inductive learning, DM depends on generalising from previous patterns. In the telecommunications domain, DM model may predict a partial network failure based on previous historical values (Weiss et al., 1998). The DM tasks may be associated with real telecommunication problems. The applications can be considered as classification tasks (e.g., is a network element faulty or not).

The key motivation of data mining is that it replaces the time-consuming and manual process of knowledge acquisition from a domain expert. In particular, DM technology benefits the telecommunications industry since:

- Telecommunications networks are usually too complicated to create thorough simulation models;
- Large amounts of data are commonly available;
- Data Mining can extract cutting-edge, hitherto unexploited knowledge where domain experts might not aware of hidden patterns in data.

Data Mining typically comprises stages including data cleaning and integration, feature selection, transformation, pattern extraction, pattern recognition, and knowledge presentation (Han and Kamber, 2006). The stage in the DM process for telecommunications applications, which usually needs the most attention, is the transformation stage (Weiss et al., 1998). This stage identifies beneficial patterns to represent the data.

The transformation stage is complicated by the fact that telecommunications networks produce huge amount of time-labeled sequences of values – where both of the individual values and their behaviour over time of the network are significant. Because many DM models do not use the temporal time-series directly, the time-series are usually transformed so that can be fed into the models. Another approach is to implement a DM model to understand temporal relations (Rocha-Mier et al., 2007; Miloucheva et al., 2003; Weiss et al., 1998). Figure 2.1 illustrates the typical stages in the KDD process in a communications network based on the perceptual framework presented in Rocha-Mier et al. (2007).

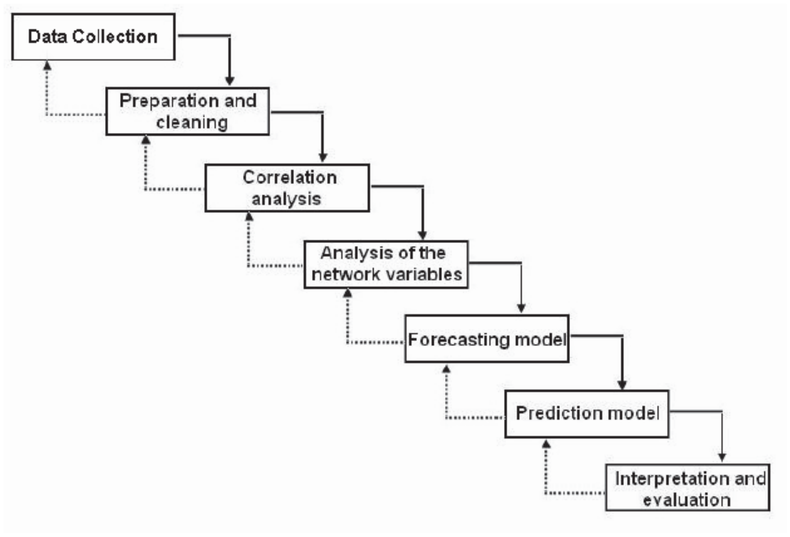


Figure 2.1: Stages of the theoretical framework of the KDD process in a communications network (adapted from Rocha-Mier et al. (2007))

In developing a conceptual framework for the entire data mining process, this thesis draws upon the work of Rocha-Mier et al. (2007); Miloucheva et al. (2003); Fayyad et al.

(1996). As shown in Figure 2.1, the theoretical framework of real-time DM contains seven steps. These steps are explained respectively in the following:

- Collection of telecommunications data from distributed network datasets;
- Data preparation and construction of time-series after transformation of raw data, e.g., deleting repeating and duplicated data, interpolating missing data and applying time window transformations. Rocha-Mier et al. (2007) affirm that the telecommunications network data should be transformed;
- Elimination of redundant network parameters using correlation analysis over those parameters (Pearson or Spearman techniques may be used in this step for correlation analysis. Rocha-Mier et al. (2007) report the latter one);
- Detection of the most relevant variables to the target parameter(s) using importance analysis;
- Implementation of a forecasting model to estimate parameters to be used by a prediction model. The most appropriate forecasting method or a combination is selected for the time-series;
- Implementation of the prediction model. Among various high-dimensional models such as Artificial Neural Network (ANN), Decision Trees (DT) and Linear Regression (LR), the most appropriate predictive model is determined based on the network variable to be predicted. Rocha-Mier et al. (2007) suggests an MLP for this purpose;
- Evaluation of the developed DM model based on the interpretation of the results.

The theoretical framework for data-driven decision-making in Figure 2.1 is founded on the notion of what it means for an agent/expert to be data driven. It has been assumed that the agents, regardless of where they are, have questions, issues, or problems for which data must be collected and analysed in order to make informed decisions. No doubt, many variables at the global and local levels may and will impact local decisions, but as the first step, this thesis examines local decisions, i.e. for two neighbour routers (peers).

This thesis employs DM methods for predicting performance parameters assigned to QoS/SLA in relation to the previous temporal sequences logged for each of them. The forecasted values can facilitate decision-making for routing management, when the informed decision-making software agents are used in directing traffic throughout the network. The DM framework presented in this subsection is used to design the NARGES conceptual framework in Chapter 3.

2.3.2 Data Network Scenarios

Weiss, Eddy, Weiss, and Dube (1998) also list three data types for telecommunications networks: call data, network data and customer data. However, the data type used for routing management is network data and there are two salient types of network data mentioned in the literature for fault prediction and isolation: alarm data and QoS data traces. Rocha-Mier et al. (2007) describe measurement sequences of variables representing the time-series based on data network statistics as part of their project. They used OPNET Modeler for generating artificial statistics and values of network data variables.

Telecommunications network QoS data traces, on the other hand, are used in novel data mining methods for automation of pattern recognition. These data mining methods use similarity analysis of significant most frequent patterns in network and forecasting the QoS datasets (Miloucheva et al., 2003). This involves analysing the time-series of QoS data traces and is required for boosting performance of QoS data mining in network autonomous fault prediction and isolation.

2.4 Prediction Methods and Models for QoS Routing

The telecommunications network environment is chaotic and can change dramatically in a very brief time. Moreover, it is impossible to analyse telecommunications network only by statistics (Rocha-Mier et al., 2007, p. 139). Data Mining techniques that depend on static off-line training may not produce satisfactory results in forecasting network states. Taking into account the eventual variations of the environment, the extracted knowledge from the data mining models can become out of date if it is not updated periodically. Rocha-Mier et al. (2007) state that the stochastic nature of the time-series of network data and the need for real-time forecasting force the prediction model to approximate the target time-series with piece–wise linear patterns. Several algorithms for piecewise linear representation of time-series data mining (TSDM) have been released (Batyrshin and Sheremetov, 2008; Keogh et al., 2005; Lin et al., 2003; Last et al., 2001). For instance, the symbolic presentation method of the time-series called Symbolic Aggregate Approximation (SAX) was reported in Lin et al. (2003) and Keogh et al. (2005). Among the reviewed literature, there are two works especially related to our project, in which the authors state the recognition of discord (outlier) patterns as the main idea of their prediction models.

2.4.1 Intelligent agents for real time data mining

Rocha-Mier et al. (2007) present a multi-agent framework for time-series data mining (TSDM) over a communications network. Their multi-agent predictive model is based on using a multilayer perceptron artificial neural network (MLP ANN) architecture and a DM theoretical framework. Figure 2.2 summarises the TSDM process model presented in Rocha-Mier et al. (2007). As they depict in this picture, a perceptual forecasting model is working within their proposed predictive DM model. The forecasting model is developed based on the Moving Approximation (MAP) transformation (Batyrshin and Sheremetov, 2008).

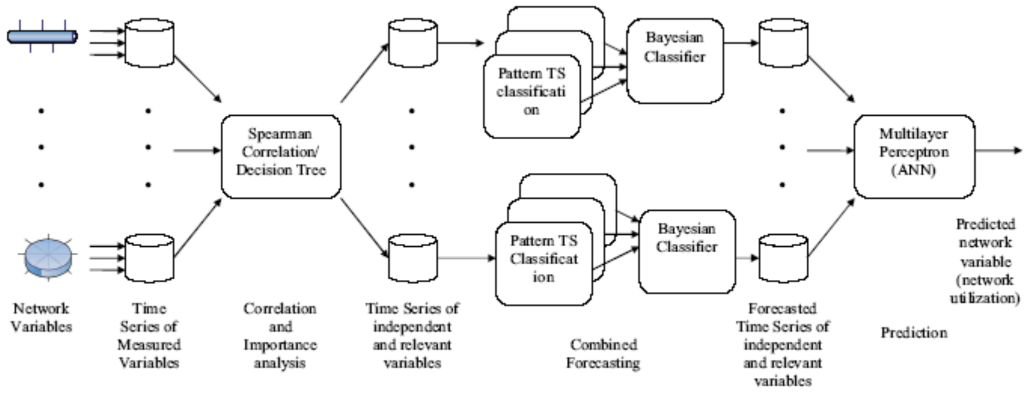

Bayesian Classifier
Pattern TS classification
Spearman Correlation/ Decision Tree
Multilayer Perceptron (ANN)
Pattern TS Classification
Bayesian Classifier
Predicted network variable (network utilization)
Network Variables
Time Series of Measured Variables
Correlation and Importance analysis
Time Series of independent and relevant variables
Combined Forecasting
Forecasted Time Series of independent and relevant variables
Prediction

Figure 2.2: Process model of a communications network TSDM (adapted from Rocha-Mier et al. (2007))

The MAP method facilitates decision-making for intelligent agents and/or expert operators. As discussed by Batyrshin and Sheremetov (2005) and Rocha-Mier et al. (2007), in MAP method the goal pattern of time-series values can be substituted by the perceptual pattern of slope values describing a class of time-series patterns with the shape of the goal pattern. The method searches for the sequence of the most similar slope values to the perceptual image of the goal pattern. Figure 2.3 shows an example of an initial time-series and the pattern of slope values as presented in Rocha-Mier et al. (2007).

The MAP method recognises all the patterns in the time-series sequences of measured values. Then, it analyses these patterns for each unit of time window - namely a day or hours - to generate classes. Rocha-Mier et al. (2007) calculate the corresponding probability of the class to be chosen. Their proposed distance function between the time-series of patterns considers certain parameters mentioned in Section 2.2.3:

$$d(x, y) = \sqrt{\sum_{i=1}^{m} \alpha(t_{x_i} - t_{y_i})^2 + \beta(q_{x_i} - q_{y_i})^2} \tag{2.4}$$

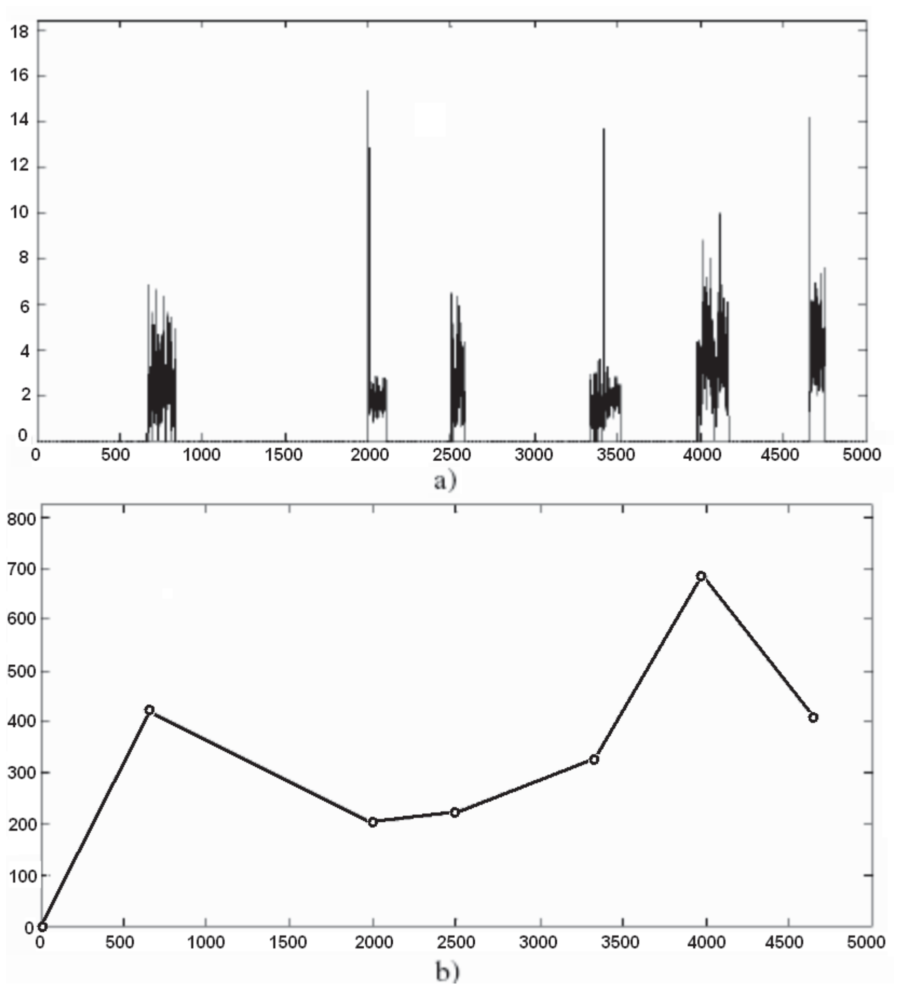


Figure 2.3: Initial and pattern time-series for a network variable a) target time-series, b) pattern time-series of slope values (adapted from Rocha-Mier et al. (2007))

where m is a number of patterns in a considered class of time-series of patterns x and y, i is an index to current pattern, the t and q in this function are the time point and intensity of patterns, respectively, α and β are normalising parameters. Rocha-Mier et al. (2007) describe that the predicted values are derived from the most similar pattern following that of the goal. That is, the location value of the pattern found is extendable to a predicted location value of the goal pattern. This helps in computation of the predicted value of the goal pattern by means of the slope value, which is directly following the found subsequence.

2.4.2 QoS pattern analysis

Pattern analysis is a specific data mining approach dependent on heuristics to investigate, analyse, model, and predict compartments and dependencies of time-series by the use of patterns (Miloucheva et al., 2003). Pattern based data mining tools were developed for various applications and fields. For instance, Keogh et al. (2005) comments that the discords annotated by a cardiologist are widely used for anomaly detection in Heart Beat patterns in an electrocardiogram (Electrocardiography (ECG)). As another example, Van Wijk and Van Selow (1999) present a clustering based pattern analysis tool for exploration and visualisation of large quantities of univariate time-series data. Query Sketch is also an online prototype program for graph centric queries (Miloucheva et al., 2003).

Time-series discords also have many uses in data mining, including quality enhancement for clustering, data cleaning, summarisation, and anomaly detection algorithms (Keogh et al., 2005). As shown in Figure 2.4, the basic idea behind this work is that the significant kinds of outliers in the time-series sequences can be used as effective anomaly and failure detectors in symmetric systems with periodic patterns.

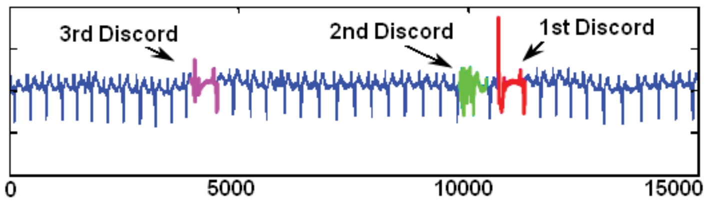


Figure 2.4: Outliers (discords) are particularly attractive as anomaly detectors (adapted from Keogh et al. (2005))

As illustrated in Figure 2.4, discords are subsequences of a longer time-series and have maximal contrast with all the rest of the time-series subsequences. They, thus, depict the most unusual subsequences within time-series and may be used to predict future failures and problems.

The importance of pattern recognition as a data mining technique in predicting network behaviour is addressed by (Estan et al., 2003). Moreover, Miloucheva et al. (2003) report a gap in the development of automated pattern recognition and similarity analysis tools for QoS data mining. The European IST-INTERMON project develop an integrated architecture for visual data mining and inter-domain QoS monitoring, analysing and modelling of network traffic. The framework developed for the European IST Inter-Domain QoS Monitoring project (INTERMON) project integrates QoS analysis within a large-scale inter-domain networking environment based on the interactions with monitoring, topology discovery and traffic analysis tools. Miloucheva, Hofmann, and Gutirrez (2003) define a pattern definition language (PDL) for their research project, in which they consider the specifics of the pattern structures derived from time-series of measured performance parameters in the telecommunications network. They introduce basic QoS pattern types including extreme value, plain, increase, and decrease and then define composite pattern types based on the basic patterns. The basic and composite patterns later are used in a spatio-temporal QoS pattern analyser developed for the INTERMON project's framework. The PDL given in Miloucheva et al. (2003) encompasses the following syntax as

$$\textbf{\textit{Composite Pattern Structure}} =$$

$$\langle \textit{Shape Parameters} \rangle, \langle QoS\ Description \rangle, \langle \textit{Basic Pattern Sequence} \rangle$$

$$\textbf{\textit{Shape Parameters}} = \text{‘}shape\text{’}\ (DistanceMeasure, \langle DefinitionOptions \rangle)$$

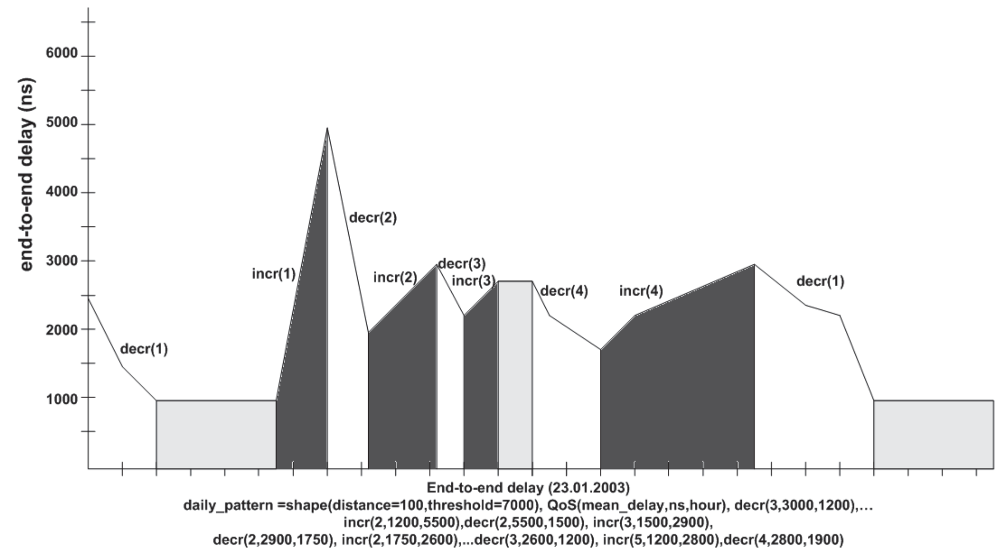
$$\textbf{\textit{QoS Description}} = \text{‘}QoS\text{’}\ (QoS\ Metric, Metric\ Unit, Monitoring\ Scale)$$

$$\textbf{\textit{Basic Pattern Sequence}} = \textit{Basic Pattern Structure}\ \langle, \textit{Basic Pattern Structure} \rangle$$

$$\textbf{\textit{Basic Pattern Sequence}} = \langle Basic\ Pattern\ Type \rangle\ \text{‘(’}\ \langle \textit{Length} \rangle \langle, \textit{Range} \rangle\ \text{‘)’}$$

$$\textbf{\textit{Basic pattern type}} = \text{‘}\textit{Extreme}\text{’}, \text{‘}\textit{Plain}\text{’}, \text{‘}\textit{Incr}\text{’}, \text{‘}\textit{Decr}\text{’}$$

In the above syntax, basic pattern structures encompass extreme, plain, increase and decrease pattern types. For all basic patterns within the composite pattern structure, shape parameters are used to explain the structure options. A Distance Measure is a shape parameter by which an interval is defined. The interval is used to elicit the particular basic pattern type. An outlier definition option measures the means of computing extreme value patterns. Figure 2.5 shows pattern structures, which are acquired from active QoS measurement discussed in Miloucheva et al. (2003).



```
daily_pattern =shape(distance=100,threshold=7000), QoS(mean_delay,ns,hour), decr(3,3000,1200),...
incr(2,1200,5500),decr(2,5500,1500), incr(3,1500,2900),
decr(2,2900,1750), incr(2,1750,2600),...decr(3,2600,1200), incr(5,1200,2800),decr(4,2800,1900)
```

Figure 2.5: PDL pattern structures from active QoS measurement (adapted from Miloucheva et al. (2003))

The developed INTERMON technology is based on automated recognition, analysis, storage and processing of QoS patterns regarding the network topology and the temporal context. The QoS patterns are derived from network topology and periodical time windows, respectively. As discussed before, these units and patterns may be employed by the agents and/or network engineers to help them make informed decisions based on the experiences of route change, abnormal QoS values, and requested QoS. Figure 2.6 shows a data mining approach utilising the spatio-temporal QoS pattern analyser based on Miloucheva et al. (2003) work.

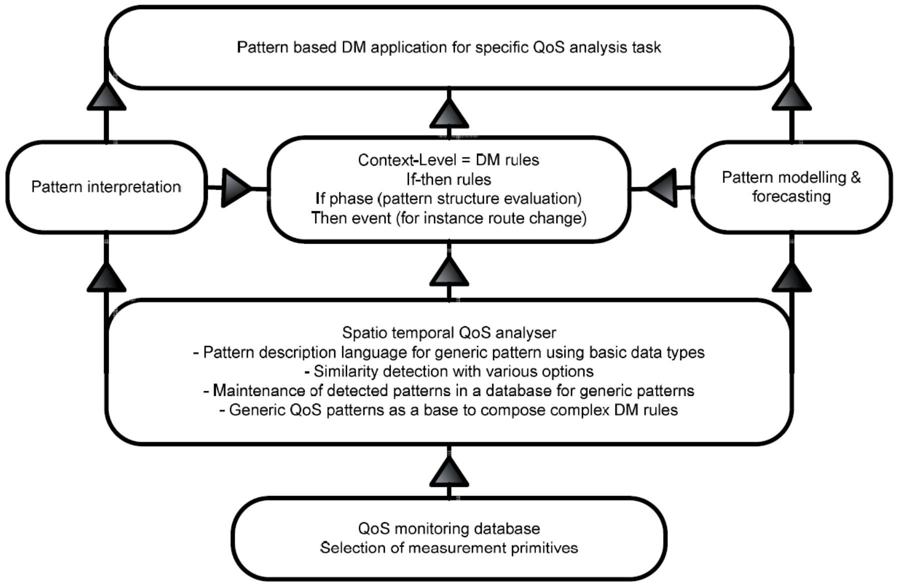

Pattern based DM application for specific QoS analysis task
Pattern interpretation
Context-Level = DM rules
If-then rules
If phase (pattern structure evaluation)
Then event (for instance route change)
Pattern modelling & forecasting
Spatio temporal QoS analyser
- Pattern description language for generic pattern using basic data types
- Similarity detection with various options
- Maintenance of detected patterns in a database for generic patterns
- Generic QoS patterns as a base to compose complex DM rules
QoS monitoring database
Selection of measurement primitives

Figure 2.6: Using Spatio-analyser for automation of the DM tasks (adapted from Miloucheva et al. (2003))

As explained later in Chapter 3, the concept of PDL is used to define pattern definitions for a subcomponent of the proposed model in this thesis.

2.4.3 Other Related Work

Internet packet–loss and delay variation show temporal dependency. If packet n is lost, packet $n+1$ is likely to be lost. This trait leads the network to a bursty packet-loss in real–time multimedia services such as VoIP (Jiang and Schulzrinne, 2000) that may degrade QoS and the effectiveness of recovery mechanisms such as *forward error correction*.

Various research projects have studied the delay and packet loss, jitter, available bandwidth and other performance traces in the network so as to predict network anomalies or differentiate them from noise. The path–load measurement by Jain and Dovrolis (2002) employs jitters to calculate the available bandwidth. However, as confirmed by Roychoudhuri and Al-Shaer (2005), there is not a detailed analysis of online prediction of the packet loss from the jitter.

Some researchers report techniques using jitter and TCP window size for congestion avoidance (Brakmo et al., 1994; Wang and Crowcroft, 1991; Jain, 1989). Others use these results for classification of loss and noise in packet transmission (Biaz and Vaidya, 1998). Tobe et al. (2000) discriminates various degrees of congestion according to the relative *one way delay* and *static delay thresholds*. They do not measure or predict delay trends and packet loss based on the congestion level Roychoudhuri and Al-Shaer (2005). Other research projects (Dovrolis et al., 2001; Paxson, 1997; Carter and Crovella, 1996) have employed packet routing techniques to estimate the path capacity and available bandwidth. Specifically, Paxson (1997) study the relation between delay and jitter traces and the packet loss. He assert the correlation between delay and packet loss, but report a weak, though not negligible, dependency between jitter and packet-loss. In contrast with Paxson's findings, Roychoudhuri and Al-Shaer (2005) predict packet-loss using patterns in delay variation, time-series, rather than the overall amount of the jitter parameter. They develop a model to forecast congestion based on packet-loss prediction by means of jitter observations.

In this regard, this thesis addresses the issue of defining a symbolic delay forecasting model, which is based on historical frequencies of observed patterns of the delay variation in adjacent uniform windows. In particular, this research proposes a DM model, as part of a informed (Debenham et al., 2008) routing management model, to be used in a smart network router for online routing management. Next section reviews models and techniques of time-series analysis and forecasting relevant to the work of this thesis.

2.5 Time-Series Analysis and Forecasting

A time-series is a time-variant sequence of data measured on a variety of cases. Time Series analysis and forecasting is an important problem in many fields (Fu, 2011).

Forecasting future events is a crucial task and the predicted data may be used as the input of planning and decision-making processes. Despite the wide range of the forecasting methods, all of them fall into two approaches: quantitative or qualitative.

Most of the qualitative forecasting methods are intrinsically subjective and require expert interpretation. These methods are typically used when there is a historical data based on which the forecast is done. Quantitative methods, on the other hand, are based on using historical data (even with missing data) and a mathematical forecasting model.

The forecasting model in quantitative methods formally extrapolates future data by means of past and current patterns in historical data (Montgomery et al., 2008). General time-series models apply statistical features of historical data to approximate the unknown factors of the model. Accurate identification of the proper forecasting method is significant and, as Montgomery et al. (2008) state, it can help to improve prediction time and accuracy.

In this thesis, a hybrid model is designed and developed that uses both of the qualitative and quantitative techniques. This enables the proposed model to have fault-tolerance in dealing with missing data while comparing favourably with a quantitative method such as ARMA — by making faster decisions based on the historical value of the QoS traces assigned to the links.

In the following subsections, relevant to the work of this thesis, certain quantitative and qualitative time-series analysis models are described.

2.5.1 Quantitative Time-Series Analysis

Autoregressive moving average (ARMA) models (Box et al., 2011) are often referred as Box-Jenkins models in time-series analysis literature (Brockwell and Davis, 2006; Chatfield, 2004) because the iterative Box-Jenkins method is typically employed to approximate the time-series data. The ARMA model is a tool for understanding and forecasting time-series sequences. The model includes two Autoregressive Model (AR) and Moving Average Model (MA) parts. It is usually then referred as the ARMA(p,q) model where p and q are the orders of the autoregressive and moving average parts, respectively.

Analysis of the conventional time-series future movements is highly based on the historical values of it. There are certain implicit factors that lying on the time-series, but appearing in other data sources, showed to have significant impacts on the time-series behavior. A typical example of such time-series is the link utilisation level. The fluctuation of a specific link utilisation level is the consequence of certain performance parameters such as delay, jitter and packet-loss, where the values of the latter parameters in each time window directly affect the utilisation level.

There are various ranges of methods used in the literature about stationary and non-stationary models for statistical time-series analysis and forecasting. These are including Autoregressive Model (AR), Moving Average Model (MA), Autoregressive Moving Average (ARMA), Autoregressive Fractionally Integrated Moving Average (FARIMA), Autoregressive Conditional Heteroskedasticity Model (ARCH) and Hidden Markov Model (HMM) models. The nonlinear model ARIMA/GARCH defined in Zhou et al. (2006) forecasts the network traces. The standard model fitting procedures do not require any assumption for the underlying structure of the system and can be used for systems without background information (Antari and Zeroual, 2009). There is, however, less certainty concerning the use of certain loss models, and misunderstanding assigned to the common models used; such as Gilbert model. In the following subsections, ARMA model is introduced to be used in Chapter 4 as a benchmark for evaluating the proposed models.

2.5.1.1 Modeling and forecasting with ARMA

In this section, the ARMA model is briefly described. The ARMA will be used later for model evaluation in chapter 4. For more information about the other models refer to the cited literatures and Appendix B. The ARMA model is based on the definition of a stochastic process in a probability space. A probability space is a measure space Ω on which an absolutely additive measure P is defined such that $P(\Omega) = 1$ (Hsu and Robbins, 1947).

A stochastic process is formally defined in Chatfield (2004) and Box et al. (2011) as the collection of random variables, ordered in time, which are discrete or continuous — discrete in this thesis. Stochastic process is a statistical phenomenon evolving in time based on probabilistic rules.

Stationarity is defined as a class of *stochastic process*. It is the characteristic of a time-series based on the statistical properties of the times series. According to Chatfield (2001) a time-series x_t is stationary if the time-series mean, $E(x_t)$, is a finite constant. That is characterised as the constant probabilistic distribution of the time-series (Montgomery et al., 2008).

The QoS time-series used in this thesis are discrete stationary processes. The AR, MA and ARMA processes are described in the following subsections.

2.5.1.2 AR Model

The process X_t is called an Autoregressive of order p ($AR(p)$) if:

$$X_t = \sum_{k=1}^{p} \phi_k X_{t-k} + Z_t \tag{2.5}$$

where ϕ_ks are constants, $0 \leq k \leq p$, and Z_t is purely random process. The mean and variance values for AR_p process is $Mean(X_t)$=0 and $Var(X_t) = \sigma_z^2 \sum_{k=1}^{p} \phi_k^2$.

2.5.1.3 MA Model

The process X_t is called a Moving Average of order q ($MA(q)$) if:

$$X_t = \sum_{l=1}^{q} \theta_l Z_{t-l} \tag{2.6}$$

where θ_ls are constants, $0 \leq l \leq q$, and Z_t is purely random process. The mean and variance values for MA_q process is $Mean(X_t)$=0 and $Var(X_t) = \sigma_z^2 \sum_{l=0}^{q} \theta_l^2$.

2.5.1.4 ARMA models

The process X_t is called Autoregressive Moving Average of order (p, q) (ARMA(p,q)) if:

$$X_t = \sum_{k=1}^{p} \phi_k X_{t-k} + Z_t + \sum_{l=1}^{q} \theta_l Z_{t-l} \tag{2.7}$$

where ϕ_ks and θ_ls are constants, $0 \leq l \leq q$ and $0 \leq k \leq p$, and Z_t is purely random process.

ARMA process can model a time-series with fewer parameters than pure MA and AR processes. This is a primary apotheosis of what is called the Principle of Parsimony.

According to Chatfield (2004), many stochastic processes may be expressed by formal algebraic modeling of the random variable based on its past values with imperceptible error process. For further information about the ARMA process the reader can refer to various literature including Box et al. (2011), Montgomery et al. (2008), Chatfield (2004), Brockwell and Davis (2006) and Chatfield (2001).

Selecting the proper ARMA(p,q) model is significant in representing an observed stationary time-series with a number of factors. These include the choice of p and q (order selection), estimation of the mean, the coefficients $\phi_k, k = 1, \ldots, p$, $\theta_l, l = 1, \ldots, q$, and the white noise variance σ^2. Final selection of the model also depends on a variety of goodness of fit tests. In Section 4.1.1 of the thesis, ARMA algorithm is described as the benchmark used for models evaluations.

2.5.1.5 Non-Stationary Models and ARIMA

Let d represents a non-negative integer. If Z_t in the Equation (2.7) is a causal ARMA(p,q) process, then X_t is an ARIMA(p,d,q) process.

Autoregressive Integrated Moving Average (ARIMA) model, as a generalisation of ARMA which includes a wide range of non-stationary time-series, is developed based on the definition of ARIMA processes. That is, after differencing the sequence finitely many times, ARIMA processes reduce to ARMA processes (Brockwell and Davis, 2006). In this regard, for a non-stationary time-series, ARIMA model may be used as an alternative benchmark instead of ARMA. This thesis, however, uses ARMA while deals with stationary time-series.

2.5.2 Qualitative Time-Series Analysis

As mentioned before, the qualitative methods are intrinsically subjective. There is a wide range of qualitative methods. In this regard, the most relevant methods to the work of the thesis is reviewed in this subsection. In the following, frequent pattern mining, perception based data mining and multilayer perceptrons are described.

2.5.2.1 Frequent Pattern Mining

Mining similar subsequences or substructures that appear frequently in an item-set with not less than a user-specified threshold is called *frequent pattern mining*. The concept of frequent pattern mining was first proposed by Agrawal et al. (1993) for mining in transaction databases.

A subsequence that occurs frequently in a history database, is called a frequent sequential pattern. Subgraphs, subtrees, or sublattices form a basic or composite substructure. Frequent pattern retrieval plays a significant role in mining associations, correlations and relationships in a database. It can benefit indexing, classification, clustering and other data mining tasks. Thus, frequent pattern mining become a robust data mining task and a research area in many fields. Three basic frequent pattern mining methodologies Apriori, Frequent Pattern (FP)-growth and Eclat, and their extensions, are reported in the literature (Han et al., 2007).

Let $I = i_1, i_2, ..., i_n$ be a set of all items (values). A $k - itemset$ (pattern), which has k items from I, is frequent if a pattern occurs in a database D a frequency no lower than $\theta|D|$ times. In this regard, θ is a user-specified minimum support threshold (called min_sup in (Han et al., 2007)), and $|D|$ is the total number of transactions in D.

In telecommunication, the time-ordered QoS traces are spatiotemporal time-series. Spatiotemporal data mining finds spatiotemporal knowledge and patterns within the QoS time-series.

Frequent pattern mining has been widely used in last two decades. Xiong et al. (2004)

and Zhang et al. (2004) use frequent pattern mining method for mining co-location patterns. Koperski and Han (1995) explains how progressive refinement method substantially optimises the algorithm time cost by performing rough computation while it refines the results at a better resolutions. Cao et al. (2005) extends the optimization ideas of mining sequential patterns, which are location and time related ; i.e. spatiotemporal patterns. Besides, Li et al. (2006) mine outliers in moving object datasets, by movement of the fragment patterns through spatial time-series data to find motifs of discords by a motif-based classification (Han et al., 2007).

Similarly, frequent pattern mining has a significant role in telecommunications data mining. Han et al. (2007) denotes that multimedia time-series can be treated as spatiotemporal data and frequently patterns occurring in multimedia and streaming time-series can be discovered. Han et al. (2007) lists various datasets that can be analysed by this method. These include huge volumes of telecommunications data streams generated within real-time communications network and Internet traffic. Unlike traditional datasets, stream data flows continuously and chaotically (Han et al., 2007). A smart router is incapable of storing the whole data stream or to scan through it multiple times due to tremendous volume of the data. Han et al. (2007) denotes that it is vital to develop single-scan and on-line mining methods for data mining over data streams.

With the aim of mining frequent values and subsequences on data streams, Manku and Motwani (2002) introduce sticky sampling and lossy counting methods for approximating frequency counts over data streams. Karp et al. (2003) also propose a counting technique for finding frequent items in stream data (2003). Yu et al. (2004) propose a frequent pattern mining algorithm with a *false-negative* oriented approach and compare it with the *false-positive* mining approach, which also called as *lossy counting*. They assert that their approach can mine the frequent elements, although it is limited to the capacity of the virtual memory. Their approach can become intractable where the number of false-positive frequent itemsets increases exponentially.

Chi et al. (2004) proposes a mining algorithm that finds closed frequent itemsets within a sliding window. They designed a closed enumeration tree structure for probing items within the sliding window and maintaining a chosen subset of transactions. Metwally, Agrawal, and El Abbadi (2005) introduce a computing method for mining frequent and *top-k* items in streams of data with a careful memory consumption. Jin and Agrawal (2005) proposes a one pass algorithm for frequent mining of itemset with a specified limit on the precision that does not need any out-of-core summary structure. Lin et al. (2005) proposes an algorithm for frequent itemsets mining from streams of data using a time-sensitive sliding window.

The method of *frequent pattern mining* is considered in the definition of the proposed model in Section 3.3. In this regard, the items and $k - itemset$ are considered as the values and patterns within a QoS time-series, respectively.

2.5.2.2 Perception Based Data Mining

Another method, which is considered for model definition in this thesis, is Perception-based Data Mining (Batyrshin and Sheremetov, 2008). Although the mapping function defined in Chapter 3 is deterministic, the idea of the fuzzy rule-based linguistic descriptions may be used in mapping time-series numerical values into categorical terms in Section 3.3.

Perception based function is defined by a set of rules R_i: "*If* X *is* T_i *then* Y *is* S_i," in which T_i are categorical terms describing certain fuzzy intervals on the set of real numbers of X and S_i are the categorical descriptions of the shapes of a function $Y(X)$ on the intervals (Batyrshin and Sheremetov, 2008). The rule is used in many fuzzy models such as Mamdani (1974). In most trivial case, the rule consequence part may include a fuzzy value of function Y, e.g.

$$R_i : If\ \textbf{DELAY}\ is\ \textbf{HIGH}\ then\ \textbf{SLOPE}\ is\ \textbf{SHARPLY INCREASING}. \qquad (2.8)$$

The above rule states that the SLOPE variable is varying with the speed categorically evaluated in the consequence part while the independent numerical variable referring to DELAY is changing within the fuzzy interval SHARPLY INCREASING. In other word, the Equation (2.8) regards the categorical expression of dependency between variables DELAY and SLOPE, such that SLOPE is a SHARPLY INCREASING function of DELAY on the fuzzy interval HIGH.

The categorical terms in the consequence part of a rule like (2.8) is regarded as directions of function trend. The explicitness of the rule depends on the granulation of these directions (Batyrshin and Sheremetov, 2005). Batyrshin and Sheremetov propose techniques of restructuring fuzzy functions based on the set of such rules. The initial fuzzy value is propagated rule by rule in the trends defined in consequence parts of the rules and eventually build a fuzzy function (fuzzy relation).

Models using fuzzy perceptions are used for realisation of the Computing with Words (CW) methodology presented in (Zadeh, 2001, 1999). According to Zadeh (2001), CW methodology is considered as a problem of calculation of a Terminal Data Set (TDS) from an Initial Dataset (IDS) in several steps. These steps are: (i) explicit interpretation of IDS as fuzzy rule-sets; (ii) propagation of constraint to transform antecedent constraints into consequent constraints, and (iii) translation of consequent constraints to linguistic form of TDS.

This thesis builds deterministic rules using the initial rule-based methods of formalisation of rules like (2.8) to define the DMF presented in Chapter 3.

2.5.2.3 Multilayer Perceptron

The term neural network describes a network (circuit) of biological or artificial neurons. The modern adoption of the term usually denotes ANN as a composition of virtual interconnected neurons (nodes). The nodes represent software agents simulating the properties of biological neurons. The ANN architecture is specified by the number of nodes, their arrangement and connectivity (Lopez et al., 2008). In this regard, the ANN can be seen as a labeled directed graph where the graph's nodes and edges are the respective ANN's nodes and inter-conductivities among the nodes, and the free parameter remarked by the labeled edge (Dongare et al., 2012). Most common forms of neural networks, including biological one, have a layered structure. Typically, a set of sensorial input nodes, one or more hidden layers of neurons and a set of output neurons form the input, hidden and output layers of the ANN, respectively.

In this thesis, a multilayer perceptron (MLP) with a feed-forward network architecture is used. In the proposed hybrid DM model the MLP is used to control the packet-loss predictions and to improve the precisions of the results.

2.6 Conclusions

The chapter reviewed qualitative and quantitative models for time-series analysis. Studying QoS time-series analysis demanded reviewing of a wide range of topics about routing in communications networks, telecommunications QoS data, prediction methods and models for QoS routing. This chapter also highlighted the time-series analysis and forecasting. There are many models and methods for time-series analysis and QoS routing.

Stochastic characteristic of inter-domain routing time-series and the lack of concurrent data at the time of predictions are a few issues that influence the ability of mining telecommunication data. Another concern is that telecommunications data is often not in a form or at a level-suitable for Data Mining. The large volume of telecommunication data sets and the need to operate in real-time are other common issues.

An important issue in telecommunications data mining is the restrictions on sharing data, information and knowledge. Companies may not utilise data from other resources out of their corporation. Consequently, knowledge discovery over telecommunication network database and data-stream are limited when the extracted knowledge may not be exploited at least at the time of predictions.

Using a hybrid model is vital to compensate the lack of concurrent data. Besides, utilising historical patterns of data is vital to approximate the trends and values of concurrent data and to be able to predict the future values. In this regard, the aim of this research is to design and develop a hybrid data mining model that is simple to use within a smart edge router and can predict packet-loss independent from the QoS data from other network autonomous systems owned by other companies.

Chapter 3 introduces NARGES data mining framework and model, which uses current trend and value of the forecasted delay and jitter, to predict future number of the packet-loss assigned to a peer network node/link within other autonomous systems.

Chapter 3

Data Mining Model for Packet Loss Prediction

"All models are false but some models are useful,"

— George E. Box

The majority of packet-loss in the Internet occurs due to congestion in links. Real-time multimedia applications over the Internet such as VoIP, video/audio conferencing and streaming are sensitive to packet-loss, and packet retransmission is not an acceptable solution with these sorts of application. Predicting packet-loss with some certainty from network dynamics of past transmissions is crucial knowledge to inform smart routers to make better decisions. Network applications need to identify congestion in the path of packets to take suitable rate control actions. Regarding the studied literature on the area of telecommunications data mining, a data mining model is designed and developed for classification of links that have a high probability of packet-loss. The model, implemented with MATLAB, is intended to contribute to making informed decisions within smart edge routers where the quality of transmissions should be controlled and is primarily determined by the level of packet-loss.

In this thesis and in the special case of packet-loss prediction, the application of human-like perceptions in making smart decisions is considered and employed. The developed data mining model memorises the frequency of various patterns for three adjacent time-series values within a sliding windows of time – three delay or jitter time-series values. The provided foreknowledge of different pattern frequencies, i.e. the learned perceptions of the current trend and value of the time-series are used in simulations to approximate the next trends and values – instead of running a costly statistical algorithm each time.

The rest of this chapter is organised as follows. Section 3.1 overviews the model. Section 3.2 gives a brief description on preliminary concepts in this chapter. Formal description of the models are introduced in Section 3.3. An implementation paradigm of the models is described in Section 3.4. The chapter concludes in Section 3.5.

3.1 Overview

The fundamental idea of the model is to predict the amount of packet-loss assigned to a link by approximating the trends and values of the delay according to previously observed patterns. The framework and models presented here can be utilised within simulations by using intelligent software agents for data mining in the edge routers. The agent predicts the number of packets lost in a link, according to the forecasted delay and jitter. This can potentially reduce the intermittent broadcasts of the link availability to its peers and the traffic made by the nodes for sending extra packets through the network to report congestions within their autonomous system can be attenuated in this way.

A hybrid data mining model is deployed to firstly approximate the real–time data values and then to use them as the input in the second part of the model, a multilayer perceptron.

The second part of the model predicts the packet-loss based on the approximated delay and jitter values and is used to enhance the precision of the model prediction in the simulation. Figure 3.1 represents the conceptual framework of the data mining model for smart routing in a communication networks (NARGES).

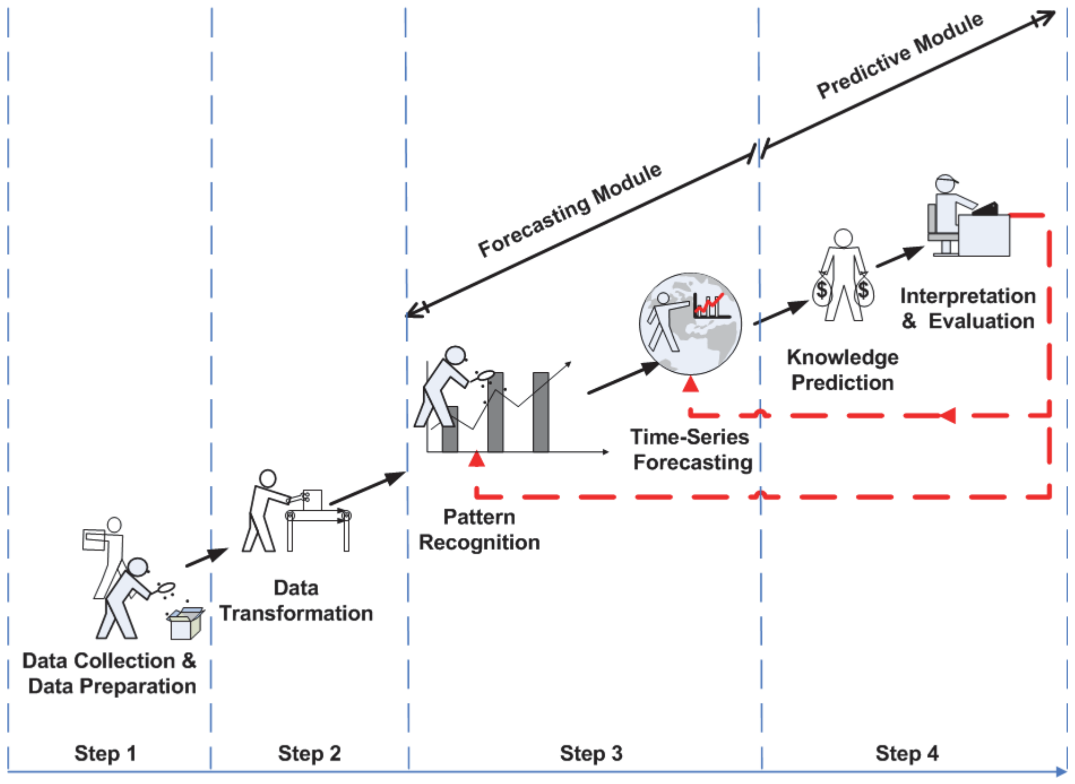


Figure 3.1: Conceptual Framework for NARGES Data Mining Model

As shown in Figure 3.1, the proposed framework for real–time data mining encompasses the following steps to predict the amount of packet-loss:

- **Step 1 - Data Collection and Preparation**

  This step gathers delay, jitter and packet-loss time-series, assigned to a telecommunication network link, from the available data sources (e.g. network data logs or management information bases from network segment routers). As mentioned, the data collection is accomplished by using:

1. D-ITG network traffic generator:

   D-ITG is an academic free software agent employed to generate network traffic. There are three main threads to be run on two physical server and client nodes in a network, namely the traffic generator (ITGSend) on the server side and the traffic receiver (ITGRecv) and ITGLog used for the data collection on client side.

2. OPNET Modeler:

   The data collection on OPNET Modeler occurs totally in a simulation environment, i.e. no physical nodes are used. The QoS data can be exported after running simulation.

- **Step 2 - Data Transformation**

  The collected data is transformed (e.g. deletion of repeated or duplicated data, normalisation of time scales or averaged) to be ready for use within the DM process. In this regard, a moving average transformation is run on the group of time-series values, within a time window, to reduce the volume of the data.

  The transformation also helps reduce the overhead of the model prediction procedure on smart router while the real-time frequently generated logs need to be processed each time to use network resources. There is obviously a trade-off between the precision of the overall algorithm output and the number of time-series values used each time for this transformation.

  The transformation window in this step is different from the sliding window used for time-series approximation," in step 3. The latter concept will be discussed in Sections 3.3.1.1.

- **Step 3 - Pattern Recognition and Data Forecasting**

  This step introduces the forecasting module of the hybrid DM model named HDAX. Five basic patterns are defined for forecasting trend changes of a time-series. These patterns are used for the definition of composite patterns and for populating the frequency entries assigned to each of the composite patterns within a look-up table. The frequencies are used for estimating the probability of each occurrence of the composite patterns. The look-up table is later used for pattern forecasting and time-series approximation. In step 4, the outputs of HDAX forecasting module, i.e. the approximations of current delay and jitter at time t, are used in the predictive module in NARGES model.

- **Step 4 - Knowledge Prediction, Interpretation and Evaluation**

  In this step, the approximated delay and jitter values at time t from step 3 are used to predict the number of packets lost at time $t + 1$. A multilayer perceptron is used to do this.

  Moreover, the MLP outcome must be interpreted and used to evaluate the model and to get the highest precision needed in a real-time scenario. This is because the occupance of the composite patterns within the delay and jitter time-series may be affected during different periods of a day or week — by the diverse network conditions such as physical failure, number of end-users, changes in network bandwidth, and the like. To overcome this, a controller may be added to the designed model as part of the MLP in the predictive module of NARGES to detect the prediction errors and send a reactive signal to reset the look-up table.

Figure 3.2 shows the design of NARGES DM model and the flow of the data. The HDAX forecasting module approximates the delay and jitter at time t based on the previous values of the time-series and their change trends at times $t - 3$ to $t - 1$. The prediction in NARGES model is based on the current approximations of the values of delay and jitter variables at time t; which are not known at the time of the prediction. As shown in Figure 3.2, the MLP and HDAX outputs are audited by an error controller to enhance the precision of the packet-loss prediction.

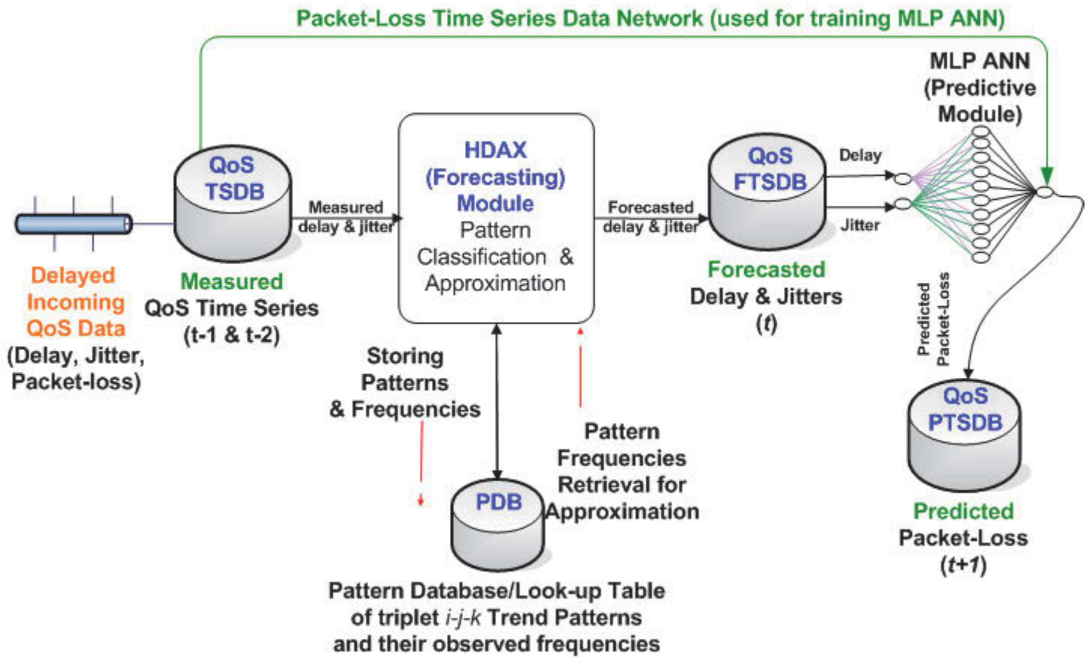

Packet-Loss Time Series Data Network (used for training MLP ANN)
MLP ANN (Predictive Module)
QoS TSDB
Measured delay & jitter
HDAX (Forecasting) Module
Pattern Classification & Approximation
Forecasted delay & jitter
QoS FTSDB
Delay
Jitter
Delayed Incoming QoS Data (Delay, Jitter, Packet-loss)
Measured QoS Time Series (t-1 & t-2)
Forecasted Delay & Jitters (t)
Predicted Packet-Loss
Storing Patterns & Frequencies
Pattern Frequencies Retrieval for Approximation
PDB
QoS PTSDB
Predicted Packet-Loss (t+1)
Pattern Database/Look-up Table of triplet i-j-k Trend Patterns and their observed frequencies

Figure 3.2: A schema of NARGES data mining model

The HDAX forecasting module supplies the forecasted input values of frequencies for each possible time period. It uses a symbolic approximation to get the less accurate but faster results for the MLP module. These are estimates of the current delay and jitter and the MLP has been previously trained to use them for packet-loss prediction. After the prediction of network packet-loss, the results may be interpreted and used to improve the performance of a *future smart router* decision-making.

3.2 Preliminary Descriptions

This section aims to deliver descriptions on certain concepts used in the rest of the Chapter. These are QoS time-series and Pattern Definition and Look-up Table.

3.2.1 QoS Time Series

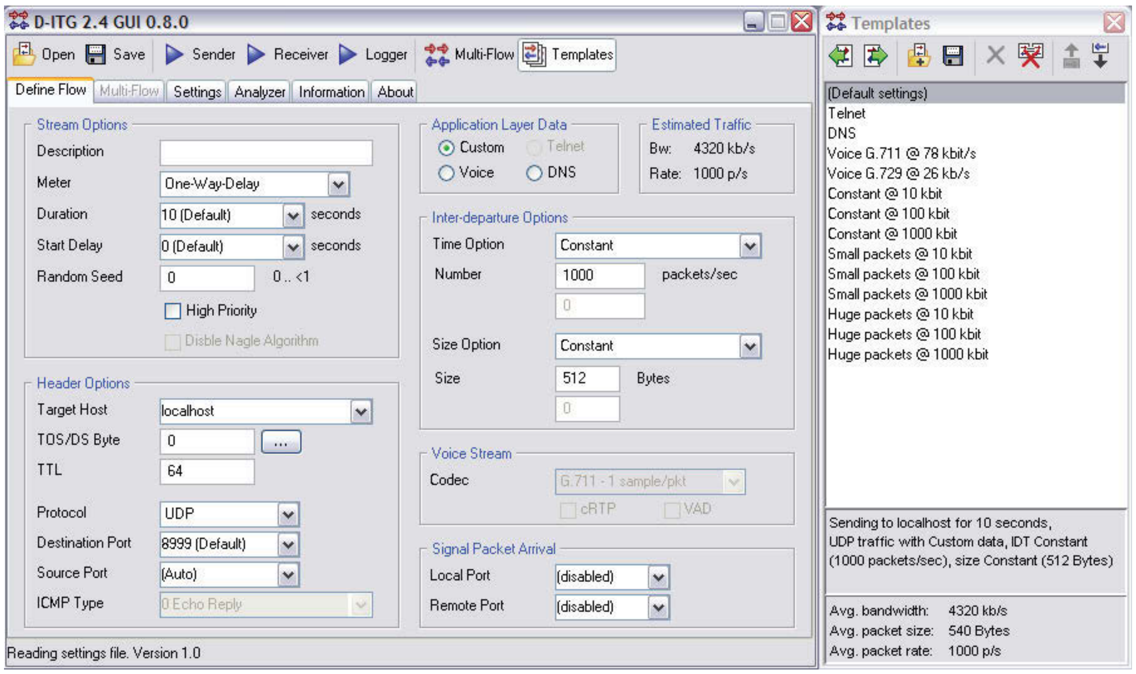
Determination of the type and quality of input data is significant for all data mining tasks. Insufficient data quality will prevent the algorithms from delivering correct output. The amount of data is another factor that should be taken into account. These are defined as

Data Type The type of data dealt with in the experiments are delay, jitter and packet-loss values generated by a traffic generator or a network simulator. The measurement unit of delay and jitter is seconds, while for packet-loss the value represents the number of lost packets in that time interval.

Data Volume The amount of the data handled in the experiments varies from small sets of logs provided by the UNINA to large logs generated by a network simulator, i.e. OPNET. More specifically, the logs volume ranges from minutes and fractions of an hour to days and fractions of a week in terms of the volume of the QoS data-sets.

The datasets are generated by D-ITG traffic generator or OPNET Modeler. As shown in Figure 3.3, D-ITG allows multiple streams to be generated with parameters to change characteristics of the flows, e.g. Diffserv/TOS markings, bandwidth per flow, constant vs. random packet sizing, source/destination port and duration of the traffic generated. The basic setup is a sender/receiver configuration where the sender is setup on a sender node to source the data streams and the receiver is installed on a client node to receive the data. There is also a logging module that can be used to store the delay, jitter and packet-loss values.

OPNET Modeler, on the other hand, allows users to analyse simulated networks and to compare the impact of various technology designs on end-to-end behavior. The network simulator provides a wide range of protocols and technologies for the design and the evaluation of communication networks, devices, protocols, and applications. The modeller



Stream Options
Description:
Meter: One-Way-Delay
Duration: 10 (Default) seconds
Start Delay: 0 (Default) seconds
Random Seed: 0 0 .. <1
High Priority
Disble Nagle Algorithm

Header Options
Target Host: localhost
TOS/DS Byte: 0
TTL: 64
Protocol: UDP
Destination Port: 8999 (Default)
Source Port: (Auto)
ICMP Type: 0 Echo Reply

Application Layer Data
[✓ Custom] Telnet Voice DNS

Estimated Traffic
Bw: 4320 kb/s
Rate: 1000 p/s

Inter-departure Options
Time Option: Constant
Number: 1000 packets/sec
0
Size Option: Constant
Size: 512 Bytes
0

Voice Stream
Codec: G.711 - 1 sample/pkt
cRTP VAD

Signal Packet Arrival
Local Port: (disabled)
Remote Port: (disabled)

Reading settings file. Version 1.0

(Default settings)
Telnet
DNS
Voice G.711 @ 78 kbit/s
Voice G.729 @ 26 kb/s
Constant @ 10 kbit
Constant @ 100 kbit
Constant @ 1000 kbit
Small packets @ 10 kbit
Small packets @ 100 kbit
Small packets @ 1000 kbit
Huge packets @ 10 kbit
Huge packets @ 100 kbit
Huge packets @ 1000 kbit

Sending to localhost for 10 seconds, UDP traffic with Custom data, IDT Constant (1000 packets/sec), size Constant (512 Bytes)

Avg. bandwidth: 4320 kb/s
Avg. packet size: 540 Bytes
Avg. packet rate: 1000 p/s

Figure 3.3: D-ITG GUI

includes a development environment to enable modeling of all network layers and types, and all technologies including VoIP.

3.2.2 Pattern Definition

As mentioned previously, two types of patterns are defined in this thesis: basic patterns and composite patterns. Basic patterns are directly defined from the change of time-series values, i.e. the trends time-series. The composite patterns are made up of triplet sequences of the basic pattern.

A DMF used for assigning the changes of time-series values (trends) to a specific basic pattern from a finite set of predefined basic patterns. Figure 3.4 shows a schema of the basic patterns defined in this thesis. The DMF function chooses one of the basic pattern in Figure 3.4 and assigns the basic pattern to the observed slope of the line between two

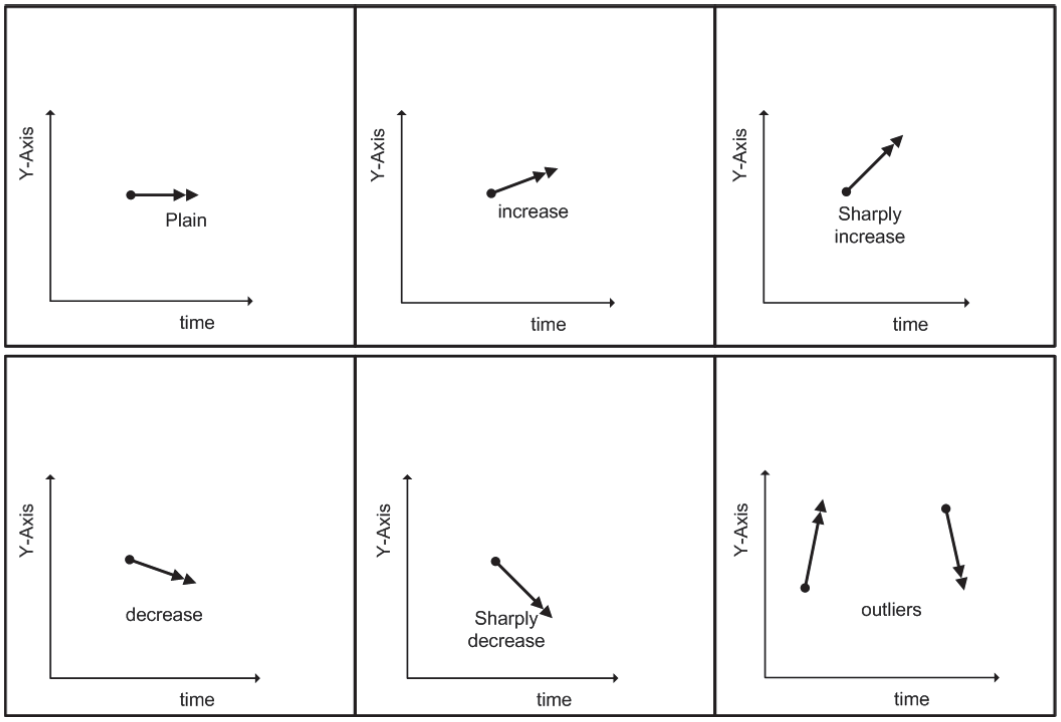
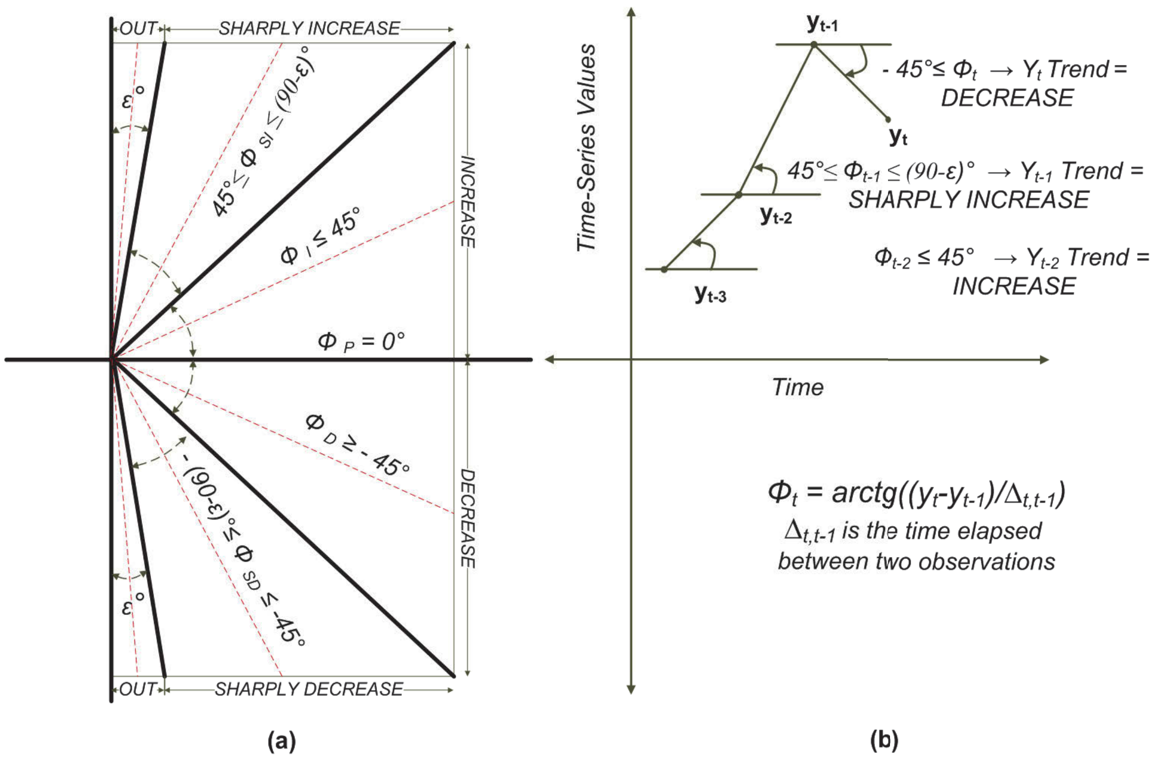


Figure 3.4: Basic Patterns

consecutive time-series values, y_{t-1} and y_t, called the Y_t trend. Composite patterns are adjacent $i - j - k$ triplet-trends defined based upon the basic patterns, i.e. each i, j and k are basic patterns.

The definition of the pattern in this thesis is based on the PDL definition discussed earlier in Section 2.4.2. Six basic patterns are defined, namely *sharply increase*, *increase*, *plain*, *decrease*, *sharply decrease* and *outlier*, for describing the trend of change in value of a time-series. The patterns are defined based on the various amounts of change in the values for two consecutive time-series values, i.e. $Y_t = y_t \ - \ y_{t-1}$, which is assigned to a basic pattern each time.

As shown in Figure 3.5 , according to the angles of the three consecutive trends at times $t-2$, $t-1$ and t categorical basic pattern are mapped to each of the numerical values of these trends, i.e. Y_{t-2}, Y_{t-1} and Y_t. Part (a) of Figure 3.5, depicts the categorical



OUT
SHARPLY INCREASE
ε°
45° ≤ Φ SI ≤ (90-ε)°
Φ I ≤ 45°
INCREASE
Φ P = 0°
Φ D ≥ - 45°
DECREASE
- (90-ε)° ≤ Φ SD ≤ -45°
ε°
OUT
SHARPLY DECREASE
(a)
Time-Series Values
y t-1
- 45° ≤ Φt → Yt Trend = DECREASE
yt
45° ≤ Φt-1 ≤ (90-ε)° → Yt-1 Trend = SHARPLY INCREASE
y t-2
Φt-2 ≤ 45° → Yt-2 Trend = INCREASE
y t-3
Time
Φt = arctg((yt-yt-1)/Δt,t-1)
Δt,t-1 is the time elapsed between two observations
(b)

Figure 3.5: Basic patterns assigned to triplet trends.

boundaries of basic patterns together with the angle internals assigned to each of them. ε is a model parameter set empirically to a small angle. Part (b) of Figure 3.5, gives a sample of composite pattern made up from three consecutive basic patterns, namely *increase*, *sharply increase* and *decrease*.

3.2.3 Look-up Table

As mentioned in previous subsection, the basic patterns are described in terms of linguistic variables by DMF. The function is defined as a conditional function resulting from human perception and is given by the deterministic rule sets of R_i: "if X is T_i then Y is S_i" where T_i and S_i can be either a value within an interval on the domain of real numbers or a trend categorical value assigned to the time-series changes, Y_t. For example, regarding the network expert's empirical knowledge about a network time-series of Y trends at times $t-2$, $t-1$ and t, a DMF rule can be as

$$R : \text{“IF } Y_{t-2} \text{ is } S_{t-2} \text{ AND } Y_{t-1} \text{ is } S_{t-1} \text{ then } Y_t \text{ is } S_t\text{”}, \tag{3.1}$$

where $Y_k s, t-2 \leq k \leq t$, are the trends of the time-series values in the time window k within intervals defined based on the window-size length in seconds. S_k are DMF categorical values assigned to the time-series of trend. These DMF categorical values are defined as *Plain, Increase, Sharply Increase, Decrease, Sharply Decrease* and *Outliers*. The next section presents a formal description of the proposed model.

Based on the definition of the basic patterns, a composite pattern is denoted within a window with three adjacent basic patterns, namely previous, current and next patterns. There are 216 composite patterns regarding the possible repeatable combinations of six basic patterns in each of the three consecutive patterns in the window.

3.3 Formal Model Description

The NARGES model, as presented in Figure 3.2, is a hybrid model that predicts the packet-loss at time $t+1$ based on the approximated values of delay and jitter at time t. The delay and jitter forecasts at time t, are approximations of the current values of the time-series forecasted by HDAX model according to the historical trends and values of the corresponding time-series in the previous two periods, i.e. $t-1$ and $t-2$. The following sections describe the two modules: HDAX and a multilayer perceptron, as the two constitutive parts of NARGES model.

3.3.1 Forecasting Module: HDAX

This section describes a novel approach to forecasting time-series values from previously observed patterns of delay and jitter. The HDAX algorithm is defined based on the model definition suggested by Tresp and Hofmann (1998).

Let y_t be the value of the discrete time-series at time t

$$y_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-N}) + \epsilon_t \tag{3.2}$$

where f is either known or approximated sufficiently well by a function approximator and ϵ_t is the zero-mean noise with probability density $P_\epsilon(\epsilon)$, which represents dynamics that are not modelled.

The underlying HDAX model of the time-series is made with order $N = 3$ as

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}) + \epsilon_t \tag{3.3}$$

Possible trends of the QoS time-series of delay and jitter values, at time intervals $t-1$ and $t-2$ are represented with categorical terms from the

$$Alphabet = \langle SI, I, P, D, SD, OUT \rangle \tag{3.4}$$

where these symbols are defined in Table 3.1. The basic trends in Table 3.1 are defined with linguistic variables based on the definition of the deterministic mapping function. Within the DMF mapping function, each of the categorical terms maps an interval on the domain of real numbers to a linguistic representative.

In Table 3.1, the y_t denotes the time-series value at time t and Y_t to denote the difference of the two consecutive values at time $t-1$ and t. For simplicity, the linear scale in the

Table 3.1: Deterministic Mapping Function (DMF), the scale of time-series trends used for mapping numerical traces to the categorical (linguistic) terms.

Case	Trend	Description	$Y_t = y_t - y_{t-1}$	Slope Angle
0	P	Plain	$Y_t = 0$	$\phi_P = 0°$
1	I	Increase	$0 < Y_t \leq \frac{max}{2}$	$0° < \phi_I \leq 45°$
2	SI	Sharply Increase	$\frac{max}{2} < Y_t \leq max - \epsilon$	$45° < \phi_{SI} \leq (90 - \varepsilon)°$
3	D	Decrease	$-\frac{max}{2} \leq Y_t < 0$	$-45° \leq \phi_D < 0°$
4	SD	Sharply Decrease	$-max + \epsilon \leq Y_t < -\frac{max}{2}$	$-(90 - \varepsilon)° \leq \phi_{SD} \leq -45°$
5	OUT	Outlier	$\|Y_t\| > max - \epsilon$	$\|\phi_{OUT}\| > (90 - \varepsilon)°$

experiment also has six linguistic grades. These are defined in Eq. (3.4) and represent categorical terms assigned to the case numbers of zero to five, respectively. The Slope Angle is the angle between the slope line and the time axis as shown in Figure 3.5.

The composite previous-current-next patterns are defined with a combination of three consecutive trends, at times $t-2$, $t-1$ and t. As as shown in Figure 3.6, these patterns are shown as $i - j - k$ patterns.

The proposed HDAX model approach consists of two phases: training and simulation. The max in Table 3.1 is the maximum value of a time-series, shown later with the dashed line in Figure 3.6.

3.3.1.1 HDAX Training

Figure 3.6 shows a sliding window that moves over the training data and makes patterns consisting of three consecutive trends, i, j and k, at times $t-2$, $t-1$ and t (previous, current and next) together with their frequency in the look–up table. The look–up table is then used in the simulation phase to approximate the next trend, at time t, and the associated delay/jitter value from the current and previously observed trend patterns, where i, j and k are the respective indices for previous, current and next trends ($0 \leq i, j, k \leq 5$) and $F(i, j, k)$ is their respective frequency. From the look–up table, the probability of

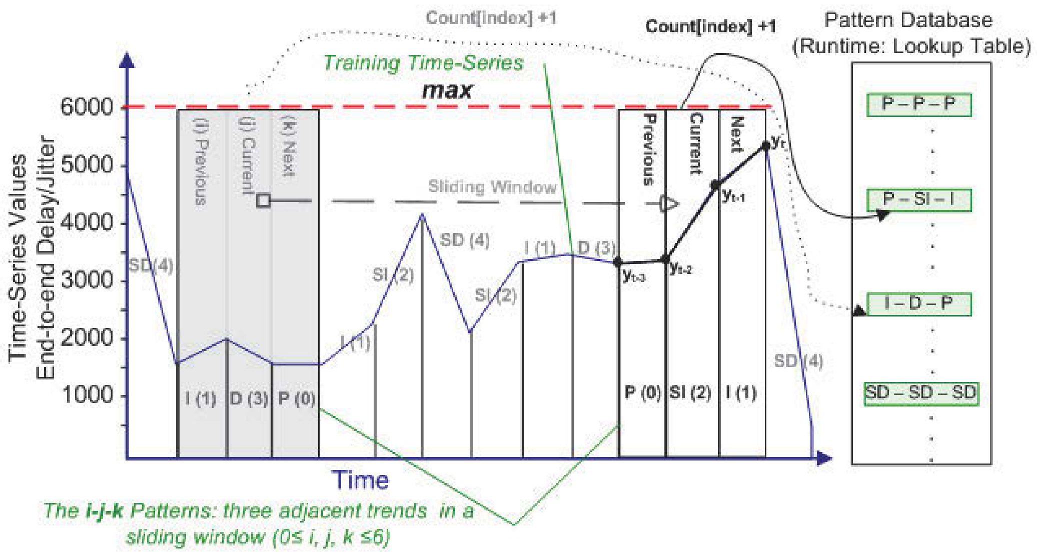


Figure 3.6: The training phase uses a time-series dataset values to recognise $i-j-k$ patterns and train the look–up table that maps each of these patterns to a respective frequency. The table is then used for forecasting the k trend at time $t+1$ in the simulation phase.

$i-j-k$ patterns is estimated as

$$\bar{P}(i,j,k) = \frac{F(i,j,k)}{N_k}, \quad N_k = \sum_{k=0}^{5} F(i,j,k). \tag{3.5}$$

where i, j and k are the indices for the respective patterns at times $t-1$, $t-2$ and t, and N_k is the number of total observations for all $i-j-k$ patterns.

3.3.1.2 HDAX Simulation

Based on the most frequently observed patterns in the last two consecutive trends at time $t-1$ and $t-2$, the HDAX algorithm uses the estimated conditional probability to approximate the trend at time t. The Y_t is a trend value at time t in the time-series of trends, defined based on the last two trends seen at times $t-1$ and $t-2$. Formally speaking, we estimate the $P_k(i,j)$ as follows

$$\bar{P}_k(i,j) = P(Y_t = k | Y_{t-1} = i, Y_{t-2} = j) + P_\epsilon(\epsilon) \tag{3.6}$$

where i and j are the indices of the observed trends at times $t-1$ and $t-2$, respectively. The trend at time t, with the index k, may take six possible values from the alphabets in Eq. (3.4). Based on this, the look-up table is used to forecast the next trend and value of the time-series based on the trend with highest frequency in this table:

$$i = t-1, j = t-2 : \hat{Y}_t = \arg\max_k (\bar{P}_k(i,j)) \tag{3.7}$$

With the trend k estimated, the $\widehat{y_t}$ can be approximated based on time step-size between y_{t-1} and y_t and the slope of the line at y_{t-1} using the Euler method.

3.3.2 Predictive Module: Multi-layer Perceptron

The predictive module calculates the average packet-loss. As described above, the outputs of HDAX are approximations of the values of the network traces at time t. They are used within a multilayer perceptron (MLP) to get better precision for real-time packet-loss prediction at time $t+1$.

The MLP output is computed as described in the following. The net value at j_{th} hidden layer(s) neuron can be obtained by

$$net_{h_j} = \sum_{i=1}^{n_i} (w_{h_{ji}} x_i + b_{h_j}), j = 1, 2, ..., n_h \tag{3.8}$$

where x_i is the input at i_{th} node of input layer, $w_{h_{ji}}$ is the connection weight of i_{th} neuron with j_{th} input, b_{h_j} is the weight of the bias at j_{th} neuron of hidden layer, n is the number of inputs (n=2 in the experiments), and n_h is the number of neurons in the hidden layer.

The output of the j_{th} neuron in hidden layer is

$$h_j = \phi(net_{h_j}) = 1/1 + e^{-net_{h_j}} \tag{3.9}$$

The net value of the k_{th} neuron then computed as

$$net_{y_k} = \sum_{j=1}^{n_h}(w_{o_{kj}} h_j + b_{o_k}), k = 1, 2, ..., n_o \tag{3.10}$$

where $w_{o_{kj}}$ is the output at j_{th} node of hidden layer with k_{th} output layer neuron, b_{o_k} is the weight of the bias at k_{th} neuron of output layer, n_h is the number neurons in hidden layer, and n_o is the number of neurons in output layer. That is, one in the experiments, as

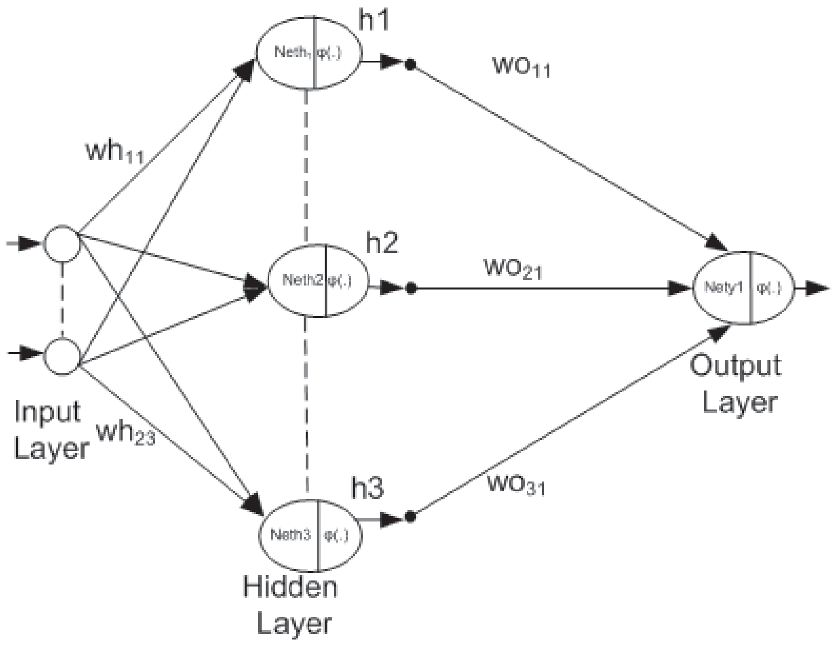


Figure 3.7: Multi-layer Perceptron

shown also in Figure 3.7.

3.4 Implementation Paradigm

This section describes the implementation of the proposed models. To obtain the optimum outcome, while using the proposed models with other datasets, certain parameters, such as windows size, might need regulations based on the characteristics of the new datasets.

3.4.1 Forecasting Module: HDAX

As discussed in previous sections, HDAX forecasts the delay/jitter values from observed patterns of these time-series, in two phases: training and simulation. The training phase utilises time-series values to recognise patterns and make the look–up pattern matrix (PDB) mapping the trend patterns to their observed frequency. A PDB is a runtime object made during an implementation phase from the concept of DMF described in Sections 3.2 and 3.3.

In this phase, a conceptual sliding window moves over the training data and counts composite patterns; made of three consecutive previous, current and next trends. PDB is then used in the simulation phase to predict the next trend and associated delay value from the current and previously observed trend patterns.

As well as the definition of the basic and composite patterns in Section 3.2.2, another notion in HDAX algorithm is a level function, which shows the level of delay observed at time t. This value is used to contrast two kinds of pattern frequency, in different contexts, as shown in Figure 3.8.

Figure 3.8 shows that we may have two (or more) contexts of having plain trend (labeled as P) in slope sequences. For instance, we may have two cases: a) when the value of y_t is near the minimum and b) when the value of y_t is near the maximum. Therefore,

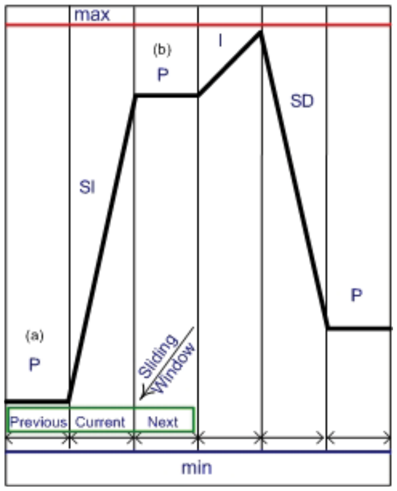


Figure 3.8: A sample string of symbolic values trend time-series $\{P, SI, P, I, SD, P\}$

we also need to store the delay level value (i.e. whether it is close to the minimum or maximum value) so as to have a finer classification between these two different cases. Consequently, a time-series level function $F(Y_t)$ is defined as

$$F(Y_t) = \begin{cases} 0 & \text{if } 0 \le Y_t < max/4 \\ 1 & \text{if } max/4 \le Y_t < 2 \times max/4 \\ 2 & \text{if } 2 \times max/4 \le Y_t < 3 \times max/4 \\ 3 & \text{if } 3 \times max/4 \le Y_t \le max \\ 4 & \text{if } max \le Y_t \end{cases} \tag{3.11}$$

The max parameter used in Figure 3.8 and Eq. (3.11) is defined by the network experts or, as is done in this work, it can be set to the max value for each dataset. In Table 3.2, each of the field types take a case number from Table 3.1 while the level function, fyt, take value from Eq. (3.11).

The pattern lookup-table (PDB) is implemented using a matrix with a row for each pattern and six columns as listed in Table 3.2. The PDB relates composite trends, listed in Table 3.1, to historical frequencies of these patterns within their corresponding matrix

Table 3.2: Description of the fields used for the pattern lookup-table implementation.

$FieldName$	$FieldType$	$Description$
$Index$	Integer	Index of the table
$Prev$	Categorical	A case number assigned to the *previous* trend from Table 3.1
$Current$	Categorical	A case number assigned to the *current* trend from Table 3.1
$Next$	Categorical	A case number assigned to the *next* trend from Table 3.1
fyt	Categorical	A case number assigned to the *level function* from Eq. (3.11)
$Frequency$	Long Integer	Observed frequency of a triplet $i-j-k$ pattern

entries. The next two subsections describe implementation of the training and simulation phases.

3.4.1.1 Training Phase

In this phase, a sliding-window moves along the training data to save the frequencies of three adjacent trends (previous, current and next trend) and increment the frequency value for each associated pattern within the PDB entries. Note that the *sliding window* here is a conceptual object for implementation. It is different from *transformation window* used for transformation of data mentioned in step 2 of the DM framework earlier. The *sliding window* length of $n = 3$ was chosen empirically as a balance between performance of the algorithm and generalisation of the composite trends.

For simplicity, the PDB is implemented as a MATLAB matrix, but alternatively it may be created as a physical table in a database. The PDB matrix contained 1080 rows and six columns, as there are alphabet containing six symbols (P, I, SI, D, SD, OUT) and five level function case in Eq. (3.11).

3.4.1.2 Simulation Phase

Once the PDB is populated by training it over a sufficiently long sequence of time-series values, it can be used to predict time-series trends and values for simulation time-series values.

In simulation phase, the previous three time-series average values are observed, to compute two adjacent previous and current patterns and then to forecast the next average time-series trend and value. The trends for the previous and current windows are calculated in the same way as described in the previous section.

The two consecutive (previous and current) basic patterns (Y_{t-2} and Y_{t-1}) are looked up in the PDB using a hashing function and the associated next pattern with the highest frequency is chosen as the expected next trend (i.e. Y_t).

The next time-series value y_t is estimated from Y_t (i.e. the next trend from the PDB) and the previous time-series value y_{t-1} using the $F(Y_t)$ function defined as

$$F(Y_t) = \begin{cases} y_{t-1} & \text{if } Y_t = 0 \\ y_{t-1} + max/4 - \sigma & \text{if } Y_t = 1 \\ y_{t-1} + 3 \times max/4 - \sigma & \text{if } Y_t = 2 \\ y_{t-1} - max/4 + \sigma & \text{if } Y_t = 3 \\ y_{t-1} - 3 \times max/4 + \sigma & \text{if } Y_t = 4 \\ y_{t-1} \pm max \pm \sigma & \text{if } Y_t = 5 \end{cases} \tag{3.12}$$

where y_{t-1} is the previous time-series value, max is a threshold for detecting the outliers and σ is the standard deviation of the y_t values in the simulation set. In the last condition of Eq. (3.12), for outlier values, the max and σ values are added if $y_{t-1} > max$, otherwise they are subtracted.

3.4.2 Predictive Module: Multi-layer Perceptron

In this section, the implementation of the predictive module of NARGES model is described. It is an MLP feed-forward network with back-propagation learning rule. The MLP has two input layer nodes fed by the forecasted delay and jitter, one hidden layer with ten neurons and predict one output, the number of packet-loss.

The multilayer perceptron network convergence time and its prediction accuracy, in terms of Normalised Root Mean Square Error (NRMSE), were tested to set the optimum parameters' value. These include number of hidden layer neurons, transformation windows size, packet size, learning rate and momentum. The rest of this section describes the network design and the MLP training algorithm.

3.4.2.1 Network Design

The MLP is a feed-forward with back-propagation learning rule. It has two input layer source nodes, namely forecasted delay (x_1) and forecasted jitter (x_2), Ten hidden layer neurons, and an output layer, the predicted packet-loss at time $t+1$. The following lines explain how the number of hidden layers and the number of hidden layer neurons are decided.

The model based on suggestions including those of Shiblee et al. (2009). According to the Kolmogorov theorem "*a single hidden layer is sufficient to approximate any continuous function*." Kurkova (1992) provides a proof for the Kolmogorov (1957) theorem. As a result, a three-layer MLP with one hidden layer is employed for the experiments. The parameters of MLP is designed and simulated using the MATLAB neural network toolbox.

The number of hidden neurons determined empirically by training the network with alternative values for each of the MLP network parameters. The difference between the observed values of packet-loss and the MLP output were calculated for making comparisons and decisions. The tests were done 10-50 times for each parameter value assigned to a single parameter while the other parameters were fixed to the default values.

For example, the MLP network was tested with various number of neurons in the hidden layer to reach the minimal average normalised root mean square error (NRMSE) within a reasonable training time.

The tests are repeated 15 times and a step-down recursive reject Holm-Sidak test is used for each group of the outputs assigned to a specific number of hidden layer neurons to decide the best number of neurons in the hidden layer. Using the MATLAB Holme-Sidak test an accept/reject criterion on a sorted set of null hypotheses is applied, starting from the lower $p-value$ and going up to the acceptance of null hypothesis.

Based on these experiments, the MLPs with ten hidden layer neurons showed the least NRMSE while the MLP with three neurons had the least training time. The Figure 3.9 shows a comparison between the accuracies and performances of MLPs with one to ten neurons in the hidden layers.

3.4.2.2 Training Algorithm

To train the designed MLP and get the best generalisation, the training algorithm fits the weight and bias values using Levenberg-Marquardt optimisation. The Levenberg-Marquardt optimisation process is based upon Bayesian regularization that minimises a synthesis of squared errors and weights, and then provides the correct combination of them to produce an optimally generalised predicting MLP. It modifies the linear synthesis to gain the best generalisation for the resulting network. For more details about Bayesian

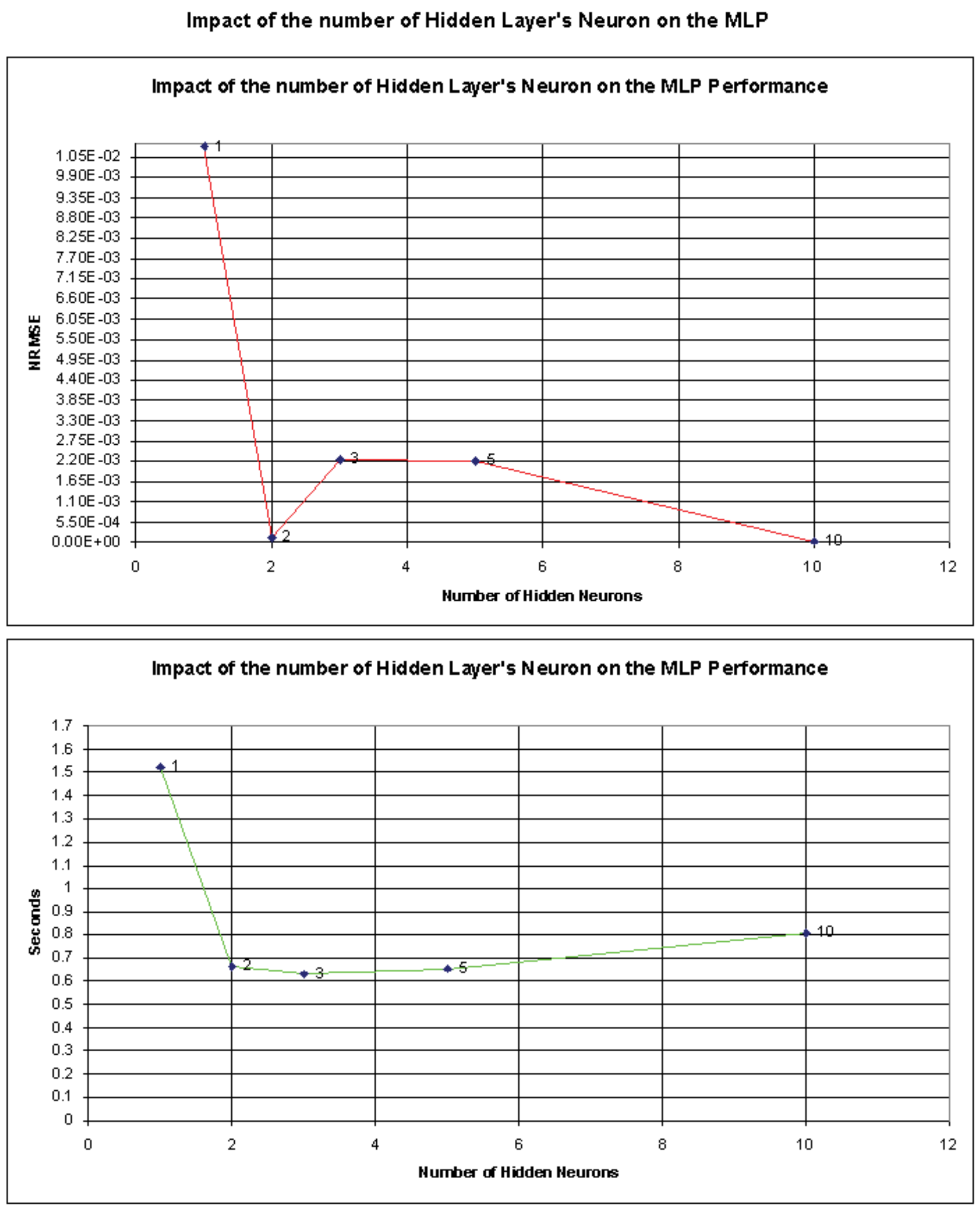


Figure 3.9: Impact of the number of Hidden Layer's Neuron on the MLP Accuracy and Performance

interpolation and generalisation the reader can refer to MacKay (1992) and Foresee and Hagan (1997).

3.5 Conclusions

The goal of this thesis is to design, implement and evaluate a data mining model for the real-time prediction of packet-loss variable based on the historical observations together with the forecasted delay and jitter. The model consist of two modules: one for approximating delay and jitter values at time t, HDAX, and the other, a multilayer perceptron, for predicting future packet-loss, at time $t+1$, based on these approximations. The predicted knowledge may be used later within a smart router — as the knowledge discovery task becomes intractable in the absence of real–time data to accomplish online data mining.

The chapter presented design and implementation of the proposed models. The HDAX and MLP models were formally described and an implementation paradigm of the models were presented in Sections 3.3 and 3.4. The HDAX model is designed and implemented for approximation of delay and jitter time-series. It is part of the NARGES Hybrid Data Mining model that uses HDAX outputs for prediction of the packet-loss. In this regard, this chapter fulfills the contribution 1 introduced in Chapter 1. Next, in Chapter 4, the HDAX and NARGES models outcomes will be compared to ARMA benchmark.

Chapter 4

Model Evaluation

"One of the great mistakes is to judge policies and programs by their intentions rather than their results."
— Milton Friedman

This chapter describes experiments for evaluating the accuracy and performance of the implemented prediction model. There is a comparison between HDAX and NARGES models and ARMA as a benchmark. QoS traces used in the experiments, were generated by D-ITG traffic generator or OPNET Modeler to test the impact of different end-to-end path and queuing policies on the proposed models, respectively.

The rest of this chapter is organised as follows. Section 4.1 overviews the benchmark and the evaluation methods. Section 4.2 describes the datasets and the tools generating them. Section 4.3 introduces the evaluation methodology. Next, in Section 4.4, experiment results are delivered. This section is divided to three subsections to study: the initial results of HDAX model, the results based on the shorter datasets end-to-end paths generated by D-ITG software, and the longer datasets based on the queuing policies generated by running simulations on OPNET. Section 4.5 summarises the results for HDAX and NARGES. Section 4.6 concludes the chapter.

4.1 Evaluation Benchmark and Measurements

In this section, the benchmark and the evaluation methods are discussed. The Box, Jenkins, and Reinsel's ARMA algorithm is described first, which is the benchmark used throughout this thesis. Next, in subsections the evaluation measurements are introduced, namely the error, speed and quality measurements.

4.1.1 ARMA Benchmark

Autoregressive Moving Average, as an alternative numerical forecasting algorithm to the thesis proposed models, is implemented based upon the definition in Box et al. (2011). An abbreviation of $ARMA(p, q)$ is used in the literature for the mixed autoregressive moving average model with p autoregressive and q moving average terms. As shown in Chatfield (2001), given a time-series of data y_t where t is an integer index (indicating time intervals in the experiments) and the y_t are discrete real numbers, then an $ARMA(p, q)$ model is written as

$$\left(1 - \sum_{i=1}^{p} \phi_i\right) y_t = \left(1 + \sum_{i=1}^{q} \theta_i\right) \epsilon_t \tag{4.1}$$

where p and q are the orders and the ϕ_i and θ_i are the coefficients of the autoregressive and the moving average parts, respectively. The ϵ_t are residuals in prediction, defined by the following recurrent expression where $\hat{y}_t$ and y_t are respective forecasted and target time-series values at time t.

$$\epsilon_t = |\hat{y}_t - y_t| \tag{4.2}$$

The version of ARMA used here has four parameters as its input and generates one step ahead of the entered ARMA time-series. The four input parameters are the time-series value y_t, the ARMA coefficients ϕ and θ, and a control argument $yesmean$ that takes the value 1 or 0 depending on whether the mean is subtracted from the input time-series or not.

Since the ARMA model is autoregressive, it has to be computed by the impulse response, that is, the coefficients of the equivalent moving average representation. The p and q orders of the ARMA sequence, y_t, may either be estimated by minimising the criterion as suggested in Hannan and Rissanen (1982) or be set empirically.

For the experiments run in this thesis, the p and q orders of the ARMA time-series are set empirically by choosing the parameters that minimise the algorithm's error and run-time. A better fit for the parameters ϕ and θ is estimated by minimising the logarithm of Residual Sum of Squares (RSS) between the actual and predicted values over a window w

$$\log \sum_{t=1}^{w} {\epsilon_t}^2 \tag{4.3}$$

where ϵ_t is the forecasting error at time t. Based on this, the optimum values of ARMA coefficients were set.

4.1.2 Error Measurement

To compare the HDAX and NARGES outputs with the target time-series values, a distance function is defined to measure their similarity. For two time-series $y = y_1, \ldots, y_n$

and $\hat{y} = \hat{y_1}, \ldots, \hat{y_n}$ with the same length of n, the Euclidean distance between y and $\hat{y}$ is

$$D(y, \hat{y}) = \sqrt{\sum_{t=1}^{n} (y_t - \hat{y_t})^2} \tag{4.4}$$

In the experiments, the distance function is calculated to estimate the accuracy of the algorithms. The model results are compared to that of ARMA to test the performance and to identify the optimal parameter settings in the model. The NRMSE measure is used to compare the errors within the time-series imputed to the model outputs. It is calculated based upon the root mean square error (RMSE) formula and the standard deviation of target time-series. The RMSE is defined as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (y_t - \hat{y_t})^2}{n}} \tag{4.5}$$

where y_t and $\hat{y_t}$ are the target time-series, of delay, jitter or packet-loss, together with a time-series of the imputed values, respectively.

According to Wu and Mencer (2009), the statistical range of target time-series values may be used instead of standard deviation in calculation of NRMSE. Thus, the NRMSE is computed as

$$NRMSE = \frac{RMSE}{max(y_t) - min(y_t)} \tag{4.6}$$

4.1.3 Speed Measurement

The timing function in MATLAB measures actual time elapsed (wall-clock time) and is, therefore, more sensitive to properties specific to the platform. The results were reported here for execution times in seconds and all determined using CPU-time.

4.1.4 Quality Measurement

Normalised cross-correlation coefficients are used to measure the quality of the HDAX and NARGES outputs compared with those made by ARMA. The cross-correlation sequence and algorithm is also described in Wilcox (2012) and Orfanidis (1985) as

$$R_{xy}(l) = E[x_{t+l} * y_t] = E[x_t * y_{t-l}] \tag{4.7}$$

where x_t and y_t, $-\infty < t < \infty$, are two time-series and E denotes the expected value operator. A MATLAB cross-correlation function approximates the above sequence, because in practice only a finite segment of one realisation of the infinite-length of a random process is available. The lagged correlation between the two time-series with no normalisation is calculated as

$$\hat{R}_{xy}(l) = \begin{cases} \sum_{t=0}^{n-l-1} x_{t+l} * y_t & l \geq 0 \\ \hat{R}_{yx}(-l) & l < 0 \end{cases} \tag{4.8}$$

The output vector CCF has elements given by $CCF(l) = \hat{R}_{xy}(l-n), l = 1, \ldots, 2n-1$ where n is the number of values in time-series.

In general, the correlation function requires normalisation to produce an accurate estimate. As stated in Chatfield (2004), the cross-correlation for two random process of x_t and y_t formula is calculated by the following lagged correlation formula

$$\rho_{xy}(\tau) = \frac{\gamma_{xy}(\tau)}{\sigma_x \sigma_y} \tag{4.9}$$

where τ is the sets of lags $[-N+1, N-1]$, σ_x and σ_y are the variances and $\gamma_{xy}(\tau)$ is the covariance of two random process x_t and y_t.

The covariance is calculated as

$$E\Big[(x_t - \mu_x)(y_{t+\tau} - \mu_y)\Big] \tag{4.10}$$

The maximum cross-correlation coefficient is then computed as

$$\rho_{max} = \arg\max_{\tau} \Big(\rho_{xy}(\tau)\Big) \tag{4.11}$$

4.2 Datasets

Each of the datasets consists of three QoS time-series: delay, jitter and packet-loss. For the evaluation of experimental results, three sets of data are considered: the target data generated by D-ITG or OPNET, the output of HDAX (or NARGES) and the output of ARMA. In the experiments, each dataset is divided into two training and test datasets with a portion of 25% and 75% of the measured data, respectively. The three QoS traces were fed into the NARGES model in the way described in Chapter 3.

In each experiment, the target data of delay and jitter time-series are fed into HDAX to forecast the delay and jitter at time t while it is proposed that the traces comes with a unit of time delay. Next, these forecasted delay and jitter time-series are fed into a Multilayer Perceptron as part of the NARGES model to predict the value of the packet-loss at time $t+1$. This is stored as a time-series of predicted packet-loss values, and is then used in the analysis of the accuracy of NARGES. Figure 4.1 shows how a target dataset, including delay, jitter and packet-loss time-series, together with the time-series of forecasted delay and jitter are used as the inputs of HDAX and MLP modules within NARGES DM model to predict the packet-loss.

The datasets were categorised: (i) according to the seven End-to-End Path (e2eP) and (ii) according to the three queuing policies used. Thus, as it can be seen in Tables 4.1 and 4.3, there are ten categories of datasets.

Forty-six datasets were used for computing results. Each dataset includes time-stamped traces, time-series, of delay, jitter and loss values. There are 36 datasets from D-ITG with an average 3000 values in each of delay, jitter and packet-loss time-series. There are also 10 datasets generated by OPNET, 9 with 48000 values and one with 1,000,000 values.

In the following, the datasets will be described based on the tools used to generate them, namely D-ITG traffic generator and the OPNET Modeler.

4.2.1 D-ITG Datasets

The data generated by D-ITG is gathered in two ways: firstly, archives from the University of Naples Federico II (UNINA) are used and secondly archives generated and probed over University of Technology, Sydney (UTS) network test-bed.

As mentioned before, these archives have different kinds of network traffic in tcpdump format. The archives are in compressed rar format containing files, in which they include traces of QoS parameters measured over seven end-to-end path types. The considered QoS parameters are packet-loss, delay, and jitter.

The traces were generated by sending probe packets using two packet rates (128 pps and 11000 pps) and a packet size of 1024 bytes. Depending on the name of the file, it contains delay, jitter, or packet-loss samples in text format. In each file, the first column indicates a reference time, i.e. the time passed from the first packet, and the second and third columns include the sample value in that time window. Samples were obtained by

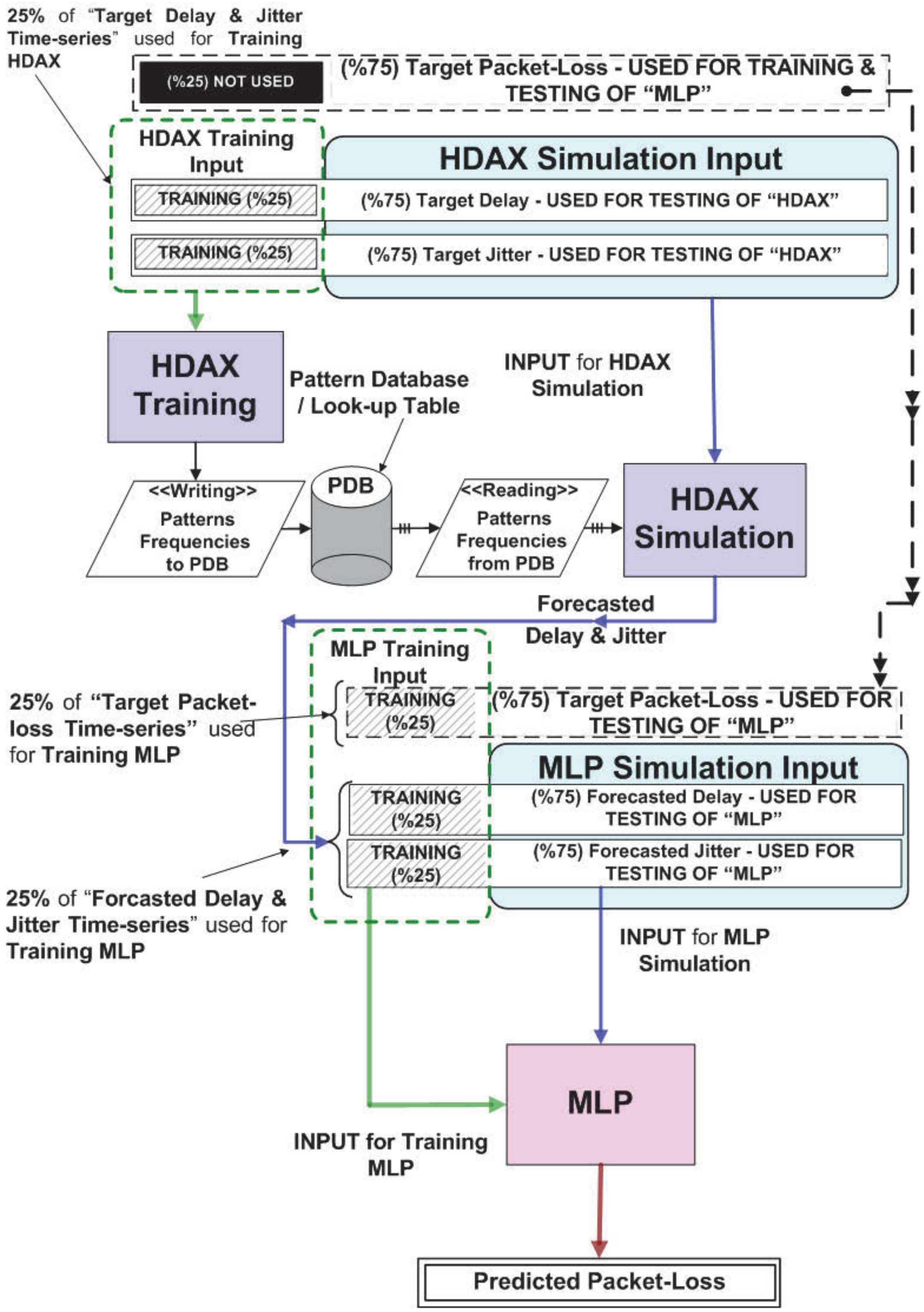

25% of "Target Delay & Jitter Time-series" used for Training HDAX
(%25) NOT USED
(%75) Target Packet-Loss - USED FOR TRAINING & TESTING OF "MLP"
HDAX Training Input
HDAX Simulation Input
TRAINING (%25)
(%75) Target Delay - USED FOR TESTING OF "HDAX"
TRAINING (%25)
(%75) Target Jitter - USED FOR TESTING OF "HDAX"
HDAX Training
Pattern Database / Look-up Table
INPUT for HDAX Simulation
<<Writing>> Patterns Frequencies to PDB
PDB
<<Reading>> Patterns Frequencies from PDB
HDAX Simulation
Forecasted Delay & Jitter
MLP Training Input
25% of "Target Packet-loss Time-series" used for Training MLP
TRAINING (%25)
(%75) Target Packet-Loss - USED FOR TESTING OF "MLP"
MLP Simulation Input
TRAINING (%25)
(%75) Forecasted Delay - USED FOR TESTING OF "MLP"
TRAINING (%25)
(%75) Forecasted Jitter - USED FOR TESTING OF "MLP"
25% of "Forcasted Delay & Jitter Time-series" used for Training MLP
INPUT for MLP Simulation
MLP
INPUT for Training MLP
Predicted Packet-Loss

Figure 4.1: Target dataset (target delay, jitter and packet-loss time-series) together with the HDAX forecasted data (forecasted delay and jitter time-series) are used in NARGES model to predict future packet-loss.

using an active measurement approach, sending probe packets by using D-ITG with a packet rate of 100 pps.

The measurement unit of the delay and jitter is milliseconds and for packet-loss the value represents the average number of lost packets in that time window. More description and details about the D-ITG traces characteristics, network test-bed, software architecture and the experimental regulations will be covered in the following subsections of the thesis.

4.2.1.1 D-ITG Data Characteristics

The D-ITG traffic generator (Botta et al., 2012) maintains various protocols including TCP, UDP, ICMP, SCTP, DNS, Telnet, VoIP (G.711, G.723, G.729, Voice Activity Detection and Compressed RTP). It provides stochastic processes where both packet size (PS) and inter departure time IDT have Constant, Uniform, Exponential, Pareto, Cauchy, Normal, Poisson or Gamma distribution.

The D-ITG archives, used in this thesis, have uniform distribution for the size of the packets and for the inter packet time and UDP, TCP and SCTP protocols are used over heterogeneous networks (Botta et al., 2008). D-ITG also provides setting of random generation seeds which permit repeating of experiments. Measurement of OWD and RTT, packet-loss, jitter and throughput is performed on receiver-side. However, the software is also capable of storing datasets about sent and received traffic on both sender and receiver sides.

The traffic generator provides Type Of Service (TOS), Time to Live (TTL) and packet size settings. The duration and start delay of experiments can be set based upon the initial time of the experiment. Moreover, a separate signaling channel shown by a red-dashed-double-dots lines in Figure 4.2, demonstrates the communication between sender and receiver as well as the log writer threads. D-ITG platform exhibits a distributed multi-component architecture. Figure 4.2 shows a graphical overview of the relationship among the main components of D-ITG platform.

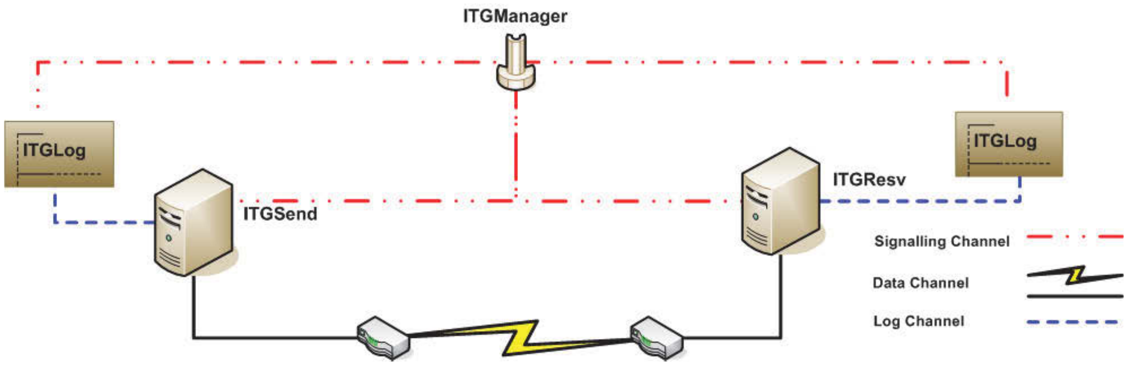


Figure 4.2: D-ITG framework

The End-to-End Path (e2eP) definition was considered for the datasets obtained from D-ITG. The definition is based on the one stated in Botta et al. (2007). More detailed information about the definition and the characteristics of end-to-end paths is in Table 4.1.

Table 4.1: Characteristics of the end-to-end paths for the data obtained from D-ITG.

Dataset Category	Access Networks	Transport Protocol	Operating Systems	End Users Device	Number of Datasets
1	ADSL-Ethernet	UDP	Linux-Linux	PC-WS	2
2	GPRS-Ethernet	UDP	Windows-Linux	Laptop-WS	6
3	UMTS-Ethernet	UDP	Windows-Linux	Laptop-WS	6
4	Ethernet-ADSL	UDP	Linux-Linux	WS-PC	6
5	Ethernet-GPRS	UDP	Linux-Windows	WS-Laptop	7
6	Ethernet-UMTS	UDP	Windows-Windows	WS-Laptop	2
7	Ethernet-WLAN	UDP	Linux-Windows	WS-Laptop	7

In summary, there are 36 datasets time-series for D-ITG generated over two test-beds of UNINA and UTS. Each of the datasets is categorised into one of seven dataset categories shown in Table 4.1.

4.2.1.2 D-ITG Network Test-beds

As stated before, testbeds from UNINA and UTS were used for generating datasets.

The UNINA heterogeneous network test-bed is described in Botta et al. (2008). The QoS traces on UNINA website were employed to provide data for running the thesis experiments. Botta, Pescapé, and Ventre generated the QoS traces using D-ITG over a real test-bed depicted in Figure 4.3.

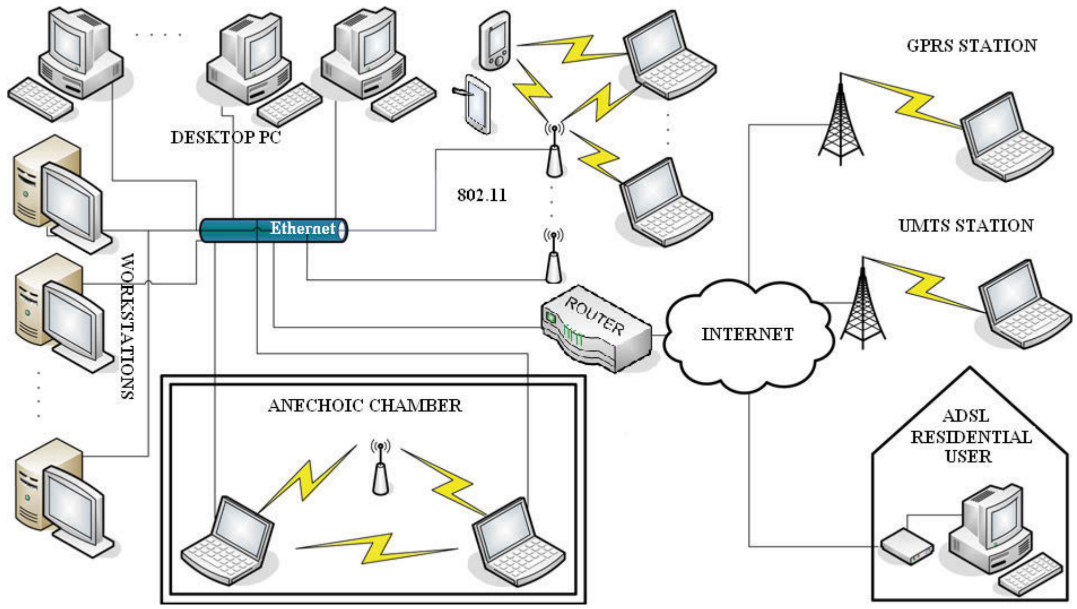


Figure 4.3: The University of Naples Federico II (UNINA) experimental test-bed used to generate D-ITG traces. (adapted from Botta et al. (2008))

The UTS network test-bed was formed between two nodes within the Faculty of Engineering and Information Technology (FEIT) LAN. These are: (i) a laptop with processor rating of 1.70 GHz, running MS Windows with Intel PRO/Wireless 2200BG (IEEE 802.11b/g) network connection and (ii) a node on the FEIT High Performance Computing (HPC) Linux Cluster, running Red Hat Enterprise Linux 6 (64bit) with processor rating between 3.06-3.46GHz and a gigabit inter-node connection. A 100Mbs LAN used to access UTS cluster via the intermediate switches/routeres between two nodes used for D-ITG QoS data generation as shown in Figure 4.4.

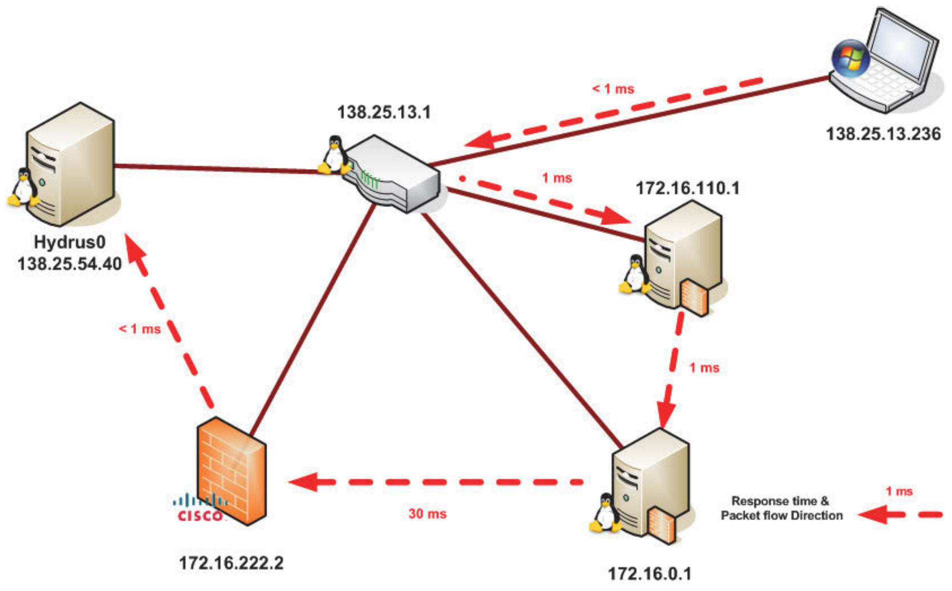


Figure 4.4: Trace Route between the two nodes used for D-ITG QoS data generation

4.2.1.3 D-ITG Software Architecture

The communication between sender and receiver in the D-ITG software is controlled by a separate signalling channel using a protocol called Traffic Specification Protocol (TSP). The three components of D-ITG architecture shown in Figure 4.2 are

ITGSend is the sender component of the D-ITG platform. It can operate in three different modes: (i) Single-flow mode, (ii) Multiple-flow mode, (iii) Daemon mode. ITGSend stores information either locally or remotely using the ITGLog.

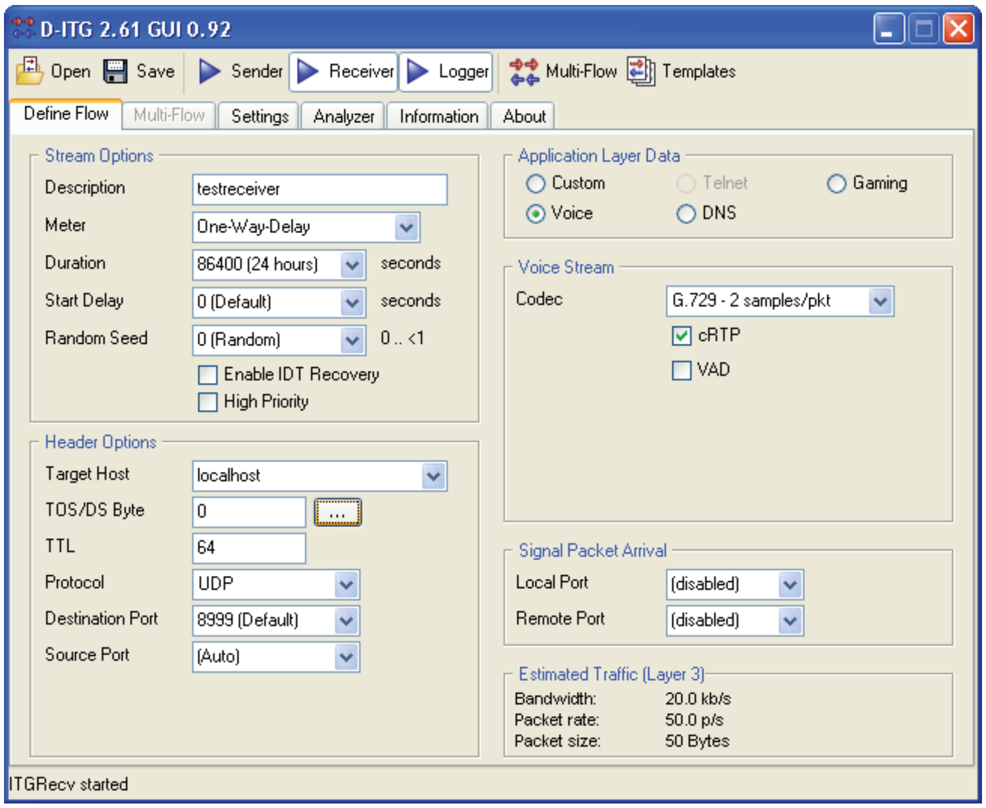
ITGRecv is the receiver component of the D-ITG platform. It works as a concurrent daemon, listening for TSP connections and generates a thread responsible for the management of the communication with the sender. Each single flow is received by a separate thread. Like ITGSend, ITGRecv can store information either locally or remotely by using the log server ITGLog.

ITGLog can be either a log server, running on a different host than ITGSend and ITGRecv, or a thread running on the same sender/receiver sides. It is capable of receiving and storing log information from multiple senders and receivers.

The logging activities is handled using a signaling protocol. This protocol allows each sender/receiver to register on, and to leave, the log server. The log information can be sent using either a reliable channel (TCP) or an unreliable channel (UDP).

4.2.1.4 Parameter Settings for D-ITG

The voice application was set for application layer data with G.711 and G.729 codec using UDP packet header compression. The duration used for generating datasets was 30 seconds. The datasets were generated in two sets of experiments for VoIP UDP transmission with quiet or peak network activity. Traffic logs were generated using Best Effort (BE) or Expedited Forward (EF) header options.



Stream Options

Description	testreceiver
Meter	One-Way-Delay
Duration	86400 (24 hours) seconds
Start Delay	0 (Default) seconds
Random Seed	0 (Random) 0 .. <1

☐ Enable IDT Recovery
☐ High Priority

Header Options

Target Host	localhost
TOS/DS Byte	0
TTL	64
Protocol	UDP
Destination Port	8999 (Default)
Source Port	(Auto)

Application Layer Data

○ Custom ○ Telnet ○ Gaming
● Voice ○ DNS

Voice Stream

Codec	G.729 - 2 samples/pkt

✓ cRTP
☐ VAD

Signal Packet Arrival

Local Port	(disabled)
Remote Port	(disabled)

Estimated Traffic (Layer 3)

Bandwidth:	20.0 kb/s
Packet rate:	50.0 p/s
Packet size:	50 Bytes

ITGRecv started

Figure 4.5: D-ITG GUI setup

There were archived datasets available from the UNINA's MagNets network backbone and those obtained on UTS network:

UNINA The archives are related to several end-to-end paths (datasets categories 1-7) as shown in Table 4.1. These datasets contain samples of measured QoS parameters over four end-to-end paths. Samples were obtained using non-overlapping windows of 30-50ms length. The generated archives are kept in tar.gz format, each of which contains sample datasets of QoS parameters - packet-loss, delay and jitter.

The datasets are generated by adopting an active measurement approach and sending probe packets of 64, 512 and 1024 bytes with a packet rate of 100 packets per second. For each generation experiment for production of OWD, RTT, packet-loss and jitter traces it is possible to initialise the random variables seed.

A non-stationary stochastic QoS time-series of TCP, UDP and SCTP probe packets were used. The packets were sent at regular time intervals (every one second) to characterise the end-to-end packet delay behaviour of the Internet. The time-series are ON/OFF background traffic sources, calculated using non-overlapping windows of 10ms length, for wired, wireless and ADSL network.

UTS The archive is related to the seventh dataset category as shown in Table 4.1. Simple unicast constant traffic with constant packet rate (PR) and payload size (PS) was generated using D-ITG considering QoS parameters, namely packet-loss, delay and jitter.

The traffic generator was set to be capable of filling a dedicated 100 Mbps Ethernet between the UTS Clusters and a laptop, which were using Redhat Linux and Windows operating systems, respectively.

4.2.2 OPNET Datasets

OPNET Modeler provides a discrete event simulation engine test-bed to generate datasets using FIFO, PQ and WFQ queuing policies. The aim of generating these datasets is to study the effect of the packet transmission policies and network congestion states on the performance of the implemented data mining models. Selection of the service discipline in the routers can affect VoIP applications and link congestion. Consequently, the model performance is evaluated using datasets generated under FIFO, PQ and WFQ packet forwarding policies. Table 4.2 compares OPNET, OMNET and NS2 network simulators (Antoniou et al., 2002).

Table 4.2: Comparison between OPNET, OMNET and NS2

Criteria	**NS-2**	**OPNET**	**OMNET++**
Programming Model	Object-oriented; event-driven simulator; written in C++; Front-end: OTcl interpreter.	Object-oriented; full access to source code; written in standard C (Proto-C).	Discrete event simulator
Operating System Supported	UNIX & Windows	UNIX & Windows	Linux & Windows
Model Hierarchy Levels	Not supported	Hierarchically Structured - Three distinct levels: network level; node level; and process level	OMNeT++ models may consist of hierarchically nested modules.
Simulation Modes	Discrete simulation mode	Simulation Modes: Discrete, Analytic, and Hybrid	Simulation Modes: Discrete, Analytic, and Hybrid
Documentation	Poor	Excellent	Good
Validating, Debugging and Tracing	Large user community; open source code; Supports: memory, Tcl, C++, and mixed Tcl/C++ debugging and off-line animation.	Large user community; open source code for protocol models; Supports: command-line debugging and off-line animation.	Small user community for validating models; Supports: Debugging and Tracing.
Running Large Networks	Limitations in memory and CPU time	Limitation in virtual memory capacity	Limitation in virtual memory capacity
Parallel Execution	Supported	Supported	Supported

Although there are other open source network modellers and simulators such as NS2 or OMNET++ for academia, the OPNET Modeler was used to generate the network traces due to more accuracy in simulating the queuing behavior (Schilling, 2005; Potemans et al., 2003; Varga, 2001). OPNET Modeler also facilitates the network analysis and modeling process with a wide range of devices, protocols and applications that improve the evaluation results.

Except for one of the simulations, simulation times were 30 minutes and 48000 values were generated for each of the delay, jitter and packet-loss time-series within the target dataset files. The exception was one of the FIFO simulations, which was run for 12 hours and generated 1,000,000 values for each of QoS time-series. Each of the time-series was exported from spreadsheet data into a text file. The text file data has two columns: a time-stamp and a value. The metric for delay and jitter time-series was seconds while for the packet-loss it was the number of packets dropped. The number of packets sent in one second was 24 packets.

4.2.2.1 OPNET Data Characteristics

A personal laptop with processor rating of 1.70 GHz was used on the UTS LAN, running MS Windows with Intel PRO/Wireless 2200BG (IEEE 802.11b/g) network connection and an individual version OPNET Modeler under academic research license.

Three more datasets categories as shown in Table 4.3 were considered for OPNET generated datasets based upon the queuing policies.

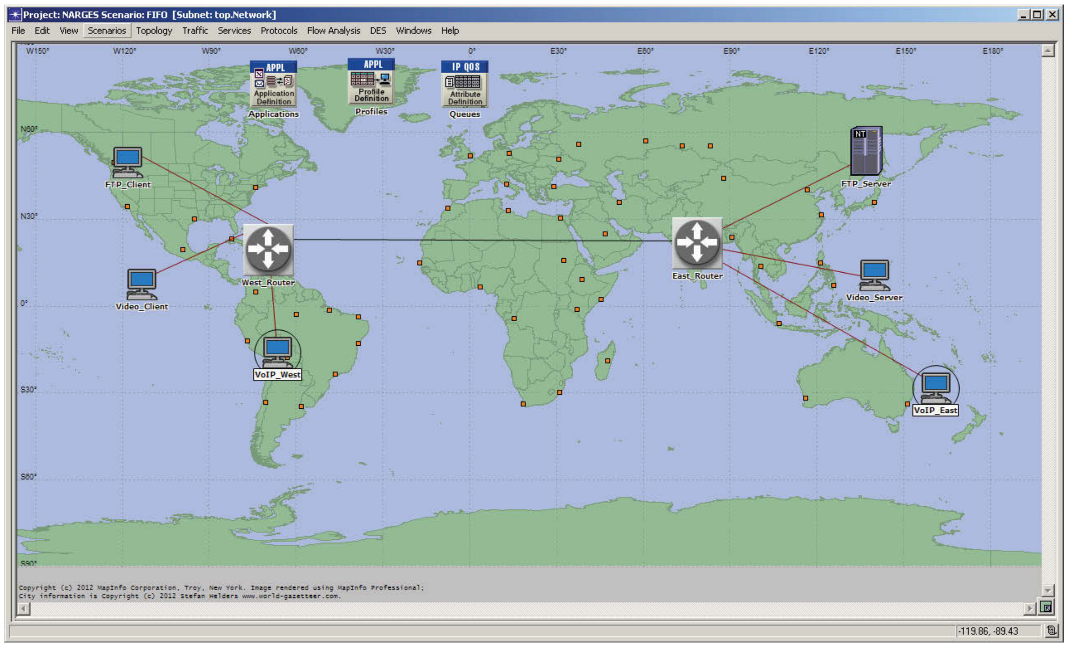
Table 4.3: Types of Queueing Policies for the data obtained from OPNET.

Dataset Category	Queueing Policy	Description	QoS Enabled	Number of Datasets
8	WFQ	Weighted Fair Queuing	Yes	3
9	FIFO	First in First out	No	4
10	PQ	Priority Queuing	Yes	3

In summary, there are 10 datasets generated in OPNET simulations and each of the datasets is categorised into one of three dataset categories shown in Table 4.3. As mentioned in the start of this section, the dataset categories described in Table 4.3 are extensions to those in Table 4.1.

4.2.2.2 OPNET Network Test-bed

The inter-network design suggested by Aboelela (2011) was implemented. As shown in Figure 4.6, there were five *ethernet workstations* used along with an *ethernet server* and two *ethernet4_slip_gtwy* edge routers in the east and the west. There were *10Base_T* links connecting the workstations and the servers to one of the routers and a bidirectional PPP_DS1 link between the east and west routers.



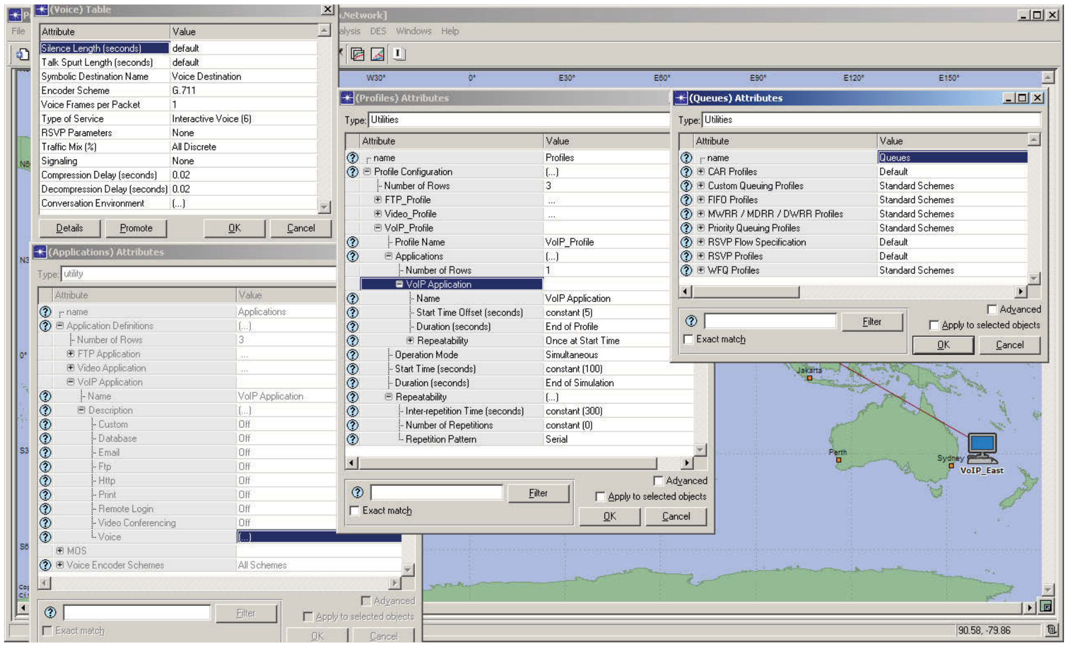
Copyright (c) 2012 MapInfo Corporation, Troy, New York. Image rendered using MapInfo Professional;
City information is Copyright (c) 2012 Stefan Helders www.world-gazetteer.com.

Figure 4.6: OPNET network design used for QoS data generation

In these simulations, each router controls the queueing strategy of packet transmissions which can impact both the transmission rate and the packet delay.

4.2.2.3 Parameter Settings for OPNET Modeller

The experiments were designed to evaluate the model predictions in various link traffic states, namely quiet, congested or bursty. The east and west routers used BGP protocols. Three types of applications, namely File Transfer Protocol (FTP), Video and VoIP applications were simultaneously used to mimic the real–world network conditions for all the simulations. Next, the delay, jitter and packet-loss time-series for VoIP applications were collected and used for the experiments in Section 4.4.3.


File
alysis DES Windows Help
W30° 0° E30° E60° E90° E120° E150°
Jakarta
Perth
Sydney
VoIP_East
Details Promote OK Cancel
Filter Exact match Advanced Apply to selected objects
90.58, -79.86

(Voice) Table

Attribute	Value
Silence Length (seconds)	default
Talk Spurt Length (seconds)	default
Symbolic Destination Name	Voice Destination
Encoder Scheme	G.711
Voice Frames per Packet	1
Type of Service	Interactive Voice (6)
RSVP Parameters	None
Traffic Mix (%)	All Discrete
Signaling	None
Compression Delay (seconds)	0.02
Decompression Delay (seconds)	0.02
Conversation Environment	(...)

(Applications) Attributes

Type: utility

Attribute	Value
name	Applications
Application Definitions	(...)
Number of Rows	3
FTP Application	...
Video Application	...
VoIP Application	
Name	VoIP Application
Description	(...)
Custom	Off
Database	Off
Email	Off
Ftp	Off
Http	Off
Print	Off
Remote Login	Off
Video Conferencing	Off
Voice	(...)
MOS	
Voice Encoder Schemes	All Schemes

(Profiles) Attributes

Type: Utilities

Attribute	Value
name	Profiles
Profile Configuration	(...)
Number of Rows	3
FTP_Profile	...
Video_Profile	...
VoIP_Profile	
Profile Name	VoIP_Profile
Applications	(...)
Number of Rows	1
VoIP Application	
Name	VoIP Application
Start Time Offset (seconds)	constant (5)
Duration (seconds)	End of Profile
Repeatability	Once at Start Time
Operation Mode	Simultaneous
Start Time (seconds)	constant (100)
Duration (seconds)	End of Simulation
Repeatability	(...)
Inter-repetition Time (seconds)	constant (300)
Number of Repetitions	constant (0)
Repetition Pattern	Serial

(Queues) Attributes

Type: Utilities

Attribute	Value
name	Queues
CAR Profiles	Default
Custom Queuing Profiles	Standard Schemes
FIFO Profiles	Standard Schemes
MWRR / MDRR / DWRR Profiles	Standard Schemes
Priority Queuing Profiles	Standard Schemes
RSVP Flow Specification	Default
RSVP Profiles	Default
WFQ Profiles	Standard Schemes

Figure 4.7: Applications profile setting on OPNET Modeller

There were two types of setting in the experiments: general and specific settings. The general settings were mostly based on the default suggested by OPNET or Aboelela (2011).

Specifically, three categories of datasets were generated on OPNET Modeler by simulating FIFO, PQ and WFQ queuing packet transmission disciplines. As shown in Figure 4.7, by setting the VoIP, Video and FTP application profiles within OPNET Modeler, various network loads are simulated. VoIP streams with the G.711 Encoder Scheme was used in all of the simulations. Constant inter-request time and simultaneous operation mode with inter-repetition time of 300 seconds and one repeatability at the start of each experiment were set for all applications.

4.3 Evaluation Methodology

Non-parametric statistical analysis used to interpret the results of the data mining model prediction and its respective forecasting module, namely NARGES and HDAX.

Three factors were used to evaluate the outputs made by the proposed models and algorithms in this thesis. The accuracy and correlation of the outputs of HDAX and NARGES models were compared to ARMA together with the speed of the algorithms. Besides, the proposed models were ranked in comparison to ARMA for each of these parameters and tested for the hypothesis of similarity. Friedman tests were conducted to test the hypothesis of similarity of the models and ARMA. $p-value$ for the Holm's test was considered to reject or accept these hypotheses.

Results were plotted using boxplot and parameters such as median, Interquartile Range (IQR), maximum and minimum values are considered in the analysis of boxplots. The boxplots are depicted for target values together with the outputs of HDAX or NARGES models and ARMA. For each of the experiments three sets of data are considered: (i) the *target datasets* generated by D-ITG or OPNET, (ii) the *output* of HDAX or NARGES, and (iii) the *output* of ARMA. Each of these outputs are labeled under corresponding boxplots in the rest of this chapter.

Pearson cross-correlation coefficient is used for similarity measurement between the target delay, jitter or loss traces and the model outputs. Two synchronised series of target and forecasted values with finite length of N are used to calculate the maximum of Cross–Correlation Function between each pair of these traces. The Cross–Correlation Function was calculated by keeping the target time-series unlagged and the forecasted time-series is delayed by lag l.

To standardise the results from CCF function lags of $l \leq 50$ of are considered. The function normalises sequences to compute lagged correlation with maximum correlations of 1 at zero lag. The maximum similarity of normalised cross-correlations is reported for correlation analysis between the two time-series. MATLAB stemplots show the CCF results over a fixed number of 101 lags, which are reduced to improve the visibility of the plot.

As mentioned earlier, the maximum cross correlation coefficient, CCF max is considered in the stemplots as a factor of the similarity of an algorithm output time-series to the target time-series of delay, jitter and packet-loss. The higher CCF max of a model is, the more similar the model outputs were to the target time-series.

4.4 Experiments and Results

In this section, experimental results are presented based upon the way each dataset was generated: First, in Section 4.4.1, the results of HDAX is compared versus ARMA using D-ITG datasets; delay time-series from these datasets are used only. Next, the results of HDAX and NARGES are compared to those of ARMA using D-ITG datasets and the datasets generated by OPNET simulation in Sections 4.4.2 and 4.4.3, respectively. The results were published in Homayounfard and Kennedy (2009) and Homayounfard et al. (2013). The **bold-printed** numbers or algorithm names distinguish statistically significant results.

4.4.1 Experiment 1: Approximating Delay Time-Series with HDAX

In these series of experiments, only HDAX results were compared to ARMA to evaluate the model accuracy and speed. The experiment was repeated to optimise the parameter settings. Setting the parameters were done based on the optimum results and speed obtained. In terms of data, for the experiments with HDAX and ARMA, the UNINA D-ITG datasets were used. Specifically, non-zero mean delay time-series of one wired network traffic archive with the packet size of 64 bytes were used from seventh dataset category in Table 4.1.

As described in previous section 4.2 and shown in Figure 4.1, datasets were divided for experimental training and simulations phases, respectively. In this regard, 25% of the delay time-series is used for training and populating HDAX trends lookup table and 75% of it used for simulation. The training data is used for the training phase to build and populate the PDB table of patterns as described in Chapter 3.

Referring to the step 2 of NARGES conceptual framework shown in Figure 3.1, the target dataset has to be transformed before using them as the inputs for HDAX and ARMA experiments. A *transformation window* of equal length 10 values were used. The average values of the delay values within each transformation window are used as the inputs of HDAX and ARMA. The transformation window length of 10 was decided based upon the D-ITG archived data characteristic only for the experiment with D-ITG datasets.

The window length for other real–world network scenarios may be decided based on the respective characteristics of the network traffic, such as the number of the packets sent in a second. For a real–time simulation a trade–off between the algorithm overhead and the performance factors to define the optimum transformation window length.

There were 4 simulation runs for each of datasets. Based on these runs, the overall NRMSE was calculated using Eq. (4.6). The target and forecasted delay values for each of the four HDAX and ARMA simulation runs are shown in Figures 4.8 to 4.11. These

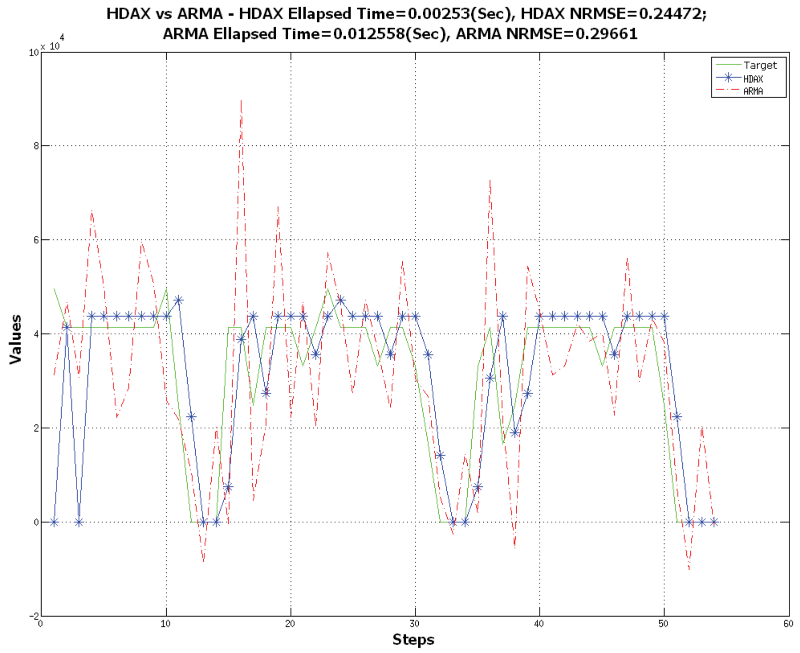


Figure 4.8: Target (solid line), HDAX predicted (star-dashed line) and ARMA predicted (dot-dashed line) delay values for simulation run 1.

results show that the predicted delay values are close to the expected values in most cases. Figures 4.8 to 4.10 show that HDAX sometimes predicts a higher delay than expected when there are sharp increases. Similarly, Figures 4.9 and 4.10 show lower than expected delay values when there are sharp decreases in delay value.

For the experiments in the next two subsections, the phase shift difference between HDAX forecasts and target values, shown in 4.11, have been reduced by refining the implementation of the algorithm.

As shown in Table 4.5, As can be seen in Table 4.5, the HDAX approximates faster than ARMA. Moreover, HDAX showed an NRMSE error of 11.27% while for ARMA the NRMSE error was 17.82%. Thus, HDAX showed better speed and accuracy within initial experiments in comparison to ARMA.

In the two following subsections 4.4.2 and 4.4.3, the HDAX and ARMA algorithms

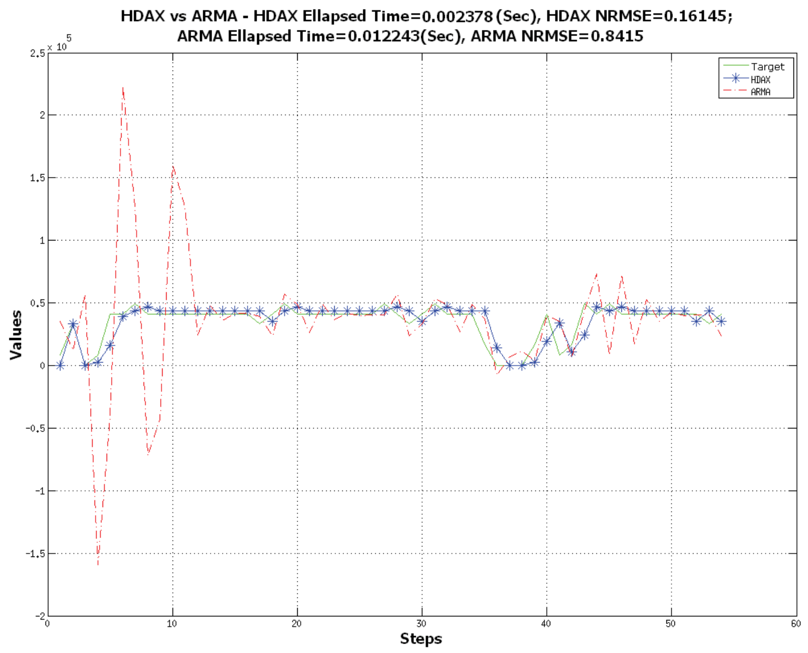


Figure 4.9: Target (solid line), HDAX predicted (star-dashed line) and ARMA predicted (dot-dashed line) delay values for simulation run 2.

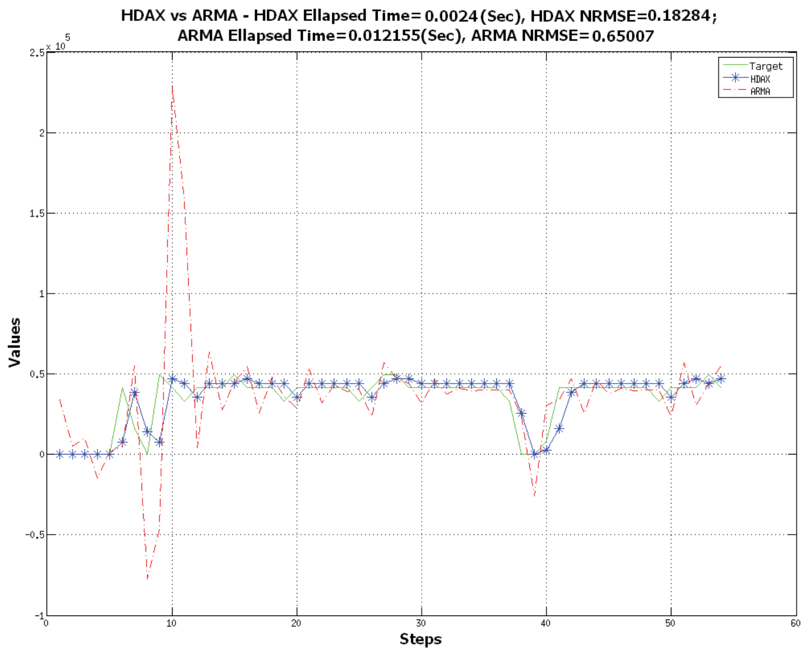


Figure 4.10: Target (solid line), HDAX predicted (star-dashed line) and ARMA predicted (dot-dashed line) delay values for simulation run 3.

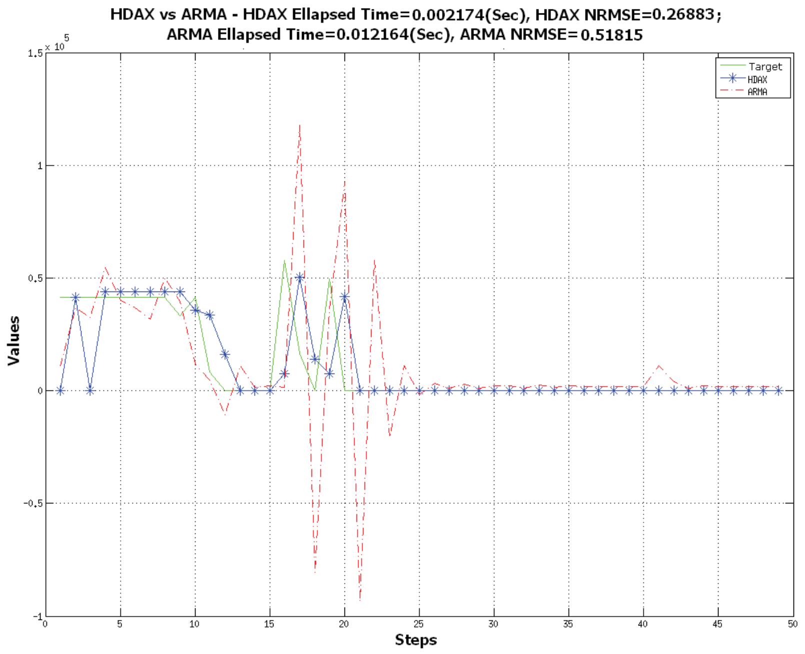


Figure 4.11: Target (solid line), HDAX predicted (star-dashed line) and ARMA predicted (dot-dashed line) delay values for simulation run 3.

Table 4.4: Accuracy of HDAX and ARMA (benchmark) on first phase of simulation runs together with speed of algorithm.

Simulation Run	HDAX NRMSE	Speed (sec)	ARMA NRMSE	Speed (sec)
1	0.245	0.003	0.297	0.013
2	0.161	0.002	0.842	0.012
3	0.183	0.002	0.650	0.012
4	0.269	0.002	0.518	0.012

ran with all the archives generated by D-ITG and those data obtained though OPNET Modeler simulations.

Table 4.5: Accuracy of HDAX and ARMA (benchmark) in the phase two of simulation runs together with speed of algorithm.

Model Name	Model NRMSE Error (%)	Speed (sec)
HDAX	11.27	0.002
ARMA	17.82	0.012

4.4.2 Experiment 2: Impact Analysis of End-to-End Path with Various Network Congestion Level on Model Predictions

In these series of experiments, the datasets are generated on the D-ITG network testbeds with various background loads – from a quiet network activity with a zero mean packet-loss to a busy network with bursty packet-loss.

As shown in Table 4.1, there are seven paths used to generate the QoS traces from the UNINA archives. The D-ITG network topologies were described in Section 4.2.2.2. The paths were categorised based upon either the sender and receiver protocols or network topologies, namely ADSL, GPRS, UMTS, WLAN and Ethernet.

4.4.2.1 Model Results with D-ITG Datasets

The accuracies of NARGES model and its HDAX subcomponent were examined by comparing the error function NRMSE. Thirty six simulations were conducted in these experiments with the target datasets, which have been generated by D-ITG over the UNINA and UTS network testbeds.

In Tables 4.6 and 4.7, the respective results of forecast error, algorithms speed and maximum cross-correlation coefficient (CCF) between the output of the algorithms and the target data – including delay, jitter and packet-loss time-series – are presented. The NRMSE show the error percentage for the algorithms, the speed is measured in seconds and the CCF max column report the maximum cross-correlation coefficients. The numbers printed in **bold** specify algorithms which are more accurate, faster or have more correlated output.

Table 4.6 reports the results of HDAX, and ARMA over delay and jitter time-series from the D-ITG datasets. As mentioned above, the bolted values in Table 4.6 mean that the corresponding model, HDAX or ARMA, had a better average result in the experiments for the reported Dataset Category and the performance parameter in each row of the table. The number of experiments for each Dataset Category is equal to the Number of Datasets in Table 4.1.

Table 4.6: Normalised root mean square error (NRMSE) together with algorithms speed and cross-correlation coefficients of HDAX and ARMA forecasts for D-ITG delay and jitter time-series.

		Forecasted Delay			**Forecasted Jitter**		
$Model$ $Name$	$Dataset$ $Category$	$NRMSE$ (%)	$Speed$ (sec)	CCF max	$NRMSE$ (%)	$Speed$ (sec)	CCF max
$HDAX$	1	**3.99**	0.010	**0.997**	3.96	0.011	**0.945**
$ARMA$	1	6.89	**0.009**	0.938	**3.07**	**0.009**	0.902
$HDAX$	2	14.50	**0.003**	**0.872**	4.22	**0.004**	**0.763**
$ARMA$	2	**12.94**	0.006	0.833	**3.18**	0.006	0.721
$HDAX$	3	8.42	**0.003**	**0.935**	5.82	**0.004**	**0.913**
$ARMA$	3	**7.54**	0.009	0.854	**3.99**	0.007	0.786
$HDAX$	4	**4.75**	**0.010**	**0.993**	4.67	0.010	**0.984**
$ARMA$	4	7.03	0.013	0.929	**4.12**	**0.009**	0.781
$HDAX$	5	10.05	**0.003**	**0.779**	8.26	**0.004**	0.592
$ARMA$	5	**6.19**	0.008	0.768	**3.40**	0.006	**0.771**
$HDAX$	6	5.72	**0.003**	**0.981**	5.70	**0.004**	**0.869**
$ARMA$	6	**5.05**	0.005	0.796	**4.61**	0.007	0.800
$HDAX$	7	**3.14**	0.008	**0.997**	5.04	**0.009**	0.717
$ARMA$	7	7.54	0.008	0.945	**2.44**	0.011	**0.772**

The predicted average packet-loss for NARGES was also compared to ARMA. As Table 4.7 demonstrates, in most cases the NARGES model predicts more precisely than ARMA but is slower. This is because NARGES has more modules and processes more data than ARMA to predict the final packet-loss values. Again, the **bold** numbers show the algorithms with better performance in terms of NRMSE, speed or maximum CCF.

4.4.2.2 Model Comparison

In this section, a comparison between the accuracy, performance (speed) and the correlation of the output of the models with the target data is performed via nonparametric Friedman tests. The tests were conducted with the results from the runs of HDAX, ARMA

Table 4.7: Normalised root mean square error together with speed of calculation and cross-correlation coefficients of NARGES and ARMA predictions for D-ITG packet-loss time-series.

$Model$ $Name$	$Dataset$ $Category$	$NRMSE$ (%)	$Model\ Speed$ (sec)	CCF max
$NARGES$	1	**0.61**	1.364	0.999
$ARMA$	1	2.37	**0.009**	0.999
$NARGES$	2	0.00	0.263	1.000
$ARMA$	2	0.00	**0.004**	1.000
$NARGES$	3	3.34	2.534	0.998
$ARMA$	3	**1.72**	0.006	**0.999**
$NARGES$	4	**0.36**	0.307	0.999
$ARMA$	4	1.34	**0.007**	0.999
$NARGES$	5	**0.85**	1.999	**0.999**
$ARMA$	5	5.34	**0.006**	0.689
$NARGES$	6	0.00	1.611	0.995
$ARMA$	6	0.00	**0.005**	**1.000**
$NARGES$	7	0.00	0.203	1.000
$ARMA$	7	0.00	**0.008**	1.000

and NARGES models over three sets of delay, jitter and loss data of D-ITG datasets.

Friedman test used as a non-parametric equivalent to the parametric repeated measures Analysis Of Variance (ANOVA) test. It computes the ranking of the measured outputs for an algorithm with other algorithms, assigning the best of them the ranking 1 and the worst the ranking k. According to the null hypothesis, it is supposed that the results of the algorithms are equivalent and the rankings are also similar.

Friedman tests were run and the statistic to store the $p-value$ used in Holm's procedure to reject or accept the null hypothesis of the similarity of the algorithms. Friedman tests also calculate the average ranking of the algorithms used in each tests. Table 4.8 shows the algorithm average ranking for each series. Table 4.8 shows that the HDAX ranking is better than ARMA for the accuracy of the results and the speed of the algorithms whereas ARMA ranking is better than HDAX for cross-correlation between the forecasted and target time-series.

Table 4.8: Average rankings as calculated using Friedman test for the results of the algorithms for accuracy, speed and cross-correlation (CCF) over delay, jitter and packet-Loss time-series. The algorithms with **bold** rank number have better ranking in each row.

$Time - series$	$Test$	$ARMA$	$HDAX$
$Delay$	$Accuracy$	1.250	**1.750**
	$Speed$	1.333	**1.667**
	CCF	**1.944**	1.056
$Jitter$	$Accuracy$	1.417	**1.583**
	$Speed$	1.389	**1.611**
	CCF	**1.583**	1.417
$Time - series$	$Test$	$ARMA$	$NARGES$
$Packet - Loss$	$Accuracy$	1.458	**1.542**
	$Speed$	**2.000**	1.000
	CCF	1.500	1.500

In the following tables, two algorithms are significantly different if their corresponding average ranks differ by at least the critical difference, which are the $p - value$s ≤ 0.05. In Table 4.9, the algorithm name shown in each row is taken as the better when the null hypothesis is rejected. Table 4.9 shows testing the algorithms for accuracy, speed and maximum CCF, for delay, jitter and packet-loss time-series. Algorithm names printed in **bold** have statistically significantly better results.

Table 4.9: Holm / Hochberg Table for $\alpha = 0.05$ (**bold** *algorithm* names).

$Data$	$Test$	$Algorithm$	$z = (R_0 - R_i)/SE$	$p - value$
$Delay$	$Accuracy$	**HDAX**	3.000	0.0027
$Delay$	$Speed$	**HDAX**	1.999	0.0455
$Delay$	CCF	**ARMA**	5.333	9.64×10^{-8}
$Jitter$	$Accuracy$	HDAX	1.000	0.317
$Jitter$	$Speed$	HDAX	1.333	0.182
$Jitter$	CCF	ARMA	0.999	0.317
$Packet - loss$	$Accuracy$	NARGES	0.499	0.617
$Packet - loss$	$Speed$	**ARMA**	6.000	1.97×10^{-9}
$Packet - loss$	CCF	NARGES	2.67×10^{-15}	0.999

According to the $p - value$s, HDAX forecasts significantly better and faster than ARMA for delay traces while ARMA has more correlated outputs in comparison to the target delay data in the 36 runs. Two algorithms, HDAX and ARMA, perform the same in forecasting jitter values because Holm's procedure accepts all null hypotheses. In terms of accuracy and quality of the predictions over D-ITG datasets, although Table 4.8 shows higher ranking grade for NARGES model, it predicts the packet-loss as accurate as ARMA does with equivalent CCF results. In terms of the speed of the models, the $p - value$ of the ARMA speed is less than the significance level ($\alpha = 0.05$). this suggests that ARMA is faster. NARGES model showed slower than ARMA in 36 runs with D-ITG datasets while the number of inputs NARGES are greater than ARMA.

4.4.2.3 Discussion on the Quality of Results

This section presents a discussion about the distributions of the delay, jitter and packet-loss time-series together with the models corresponding outputs over one of datasets, as a representative of the model results. As stated in Section 4.3, Box and whisker plots (boxplots) are used to compared the target time-series values distribution with the distribution of the results generated by HDAX, NARGES and ARMA.

Figures 4.12, 4.14 and 4.16 show the distribution of the target values and the outputs of HDAX, NARGES and ARMA models. Figures 4.13, 4.15, 4.17 show the stemplots for the respective results generated by HDAX, NARGES model and ARMA models.

4.4.2.4 HDAX

The boxplot in Figure 4.12 shows that target data was spread between 30 and 50 seconds, with IQR of 20 seconds, minimum of 0 second, maximum of 58 and median of 40 seconds. HDAX forecasts have more similar distribution to target delay time-series compared to ARMA. HDAX also shows less outliers compared to ARMA which has a narrower distribution with an IQR about 5 seconds, a minimum and maximum between 20 and 40 seconds and extra outliers.

As shown in Figure 4.13, in the CCF stemplot for ARMA and HDAX forecasted delay

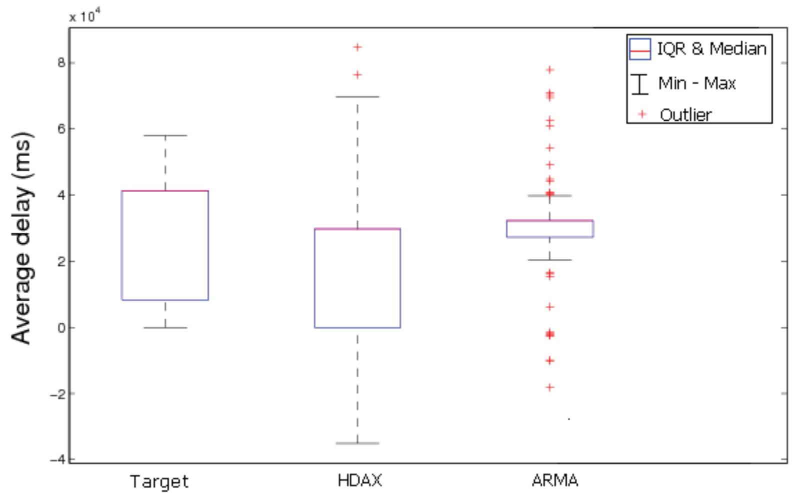


Figure 4.12: Boxplots of distributions of target delay time-series for dataset 13 together with those for outputs of HDAX and ARMA

time-series, the HDAX forecasted time-series

had a slightly higher maximum cross-correlation coefficient. It means that HDAX forecasts had a better correlation with the target delay time-series than those of ARMA.

For jitter values, as shown in Figure 4.14,

the target data has a median of 5 ms, minimum 0 and maximum 7 ms with a 2 ms IQR while HDAX shows a slightly wider IQR of 4 ms. Although ARMA has a slightly closer median to the target values, HDAX is more accurate in approximating the outliers. As shown in Figure 4.15, HDAX forecasts had a maximum cross-correlation coefficient of 0.953 and was more correlated to target jitter time-series while ARMA had a $CCF\ max$ of 0.794.

In this regard, the HDAX shows better correlation and closer distribution to the target data.

In summary, ARMA forecasts are slightly more accurate, but less correlated to target jitter time-series. Although ARMA results show a closer median to the median of target values distribution, HDAX forecasts are more correlated to target delay time-series. Moreover,

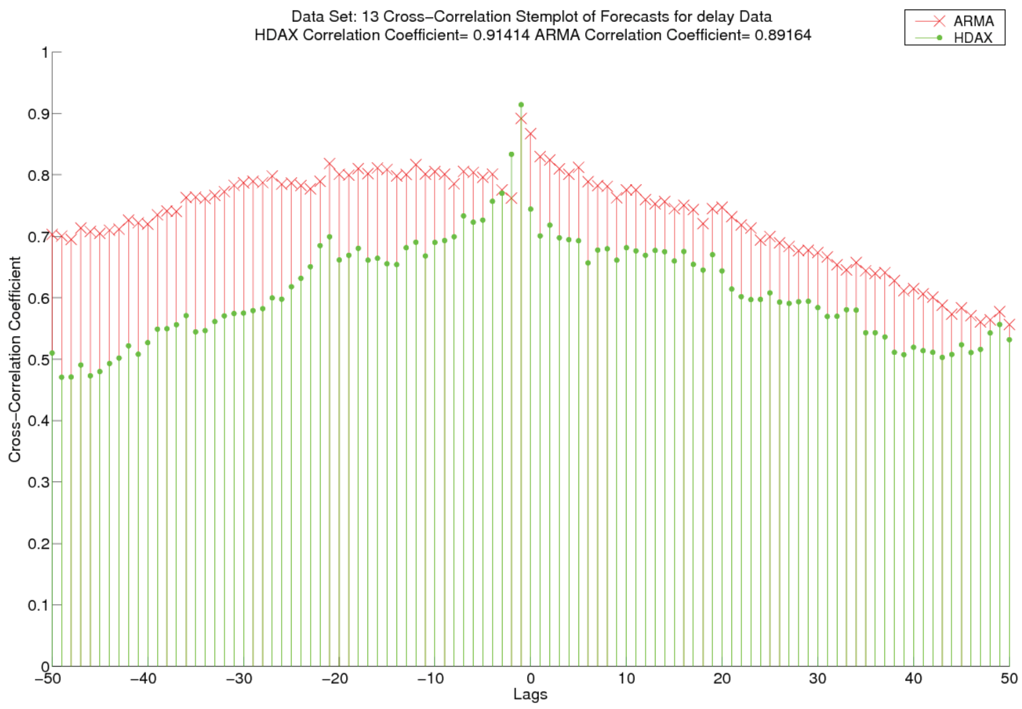


Figure 4.13: Stemplots of cross-correlation of HDAX forecasts and target delay time-series for dataset 13 together with those of ARMA

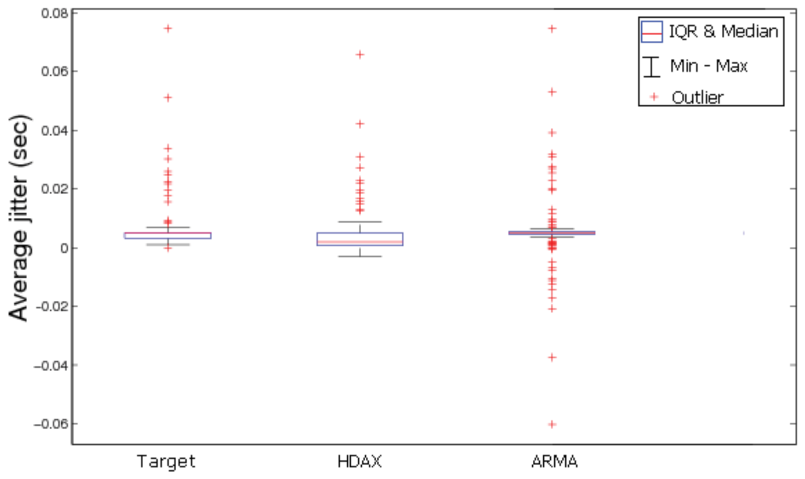


Figure 4.14: Boxplots of distribution for target jitter time-series within dataset 13 together with those for outputs of HDAX and ARMA

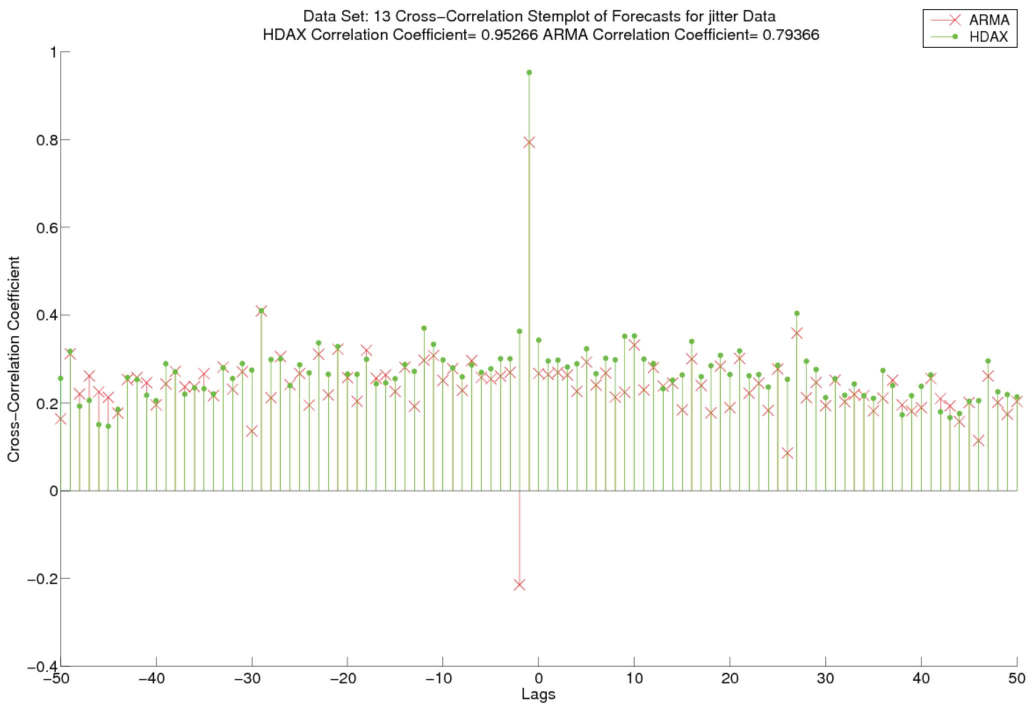


Figure 4.15: Stemplots of cross-correlation of HDAX forecasts and target jitter time-series for dataset 13 together with those of ARMA

HDAX has a better approximation of the outliers for both delay and jitter time-series, which are valuable in detecting the anomalies and congestion in the partial network.

4.4.2.5 NARGES

As shown in Figure 4.16, all the target values as well as NARGES and ARMA model predictions show a zero mean distribution.

The boxplots for the Target, NARGES and ARMA distributions in Figure 4.16and the stemplots in Figure 4.17 represents boxplots with a minimum zero packet-loss and a zero median lines. Compared to ARMA, NARGES has a better prediction of outliers while ARMA is less accurate in detecting outliers.

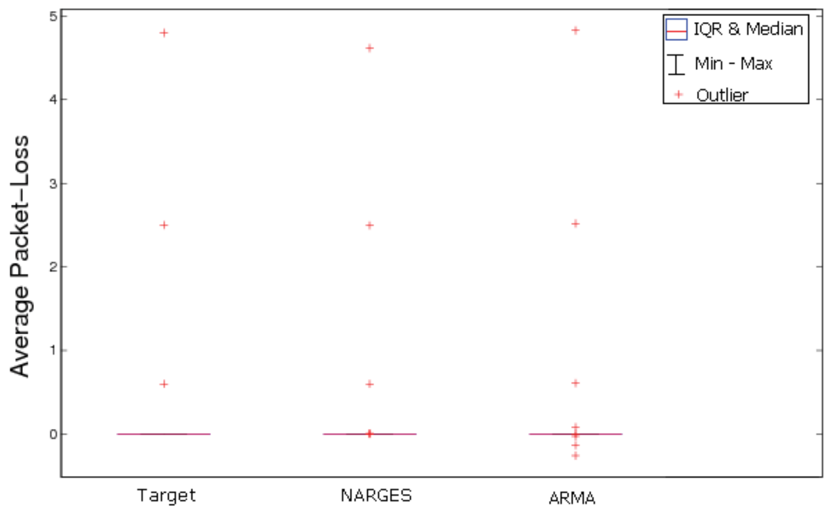


Figure 4.16: Boxplots of distributions for target packet-loss time-series within dataset 13 together with those for outputs of NARGES and ARMA

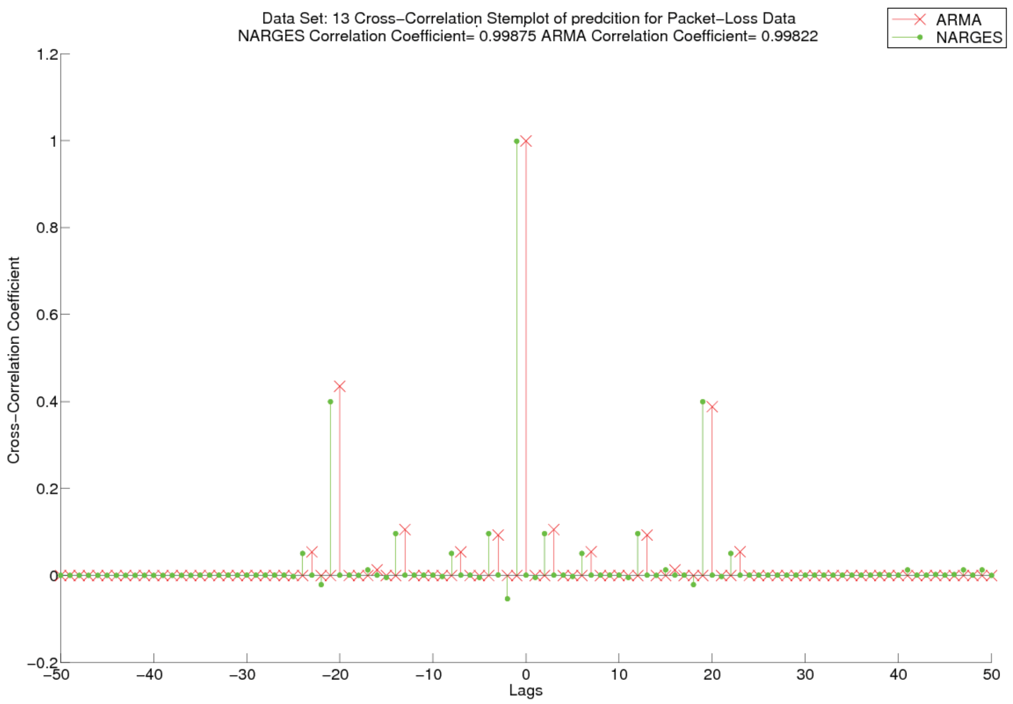


Figure 4.17: Stemplots of cross-correlation of NARGES predictions and target packet-loss time-series for dataset 13 together with those of ARMA

4.4.3 Experiment 3: Impact Analysis of Network Queuing Policies on Model Prediction

For these experiments, the data is generated using OPNET on a network both with and without QoS enabled, and with various background loads. The OPNET application parameters were set to generate a Low Load FTP flow, Low Resolution video and constant calls with 100 second durations. In the experiments, the queuing policy used for packet switching between the routers was FIFO, PQ or WFQ. There were a minimum VoIP packet-loss from zero to 300 in various experiments. The generic network topology in these experiments was the one described in Section 4.2.2.2.

4.4.3.1 Model Results with OPNET Datasets

Similar to experiment 1, the accuracies of the NARGES model and its HDAX subcomponent were compared to ARMA using the error function NRMSE. Ten simulation runs were to obtain the results. Maximum CCFs between the outputs of HDAX, NARGES or ARMA, and the target time-series are presented in Tables 4.10 and 4.11.

The queueing policies eight to ten, in Table 4.10 are described in Table 4.3. The numbers printed in **bold** distinguish algorithms which are more accurate, faster or have more correlated output.

The predicted average packet-loss for NARGES was also compared to that of ARMA. As Table 4.11 shows, the NARGES model predicts more precisely than ARMA but is slower to do so. This is because NARGES has more modules and processes more data than ARMA to predict the final packet-loss values.

Table 4.10: Normalised root mean square error (NRMSE) together with algorithms speed and cross-correlation coefficients of HDAX and ARMA forecasts for OPNET delay and jitter time-series.

		Forecasted Delay			**Forecasted Jitter**		
$Model$ $Name$	$Dataset$ $Category$	$NRMSE$ (%)	$Speed$ (sec)	CCF max	$NRMSE$ (%)	$Speed$ (sec)	CCF max
$HDAX$	8	3.27	0.031	0.975	**3.61**	0.030	**0.981**
$ARMA$	8	**1.95**	**0.018**	**0.990**	3.39	**0.023**	0.900
$HDAX$	9	2.50	0.205	0.980	3.73	0.208	**0.990**
$ARMA$	9	**2.47**	**0.101**	**0.988**	**3.10**	**0.106**	0.907
$HDAX$	10	3.49	0.030	0.968	3.57	0.031	**0.995**
$ARMA$	10	**2.60**	**0.021**	**0.998**	**2.68**	**0.023**	0.858

Table 4.11: Normalised root mean square error (NRMSE) together with algorithms speed and cross-correlation coefficients of NARGES and ARMA forecasts for OPNET packet-loss time-series.

$Model$ $Name$	$Dataset$ $Category$	$NRMSE$ (%)	$ModelSpeed$ (sec)	CCF max
$NARGES$	8	**0.01**	2.091	0.999
$ARMA$	8	3.31	**0.013**	0.999
$NARGES$	9	**0.04**	2.974	0.999
$ARMA$	9	3.47	**0.057**	0.999
$NARGES$	10	**1.23**	7.124	**0.999**
$ARMA$	10	5.63	**0.012**	0.998

4.4.3.2 Model Comparison

As in the previous experiments over D-ITG datasets, the accuracy, speed and the correlation of the model outputs are compared with the corresponding target data and the similarity, null hypothesis, is tested by Friedman tests. Table 4.12 presents the algorithm average ranking for each of the models. The **bold** printed rank numbers distinguish the algorithms with better rank.

As already mentioned, Friedman's test computes the ranking of the measured outputs for an algorithm with other k algorithms, assigning the best of them the rank 1 and the worst the rank k. Table 4.13 suggests that the null hypothesis is rejected when the $p-value$ for the corresponding component is less than the significance level ($\alpha = 0.05$).

Table 4.12: Average Rankings of the algorithms; Note that in testing the algorithms for accuracy, speed and cross-correlation (CCF) over Delay, Jitter and Packet-Loss

$Time-series$	$Test$	$ARMA$	$HDAX$
$Delay$	$Accuracy$	0.999	**1.999**
	$Speed$	**1.999**	0.999
	CCF	1.099	**1.899**
$Jitter$	$Accuracy$	1.499	**1.500**
	$Speed$	**1.999**	0.999
	CCF	**1.999**	0.999
	$Predictions$	$ARMA$	$NARGES$
$Packet-loss$	$Accuracy$	0.999	**1.999**
	$Speed$	**1.999**	0.999
	CCF	**1.999**	0.999

Table 4.13: Holm / Hochberg Table for $\alpha = 0.05$. Note that in testing the algorithms for accuracy, speed and cross-correlation (CCF) over Delay, Jitter and Packet-Loss models printed in **bold** are statistically significantly better.

$Data$	$Test$	$Algorithm$	$z = (R_0 - R_i)/SE$	p
$Delay$	$Accuracy$	**HDAX**	3.163	0.002
$Delay$	$Speed$	**ARMA**	3.162	0.002
$Delay$	CCF	**HDAX**	2.530	0.011
$Jitter$	$Accuracy$	HDAX	1.40×10^{-15}	0.9999
$Jitter$	$Speed$	**HDAX**	3.1623	0.002
$Jitter$	CCF	**ARMA**	3.162	0.002
$Packet-Loss$	$Accuracy$	**NARGES**	3.162	0.002
$Packet-Loss$	$Speed$	**ARMA**	3.162	0.002
$Packet-Loss$	CCF	**NARGES**	3.162	0.002

As shown in Table 4.12 the Holm's procedure shows that HDAX ranking is better than ARMA for the accuracy of the results while for the speed of the algorithms ARMA has higher rank and is better than HDAX. For cross-correlation between the forecasted and target time-series, HDAX has a higher rank for delay, but not for the jitter, values.

Moreover, according to the $p-value$s, HDAX results are significantly more accurate and correlated than ARMA for delay traces while ARMA is faster. For jitter values, HDAX is faster. For jitter values, the HDAX results are as accurate as ARMA while HDAX forecasts are less correlated compared to ARMA.

For the NARGES model, the output accuracy and quality of the predictions for packet-loss are statistically significantly better than ARMA, as shown in Table 4.13. However, as was the case with the D-ITG, NARGES model is slower than ARMA.

4.4.3.3 Discussion on the Quality of Results

As a representative, this section describes the distributions of the delay, jitter and packet-loss for dataset 41, generated by OPNET. In this experiment, 1,000,000 values were generated under a FIFO queuing discipline. Figures 4.18 and 4.19 show the boxplots of the distributions of the target delay and jitter time-series together with those generated by HDAX and ARMA.

Figure 4.22 shows boxplots of packet-loss time-series for the Target data and, NARGES and ARMA predictions. Figures 4.20, 4.21, 4.23 show the stemplots for the forecasted or predicted values generated by HDAX, NARGES or ARMA models.

4.4.3.4 HDAX

As shown in Figure 4.18, the target delay values for dataset 41 are spread between 800 and 950 ms with IQR of 40 ms, minimum of 650 ms, maximum of 950 ms and median of 870 ms. Considering the HDAX and ARMA boxplots, it can be seen that HDAX forecasts seems to be more correlated to the target delay values. HDAX also shows less outliers compared to ARMA which has a narrow distribution, spread all over the interval between 830 and 930 ms, with an IQR of 30 ms, a minimum of 650 and maximum of 1070 ms with extra outliers over the maximum.

Figure 4.19 shows that for jitter values, HDAX and the target data have a median of 5 ms. While HDAX output seems to have a similar distribution to the target, ARMA shows wider spread of values and a bigger IQR compared to the target distribution. HDAX

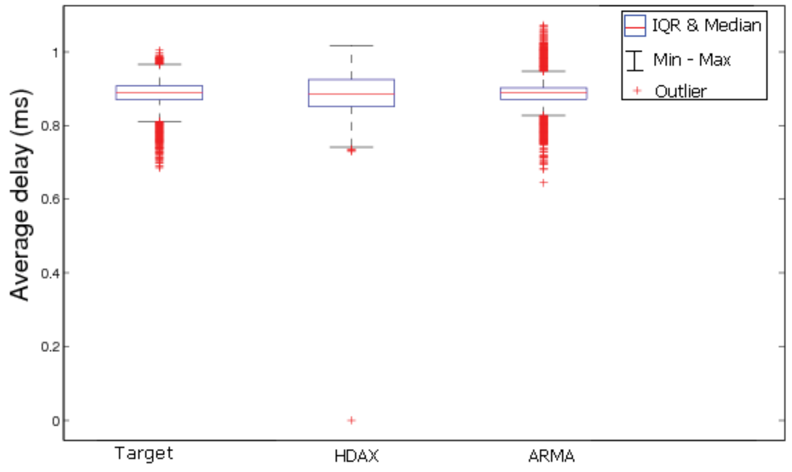


Figure 4.18: Boxplots of distributions of target delay time-series for dataset 41 together with those for outputs of HDAX and ARMA

shows a closer distribution to the target and more accuracy in estimation of the outliers. In Figure 4.19 ARMA shows extra outliers under the minimum.

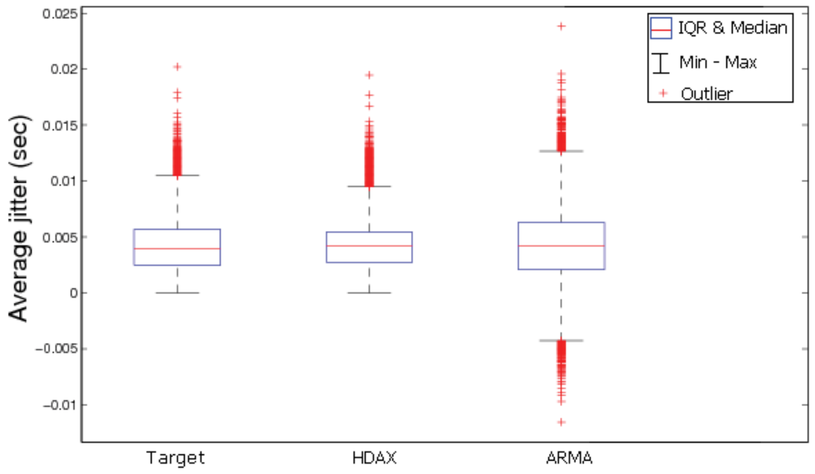


Figure 4.19: Boxplots of distributions of target jitter time-series for dataset 41 together with those for outputs of HDAX and ARMA

Compared to the target values of delay, the cross-correlation coefficients of the forecasted values are shown in Figure 4.20 – the green dash-dot and red dash-cross lines of ARMA and HDAX.

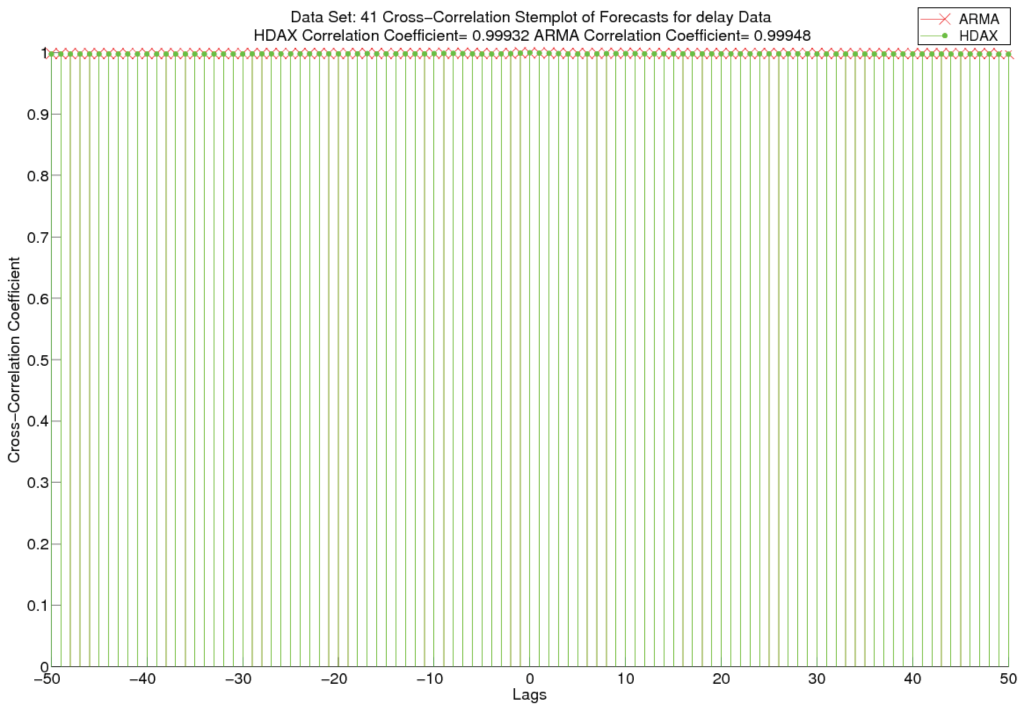


Figure 4.20: Stemplots of cross-correlation of HDAX forecasts and target delay time-series for dataset 41 together with those of ARMA

As shown in Figure 4.21, ARMA cross-correlation coefficients, depicted as the dash-cross red line, are showing less similarity to the target jitter data compared to HDAX. In this regard, the dash-dor green line of HDAX shows a higher correlation with the target data with a maximum cross-correlation of 0.999.

In summary, the forecasts of HDAX are more similar to the target values for both delay and jitter time-series in dataset 41. HDAX shows the same accuracy in forecasting compared to ARMA, although it is more accurate in forecasting outliers for both delay and jitter time-series.

4.4.3.5 NARGES

Figure 4.22 shows boxplots for target values and predictions of NARGES and ARMA models for dataset 41. These are similarly distributed all over the interval between 50

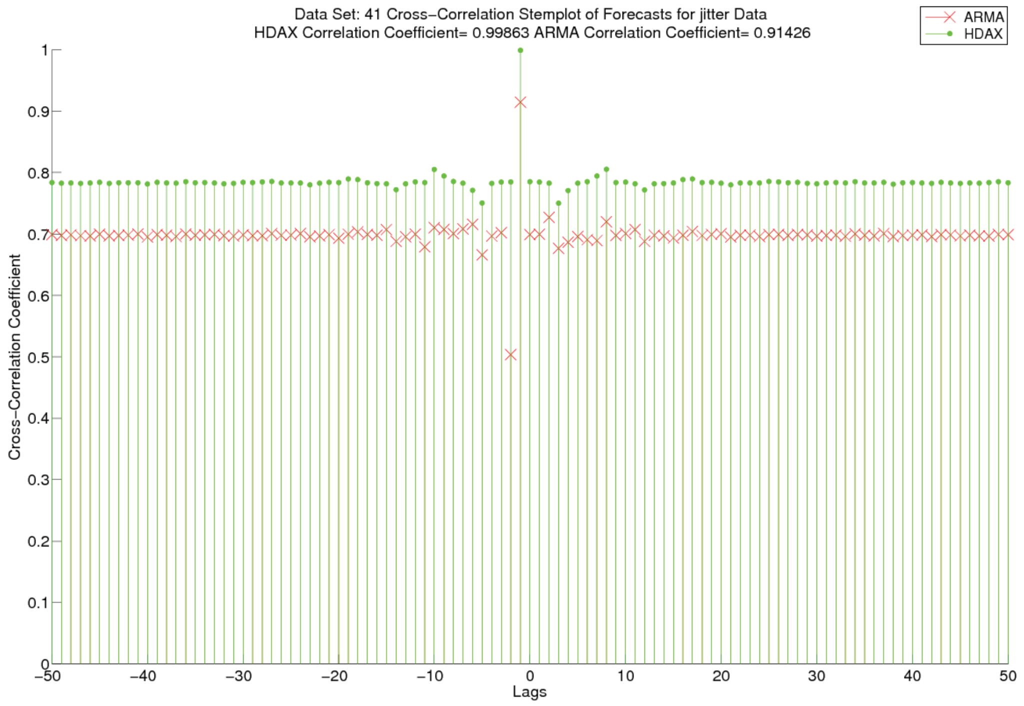


Figure 4.21: Stemplots of cross-correlation of HDAX forecasts and target jitter time-series for dataset 41 together with those of ARMA

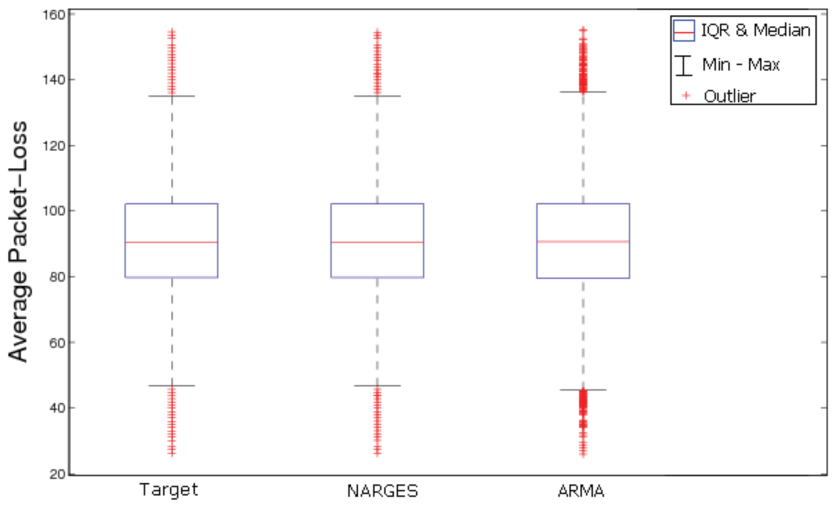


Figure 4.22: Boxplots of distributions of target packet-loss time-series for dataset 41 together with those for outputs of NARGES and ARMA

and 135 with median of 90 and an IQR of 20 packet-loss.

The stemplots in Figure 4.23 represents the cross-correlation coefficient plots for the pre-

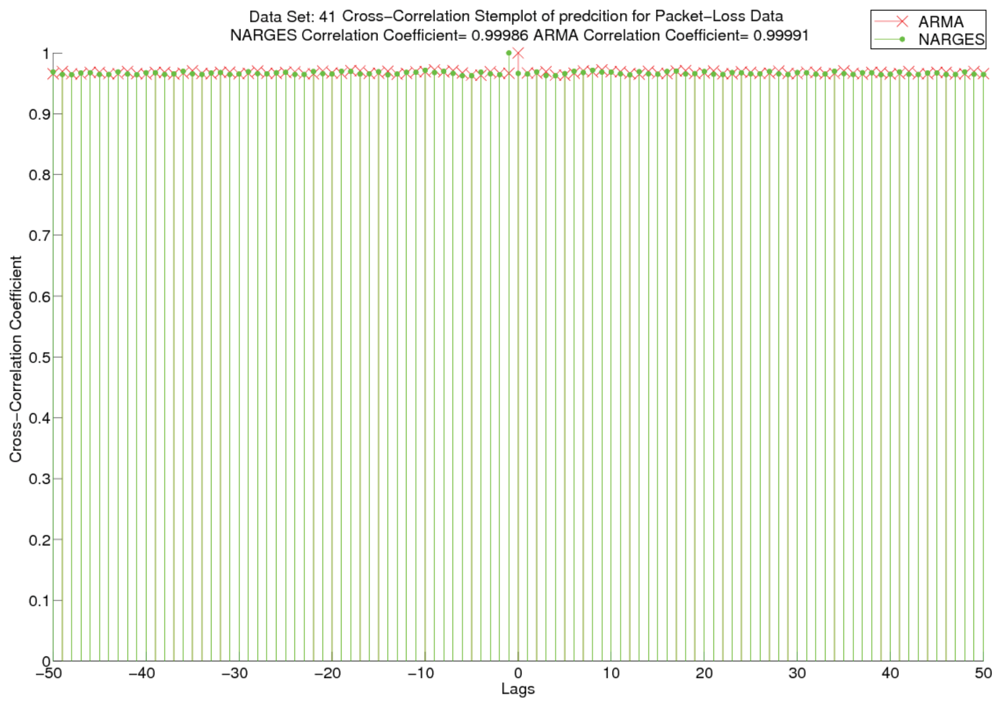


Figure 4.23: Stemplots of cross-correlation of NARGES predictions and target packet-loss time-series for dataset 41 together with those of ARMA

dicted values of NARGES and ARMA, compared to the target packet-loss time-series. It can be seen in Figure 4.23 that NARGES and ARMA predictions have the same maximum CCF.

4.5 Summary of Model Performance

Figure 4.24 summarises the results of HDAX for delay and jitter traces (first and second rows) as well as the NARGES results over packet-loss traces (third row) in a top down order.

The average *accuracy* of the forecasted delay and jitter time-series of HDAX and ARMA as well as the NARGES *accuracy* for the predicted packet-loss are shown in column (a) in Figure 4.24. Column (b) shows a comparison between the *speed* of the models with ARMA. Column (c) in Figure 4.24 shows the respective correlation coefficient between the output of the models and the target data.In terms of similarity measurement between the target time-series and the output by HDAX, NARGES and ARMA, the maximum similarity of normalised cross-correlations was used for correlation analysis between the time-series.

Regarding the *dataset categories* description in Tables 4.1 and 4.3 for D-ITG and OPNET datasets, each of twin bar charts are labeled with a number assigned to its corresponding *Dataset Category*. The numbers 1-7 refer to the results over D-ITG datasets and numbers 8-10 represents the results over OPNET datasets.

The predicted average packet-loss for NARGES was compared to ARMA. As the Figure 4.24 shows, the NARGES model generally predicts more precisely than ARMA with the OPNET datasets (labeled 8-10) but is slower. This is because NARGES has more modules and processes more data than ARMA to predict the final packet-loss values. The training time of the MLP module accounts for the longer time taken to run the model.

Currently network routers must send information between routers to inform about the peer status. The results in this thesis demonstrate, in a simulated setting at least, that a data mining agent can predict the peer status to reduce or perhaps eliminate the unnecessary network data transmission overhead and the time required for sending and receiving data network packets.

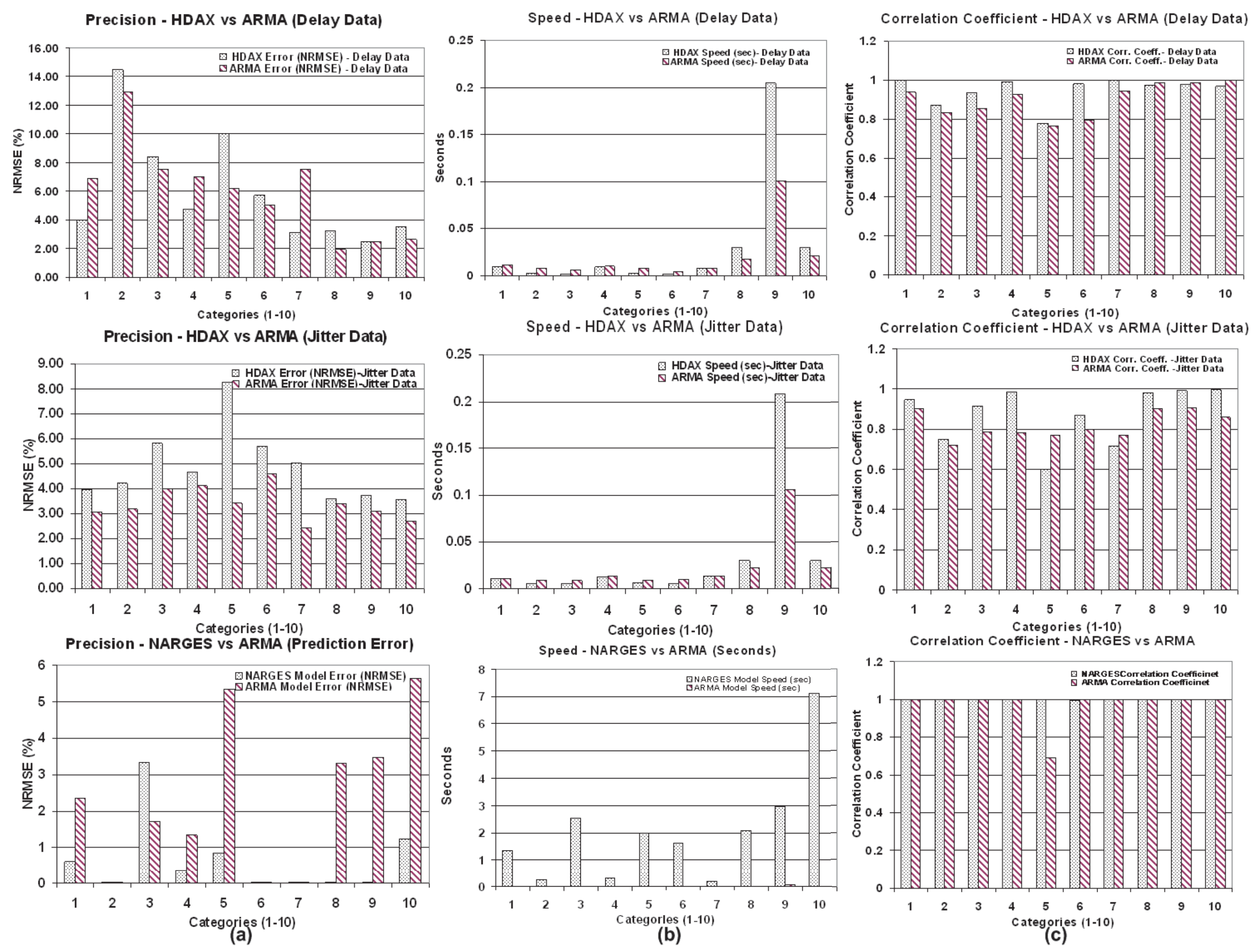


Figure 4.24: Error (NRMSE) of HDAX and NARGES vs ARMA together with speed of algorithm and cross-correlation coefficients are shown in the column (a) to (c), respectively. The first and second rows are the HDAX results and the last row shows the whole model (NARGES) results. In the twin bar charts, the left gray bars shows HDAX (in the first two rows) and NARGES (in the last row) while the right bar filled with wide downward diagonal pattern denotes ARMA outcomes.

4.6 Conclusions

The chapter presented the results evaluating the proposed models. The NARGES model and its subcomponent HDAX were validated with heterogeneous QoS traces. The results show that the quality and the accuracy of the proposed models are significantly better than ARMA. However, NARGES was slower than ARMA because it has to process more inputs. As can be seen from the competing speed of HDAX module, it is the training time of the MLP module that degrades the speed of the model. In Table 4.9, the significant difference between the $p - value$ of the Holm's procedure for the D–ITG and the longer OPNET datasets suggests that the model can work faster in a real–time network experiment, as the training time of MLP module is negligible in an infinite real-time run.

There are small but not yet ignorable errors caused by HDAX outputs. This is only a case for the experiments using D-ITG datasets and the inputs that had zero mean, i.e. stationary zero-mean time-series. HDAX showed a small "Standard Deviation" greater than zero, which led to a wider distribution in the results. However, HDAX can forecast outliers even in such a situation, and moreover, the above error can be eliminated by optimising the implementation. This is resolved for longer experiments, i.e. the experiments with OPNET generated data, and also for the experiments in which the algorithms dealing with stochastic and non-stationary data.

The evaluation of the proposed models in this chapter fulfills the contributions 3 and 4 in Chapter 1. Next, in Chapter 5, conclusions of the thesis and suggested future research directions will be discussed.

Chapter 5

Conclusions and Future Work

"We cannot solve our problems with the same thinking we used when we created them,"
— Albert Einstein

There is a reported dependency between packet-loss and the delay and jitter time-series assigned to a telecommunication link (Markopoulou et al., 2006; Jiang and Schulzrinne, 2000). Multimedia applications such as Voice over IP are sensitive to loss and packet recovery is not a merely efficient solution with the increasing number of Internet users. Thus, predicting packet-loss from network dynamics of past transmissions is crucial to inform the next generation of routers in making smart decisions. Regarding this, the current research aimed at modeling packet-loss traces based on other QoS parameters, namely delay and jitter time-series.

In this thesis, a hybrid data mining model, NARGES, is proposed. It is designed and implemented for predicting packet-loss based on the forecasted delays and jitters. NARGES consists of two parts: a historical symbolic time-series approximation module, called HDAX to approximate delay and jitter values, and a Multilayer Perceptron (MLP) to predict future packet-loss.

The outcomes are validated with real-time and simulated heterogeneous quality of service traces, namely delay, jitter and packet-loss time-series from UNINA and OPNET simulations. The results showed improved precision and correlation of the imputed values of the proposed models compared to autoregressive moving average, ARMA, model. Thus, the proposed models may benefit decision-making for routing management in a communications network.

This final chapter summarises the specific contributions of the thesis and discusses open research directions about this work.

5.1 Conclusions

This thesis makes four contributions to the state of the art in telecommunications time-series data mining. This includes:

1. In Section 3.3, the thesis proposed "*a real time Data Mining (DM) model called HDAX.*"

   An historical symbolic time-series approximation model, HDAX, was introduced. The HDAX approximates delay and jitter in real–time streams based on the observed delay and jitter trends and values in two previous consecutive values in corresponding trend sequences.

   The model uses most frequent patterns in forecasting current trends and values of delay and jitter time-series. That is, based on the observed frequency of trend patterns in a time-series, the model takes the most frequent trend patterns to approximate the maximum likelihood of the next trend, and to calculate the next value, of the time-series.

2. In Section 3.3, the thesis also proposed "*a hybrid DM model called NARGES*," an informed decision-making model for routing management in a communications network.

   The model uses forecasted delay and jitter traces obtained from HDAX module to predict packets lost within a network node (usually an edge router). The motivation is to use this predicted value to indicate the current degree and severity of congestion and likelihood of packet-loss, and to use it as a vital component in sender–based error and rate control mechanisms for multimedia.

3. In Section 4.4.2, the "*evaluation of the proposed DM models with D-ITG*" data has been described.

   In this regard, the proposed models in this thesis were validated with the Distributed Internet Traffic Generator (D-ITG) data described in Section 4.2.1. As discussed before, three sets of time-series are considered for each experiment: the original data generated by D-ITG, the output of HDAX (or NARGES) and the output of ARMA.

4. In Section 4.4.3, the "*evaluation of the proposed DM models with OPNET data*" has been described.

   In this regard, the proposed models were validated with the OPNET Modeler data described in Section 4.2.2. As discussed before, three sets of time-series are considered for each experiment: the original data generated by OPNET, the output of HDAX (or NARGES) and the output of ARMA.

The datasets were categorised in two ways: (i) according to the end-to-end path for the datasets generated by D-ITG; and (ii) according to the queuing policy used for the datasets generated by OPNET.

Forty-six datasets were used for computing results. Each dataset includes time-stamped traces of delay, jitter and loss values. There were thirty-six datasets generated by D-ITG with an average 3000 values in each of delay, jitter and packet-loss time-series. There were also ten datasets generated by OPNET, nine with 48000 values and one with 1,000,000 values. The data generated by D-ITG are gathered in two ways: (i) the archives obtained from UNINA; (ii) the data probed over UTS network test-bed.

The datasets generated by D-ITG traffic generator were described in Section 4.2.1 and used in running experiments in Section 4.4.1 and Section 4.4.2. A selection of seven categories of datasets, based on the End-to-End Path (e2eP) categories explained in Section 4.2.1.1, were used. The performance of the data mining prediction model was evaluated using these datasets and the model validation with the heterogeneous QoS traces showed the quality of the models and the significant improvement in precision of the proposed model compared to ARMA model.

OPNET datasets, on the other hand, were used to study the effect of different packet transmission policies, longer experiments and network congestion states on the performance of the implemented data mining model. A series of queuing disciplines in the routers (FIFO, WFQ and PQ) that can affect VoIP applications and link congestion was designed and simulated on OPNET Modeler to generate the OPNET datasets. These datasets were used in running experiments in Section 4.4.3.

The performance of the data mining prediction model was evaluated using these datasets and the model validation with offline heterogeneous QoS traces demonstrated the quality and improvement of the proposed model compared to ARMA model.

The results in the thesis illustrate that the proposed DM model, NARGES, may predict the peer status, namely the packet-loss numbers assigned to a peer link, for all End-to-End Path (e2eP) and all queuing scenarios considered in the experiments within Section 4.4.

In this regard, the DM predicted knowledge by NARGES model was statistically shown to have significant correlation and precision compared to the measured values on a link. Consequently, the NARGES DM model can reduce unnecessary network data transmission overhead and may eliminate the time required for sending and receiving data network packets.

5.2 Limitations

The primary limitations in this research were in the evaluation step. Evaluations were limited to using off-line simulated data obtained from the OPNET simulations and D-ITG traffic generator. Although there is a built-in co-simulation module within OPNET simulator, the poor synchronisation between MATLAB and OPNET made a co-simulation too slow and almost impractical for an online evaluation. Moreover, the BGP protocol should be updated to accept new metrics for routing decision-making, which were out of scope of this study. As mentioned before, telecommunications data mining is also restricted to sharing data, information and knowledge. That is, utilising data from other resources from another autonomous system is limited.

5.3 Future Work

The model outcomes in this thesis inspired several ideas for novel ways to enhance time-series data mining and forecasting. A few of the ideas and challenges which are useful for future study in the area of Data Mining (DM) and Telecommunications are

1. Making a theoretical analysis of the time complexity of the proposed models.

2. Refining HDAX model and algorithm to improve the proposed model and its implementation for forecasting with missing values.

   Tresp and Hofmann (1998) introduce formal way of modeling a nonlinear time-series with missing values. Assuming that in the Eq. (3.2) y_{t-2} is missing, the typical agenda is to approximate $\hat{y}_{t-2}$ of the missing value and then substitute it in HDAX. Eq. (3.2) will then be changed to

$$\hat{y_t} = f(y_{t-1}, \hat{y_{t-2}}, \ldots, y_{t-N}) + \epsilon_t \tag{5.1}$$

3. Refining HDAX metrics for using statistical techniques such as estimation theory (Mateo et al., 2013).

   This can improve the accuracy and efficiency of the algorithm. The metrics such as maximum delay threshold and window size should be determined dynamically to reflect the prevailing online status of the network. This will enable us to deploy HDAX more reliably.

4. Using other prediction models such as Hidden Markov Model, Bayesian Network and Linear Regression in the proposed conceptual framework for NARGES DM model presented in Section 3.1 and comparing the results to those made by using the Multilayer Perceptrons.

   The MLP predictive module in NARGES model was chosen based on the higher reputed prediction ability of an MLP. However, for a curious researcher there is a potential gap to compare the performance parameters of NARGES using other predictive models instead of the MLP.

   Moreover, data stream mining methods may be applicable to the stream of data in this domain.

5. Implementing an interactive co-simulation environment between MATLAB and

OPNET to obtain real-time simulation results. The NARGES model, implemented on OPNET BGP routers on different autonomous systems, provides knowledge for packet switching between routers.

For realtime co-simulations, MATLAB Executable external interface function (MEX-files) may be used to exchange measured and forecasted time-series to and from the OPNET routines on simulated BGP routers. Explanation about OPNET co-simulation with an external program may be find at in Fujita (2003).

The motivation for this task is to check the impact of the NARGES model on a router decision for packet switching to its peer in a real-time scenario and to study the network utilisation of other links with and without the presence of the DM model.

The challenges assigned with this task are: (i) synchronisation between MATLAB and OPNET, and (ii) manipulation of an inter-network protocol such as BGP is necessary to be able to utilise the NARGES model knowledge in router decision making and updating the routing table.

The results in this thesis suggest that Data Mining has a potential to precisely predict a link packet-loss based upon the approximation of current packet delay and jitter. The knowledge provided by DM module, which is ideally installed on a future smart router, can assist in detection of partial network failure, e.g. a link or peer failure, and is beneficial to routing management decision-making. Future challenges in non-numerical modeling of stochastic time-series remain open for researchers to model online time-series with unknown parameters and missing values.

Appendix A

List of Acronyms

ACF	Autocorrelation Function
ADSL	Asymmetric bit rate Digital Subscriber Line
ANN	Artificial Neural Network
ANOVA	Analysis Of Variance
AR	Autoregressive Model
ARCH	Autoregressive Conditional Heteroskedasticity Model
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BE	Best Effort
BG	IEEE 802.11b/g
BN	Bayesian Network
BGP	Border Gateway Protocol

CCF	Cross–Correlation Function
CPU	Central Processing Unit
CW	Computing with Words
DB	Database
D-ITG	Distributed Internet Traffic Generator
DM	Data Mining
DMF	Deterministic Mapping Function
DNS	Domain Name System
DS1	DS1 is a standard in telecommunications to transmit voice and data
DT	Decision Trees
e2eP	End-to-End Path
ECG	Electrocardiography
EF	Expedited Forward
FARIMA	Autoregressive Fractionally Integrated Moving Average
FEC	Forwarded Error Correction
FEIT	Faculty of Engineering and Information Technology
FIFO	First In First Out
FP	Frequent Pattern
FTP	File Transfer Protocol

FTSDB	Forecasted Time-Series Database
GARCH	Generalised Autoregressive Conditional Heteroskedasticity Model
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HDAX	Historical Symbolic Delay Approximation Model
HMM	Hidden Markov Model
HPC	High Performance Computing
ICMP	Internet Control Message Protocol
IDS	Initial Dataset
IDT	Inter Departure Time
INTERMON	European IST Inter-Domain QoS Monitoring project
IP	Internet Protocol
IPDV	IP Packet Delay Variation
IQR	Interquartile Range
IRP	Interior Router Protocol
IS-IS	Intermediate System To Intermediate System
ITU	International Telecommunications Union
KDD	Knowledge Discovery in Databases
LAN	Local Area Networks

LR	Linear Regression
M2E	Mouth-to-Ear
MA	Moving Average Model
MagNets	A next-generation wireless access network deployed in the city of Berlin.
MAP	Moving Approximation
MATLAB	The language of technical computing
MEX	MATLAB Executable external interface function
MHA	Minimum Hop-Count Algorithm
MLE	Maximum Likelihood Estimator
MLP	Multilayer Perceptrons
MPLS	Multi-Protocol Label Switching
MRB	Maximum Reservable Bandwidth
MS	Microsoft
NARGES	Data Mining Model for Smart Routing in Communications Networks
NRMSE	Normalised Root Mean Square Error
NS2	NS2
OMNET	OMNET++
OO	Object-oriented
OPNET	OPNET Modeller

OSPF	Open Shortest Path First
OWD	One Way Delay
PC	Personal Computer
PD	Pattern Definition
PDB	Pattern Database
PDL	Pattern Definition Language
PPP	Point-to-Point Protocol
PQ	Priority Queueing
PS	Payload Size
QoS	Quality of Service
RSS	Residual Sum of Squares
RTP	Routing Update Protocol
RTT	Round Trip Time
SAX	Symbolic Aggregate Approximation
SCTP	Stream Control Transmission Protocol
SD	Standard Deviation
SLA	Service Level Agreement
SWP	Shortest-Widest Path Routing Strategy
TCP	Transmission Control Protocol

TDS	Terminal Data Set
TE	Traffic Engineering
Telnet	Telnet Protocol
TOS	Type Of Service
TS	Time-Series
TSDB	Database of QoS Measured Time Series
TSDM	Time-series Data Mining
TSP	Traffic Specification Protocol
TTL	Time to Live
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications Systems
UNINA	University of Naples Federico II
UTS	University of Technology, Sydney
VoIP	Voice over IP
WFQ	Weighted Fair Queuing
WLAN	Wireless LAN
WS	Workstation
WSP	Widest-Shortest Path Routing Strategy

Appendix B

ARMA Parameter Estimation

The appendix reviews four techniques for ARMA model parameter estimation, namely YuleWalker Estimation, Burg's Algorithm, Hannan-Rissanen Algorithm and Maximum Likelihood Estimation.

B.1 Preliminary Estimation

There are four techniques considered in Brockwell and Davis (2006) for preliminary estimation of the θ and ϕ coefficients of casual ARMA(p,q) process 2.7 and σ^2 of the process $X_t = x_1, x_2, \ldots, x_n$.

The Yule-Walker and Burg methods are based on fitting the pure AR models. The Yule-Walker method may be applied to models with $q > 0$. In this regard, Hannan-Rissanen algorithms give preliminary approximates of the ARMA parameters when $q > 0$.

- **YuleWalker Estimation**

  According to (Brockwell and Davis, 2006, pp. 139-146), in a pure AR model, where the θ coefficient is correspondingly 1, the process X_t may be written in the form

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j}$$

The Yule-Walker equations will be as

$$\Gamma_p \phi = \gamma_p$$

and

$$\sigma^2 = \gamma(0) - \phi' \gamma_p$$

where Γ_p is the covariance matrix determined from σ^2 and ϕ coefficient, accordingly.

- **Burg's Algorithm**

  The Yule-Walker coefficients defines the most precise coefficients of linear predictor of X_{p+1} using X_1,..., X_p. This assumes that the ACF of X_t matches the sample ACF at lags[1] 1, ..., p.

  Given the Yule-Walker coefficients $\hat{\phi_{p1}}$, ..., $\hat{\phi_{pp}}$ at lags 1,...,p, Burgs algorithm approximates the partial autocorrelation function ϕ_{11},ϕ_{22},... by minimizing sums of square error assigned to each of the coefficients ϕ_{ii} (Brockwell and Davis, 2006).

  Let x_1, ..., x_n be the probes of a stationary zero-mean time-series X_t. Let also define the respective Burgs algorithm backward and forward errors as: (i) $v_i(t)$, t = $i + 1$, ..., n, $0 \leq i < n$, as the difference of x_{n+1-t} and x_{n+1-t} best linear

[1]An ACF value obtained at time $t + k$ ($k > 0$) is called to lag behind another value obtained at time t and the extent (length) of the lag is k. In this regard, a value obtained k time units before may be regarded as having a negative lag.

estimate, in terms of the subsequent i probes and (ii) $u_i(t)$, t = $i + 1$, ..., n, as the difference of $x_{n+1+i-t}$ and the best linear estimate of $x_{n+1+i-t}$ in terms of the preceding i probes. In this regard, the following recursive equations are satisfied by the backward and forward prediction errors $v_i(t)$ and $u_i(t)$:

$$v_i(t) = v_{i-1}(t) - \phi_{ii} u_{i-1}(t-1) \tag{B.2}$$

and

$$\begin{aligned} u_0(t) &= v_0(t) = x_{n+1-t} \\ u_i(t) &= u_{i-1}(t-1) - \phi_{ii} v_{i-1}(t) \end{aligned} \tag{B.3}$$

On this basis and with respect to ϕ_{pp}, Burgs estimation for $\phi_{pp}^{(B)}$ of ϕ_{pp} $(1 \leq p \leq n-1)$ can be computed by minimizing the Eq. (B.4)

$$\sigma_p^2 := \frac{1}{2(n-p)} \sum_{t=p+1}^{n} [u_p^2(t) + v_p^2(t)] \tag{B.4}$$

- **Hannan-Rissanen Algorithm**

  The defining equations for a causal AR(p) model have the form of a linear regression model with coefficient vector ϕ = (ϕ_1, ..., ϕ_p)'.

  This suggests the use of simple least squares regression for obtaining preliminary parameter estimates when $q = 0$. Application of this technique when $q > 0$ is complicated by the fact that in the general ARMA(p,q) equations X_t is regressed not only on X_{t-1}, ..., X_{t-p}, but also on the unobserved quantities Z_{t-1}, ..., Z_{t-q}. Nevertheless, it is still possible to apply least squares regression to the estimation of ϕ and θ by first replacing the unobserved quantities Z_{t-1}, ..., Z_{t-q} in 2.7 by estimated values Z_{t-1}, …,Z_{t-q} . The above ϕ and θ parameters are then set by regressing X_t onto $X_{t-1}, \ldots, X_{t-p}$,$Z_{t-1}, \ldots$,Z_{t-q}.

These are two steps in the procedure of Hannan-Rissanen estimation

Step 1. An AR(m) model ($m >$ max(p,q)) is fitted using the Yule-Walker estimates of Section 2.5.1.4. If ($\hat{\phi_{m1}}$, …, $\hat{\phi_{mm}}$)′ is the vector of coefficients estimate, then the estimate of the residuals are formulated as

$$\hat{Z}_t = X_t - \hat{\phi_{m1}} X_{t-1} - \ldots - \hat{\phi_{mm}} X_{t-m}, t = m+1, \ldots, n.$$

Step 2. Once the estimate of the residuals Z_t, $t = m + 1, \ldots, n$, have been calculated as in Step 1, the parameter vector, β = (ϕ',θ')′ is approximated using least squares linear regression of X_t onto (X_{t-1}, …, X_{t-p},Z_{t-1}, …, Z_{t-q}), t = m + 1 + q, …, n, i.e., by minimizing the sum of squares

$$S(\beta) = \Sigma_{t=m+1+q}^{n} (X_t - \phi_1 X_{t-1} - \ldots - \phi_p X_{t-p} - \theta_1 \hat{Z_{t-1}} - \ldots - \theta_1 \hat{Z_{t-1}})^2$$

with respect to β.

The following equation calculates the Hannan-Rissanen approximation

$$\hat{\beta} = (Z'Z)^{-1} Z' X_n,$$

where $X_n = (X_{m+1+q}, ..., X_n)'$ and Z is the $(n-m-q) \times (p+q)$ matrix (Brockwell and Davis, 2006).

B.2 Maximum Likelihood Estimation

Maximum likelihood estimation is based upon the mathematical expression of a probability function of the sample data. It is an analytical maximisation procedure that applies to any kind of missing data. The expression includes the unknown parameters the model. In this regard, the parameter values that maximise the sample probability are called Maximum Likelihood Estimator (MLE).

Regarding X_t to be a zero-mean Gaussian time-series with ACF $\gamma(i, j) = E(X_i X_j)$, let $X_n = (X_1, \ldots, X_n)'$ and $\hat{X}_n = (\hat{X}_1, \ldots, \hat{X}_n)'$, where $\hat{X}_1 = 0$ and $\hat{X}_j$ = $E(X_j | X_1, \ldots,$ $X_{j-1}) = P_{j-1} X_j$, $j \geq 2$. Let also Γ_n denotes the covariance matrix Γ_n = $E(X_n X_n')$, and is non-singular. The probability of X_n is as

$$L(\Gamma_n) = (2\pi)^{-n/2} (det \Gamma_n)^{-1/2} exp(-\frac{1}{2} X_n' \Gamma_n^{-1} X_n) \tag{B.5}$$

As shown in (Brockwell and Davis, 2006) and (Chatfield, 2004), the direct computation of det Γ_n and Γ_n^{-1} may be expressed by means of the one-step prediction errors $X_j - \hat{X}_j$ and their variances v_{j-1}, j = 1, . . ., n.

Appendix C

Implementation of Loss Predictor - Source Code

The appendix illustrates an implementation of Loss Predictor - Source Code - used in Chapter 4

C.1 NARGES Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright and terms of use (DO NOT REMOVE):
% The code is made freely available for non-commercial
% uses only, provided that the copyright
% header in each file not be removed, and suitable
% citation(s) (see below) be made for papers
% published based on the code.
%
% Copyright (c) 2009-2012, Hooman Homayounfard,
%                          hoomanhm@it.uts.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [out,NARGESNet]= NARGES(Ddata,Jdata,Pdata,winsize,sd,sj,sp,pici)

```

```

%% Loading data if nargin.%%
%% ALTRENATIVELY, ONE OF THESE DATASET MAY HAVE BEEN USED FOR AN INDIVIDUAL EXPERIMENT
    WHEN NEEDED %%
if nargin == 0
%Ddata=load('/delay_adsl-to-wired-recv-const-1-udp-64.log.dat');
%Jdata=load('/jitter_adsl-to-wired-recv-const-1-udp-64.log.dat');
%Pdata=load('/packetloss_adsl-to-wired-recv-const-1-udp-64.log.dat');
%pici=1;
%Ddata=load('/delay_adsl-to-wired-recv-const-1-udp-512.log.dat');
%Jdata=load('/jitter_adsl-to-wired-recv-const-1-udp-512.log.dat');
%Pdata=load('/packetloss_adsl-to-wired-recv-const-1-udp-512.log.dat');
%pici=2;
%Ddata=load('/delay_gprs-to-wired-winlin-tcp-512.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winlin-tcp-512.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winlin-tcp-512.log.dat');
%pici=3;
%Ddata=load('/delay_gprs-to-wired-winlin-tcp-1024.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winlin-tcp-1024.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winlin-tcp-1024.log.dat');
%pici=4;
%Ddata=load('/delay_gprs-to-wired-winlin-udp-512.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winlin-udp-512.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winlin-udp-512.log.dat');
%pici=5;
%Ddata=load('/delay_gprs-to-wired-winlin-udp-1024.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winlin-udp-1024.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winlin-udp-1024.log.dat');
%pici=6;
%Ddata=load('/delay_gprs-to-wired-winwin-tcp-512.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winwin-tcp-512.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winwin-tcp-512.log.dat');
%pici=7;
%Ddata=load('/delay_gprs-to-wired-winwin-tcp-1024.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winwin-tcp-1024.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winwin-tcp-1024.log.dat');
%pici=8;
%Ddata=load('/delay_gprs-to-wired-winwin-tcp-512.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winwin-tcp-512.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winwin-tcp-512.log.dat');
%pici=9;
```

```
%Ddata=load('/delay_gprs-to-wired-winwin-tcp-1024.log.dat');
%Jdata=load('/jitter_gprs-to-wired-winwin-tcp-1024.log.dat');
%Pdata=load('/packetloss_gprs-to-wired-winwin-tcp-1024.log.dat');
%pici=10;
%Ddata=load('/delay_umts-to-wired-tcp-64.log.dat');
%Jdata=load('/jitter_umts-to-wired-tcp-64.log.dat');
%Pdata=load('/packetloss_umts-to-wired-tcp-64.log.dat');
%pici=11;
%Ddata=load('/delay_umts-to-wired-tcp-512.log.dat');
%Jdata=load('/jitter_umts-to-wired-tcp-512.log.dat');
%Pdata=load('/packetloss_umts-to-wired-tcp-512.log.dat');
%pici=12;
%Ddata=load('/delay_umts-to-wired-tcp-1024.log.dat');
%Jdata=load('/jitter_umts-to-wired-tcp-1024.log.dat');
%Pdata=load('/packetloss_umts-to-wired-tcp-1024.log.dat');
%pici=13;
%Ddata=load('/delay_umts-to-wired-udp-64.log.dat');
%Jdata=load('/jitter_umts-to-wired-udp-64.log.dat');
%Pdata=load('/packetloss_umts-to-wired-udp-64.log.dat');
%pici=14;
%Ddata=load('/delay_umts-to-wired-udp-512.log.dat');
%Jdata=load('/jitter_umts-to-wired-udp-512.log.dat');
%Pdata=load('/packetloss_umts-to-wired-udp-512.log.dat');
%pici=15;
%Ddata=load('/delay_umts-to-wired-udp-1024.log.dat');
%Jdata=load('/jitter_umts-to-wired-udp-1024.log.dat');
%Pdata=load('/packetloss_umts-to-wired-udp-1024.log.dat');
%pici=16;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-tcp-64.log.dat');
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-tcp-64.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-tcp-64.log.dat');
%pici=17;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-tcp-256.log.dat');
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-tcp-256.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-tcp-256.log.dat');
%pici=18;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-tcp-1024.log.dat');
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-tcp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-tcp-1024.log.dat');
%pici=19;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-udp-64.log.dat');
```

```
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-udp-64.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-udp-64.log.dat');
%pici=20;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-udp-512.log.dat');
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-udp-512.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-udp-512.log.dat');
%pici=21;
%Ddata=load('/delay_wired-to-adsl-recv_const-1-udp-1024.log.dat');
%Jdata=load('/jitter_wired-to-adsl-recv_const-1-udp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-adsl-recv_const-1-udp-1024.log.dat');
%pici=22;
%Ddata=load('/delay_wired-to-gprs-linwin-tcp-64.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-tcp-64.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-tcp-64.log.dat');
%pici=23;
%Ddata=load('/delay_wired-to-gprs-linwin-tcp-512.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-tcp-512.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-tcp-512.log.dat');
%pici=24;
%Ddata=load('/delay_wired-to-gprs-linwin-tcp-1024.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-tcp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-tcp-1024.log.dat');
%pici=25;
%Ddata=load('/delay_wired-to-gprs-linwin-udp-64.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-udp-64.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-udp-64.log.dat');
%pici=26;
%Ddata=load('/delay_wired-to-gprs-linwin-udp-512.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-udp-512.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-udp-512.log.dat');
%pici=27;
%Ddata=load('/delay_wired-to-gprs-linwin-udp-1024.log.dat');
%Jdata=load('/jitter_wired-to-gprs-linwin-udp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-linwin-udp-1024.log.dat');
%pici=28;
%Ddata=load('/delay_wired-to-gprs-winwin-tcp-64.log.dat');
%Jdata=load('/jitter_wired-to-gprs-winwin-tcp-64.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-winwin-tcp-64.log.dat');
%pici=29;
%Ddata=load('/delay_wired-to-gprs-winwin-tcp-1024.log.dat');
%Jdata=load('/jitter_wired-to-gprs-winwin-tcp-1024.log.dat');
```

```
%Pdata=load('/packetloss_wired-to-gprs-winwin-tcp-1024.log.dat');
%pici=30;
%Ddata=load('/delay_wired-to-gprs-winwin-udp-64.log.dat');
%Jdata=load('/jitter_wired-to-gprs-winwin-udp-64.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-winwin-udp-64.log.dat');
%pici=31;
%Ddata=load('/delay_wired-to-gprs-winwin-udp-512.log.dat');
%Jdata=load('/jitter_wired-to-gprs-winwin-udp-512.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-winwin-udp-512.log.dat');
%pici=32;
%Ddata=load('/delay_wired-to-gprs-winwin-udp-1024.log.dat');
%Jdata=load('/jitter_wired-to-gprs-winwin-udp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-gprs-winwin-udp-1024.log.dat');
%pici=33;
%Ddata=load('/delay_wired-to-umts-tcp-64.log.dat');
%Jdata=load('/jitter_wired-to-umts-tcp-64.log.dat');
%Pdata=load('/packetloss_wired-to-umts-tcp-64.log.dat');
%pici=34;
%Ddata=load('/jitter_wired-to-umts-tcp-512.log.dat');
%Jdata=load('/delay_wired-to-umts-tcp-512.log.dat');
%Pdata=load('/packetloss_wired-to-umts-tcp-512.log.dat');
%pici=35;
%Ddata=load('/delay_wired-to-umts-tcp-1024.log.dat');
%Jdata=load('/jitter_wired-to-umts-tcp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-umts-tcp-1024.log.dat');
%pici=36;
%Ddata=load('/delay_wired-to-wireless-sctp-64.log.dat');
%Jdata=load('/jitter_wired-to-wireless-sctp-64.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-sctp-64.log.dat');
%pici=37;
%Ddata=load('/delay_wired-to-wireless-sctp-512.log.dat');
%Jdata=load('/jitter_wired-to-wireless-sctp-512.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-sctp-512.log.dat');
%pici=38;
%Ddata=load('/delay_wired-to-wireless-sctp-1024.log.dat');
%Jdata=load('/jitter_wired-to-wireless-sctp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-sctp-1024.log.dat');
%pici=39;
%Ddata=load('/delay_wired-to-wireless-tcp-64.log.dat');
%Jdata=load('/jitter_wired-to-wireless-tcp-64.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-tcp-64.log.dat');
```

```
%pici=40;
%Ddata=load('/delay_wired-to-wireless-tcp-512.log.dat');
%Jdata=load('/jitter_wired-to-wireless-tcp-512.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-tcp-512.log.dat');
%pici=41;
%Ddata=load('/delay_wired-to-wireless-tcp-1024.log.dat');
%Jdata=load('/jitter_wired-to-wireless-tcp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-tcp-1024.log.dat');
%pici=42;
%Ddata=load('/delay_wired-to-wireless-udp-64.log.dat');
%Jdata=load('/jitter_wired-to-wireless-udp-64.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-udp-64.log.dat');
%pici=43;
%Ddata=load('/delay_wired-to-wireless-udp-512.log.dat');
%Jdata=load('/jitter_wired-to-wireless-udp-512.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-udp-512.log.dat');
%pici=44;
%Ddata=load('/delay_wired-to-wireless-udp-1024.log.dat');
%Jdata=load('/jitter_wired-to-wireless-udp-1024.log.dat');
%Pdata=load('/packetloss_wired-to-wireless-udp-1024.log.dat');
%pici=45;
%Ddata=load('/NARGES-DiffSrv-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-DiffSrv-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-DiffSrv-DES-1_IP.loss.dat');
%pici=46;
%Ddata=load('/NARGES-Diffsrv_Video-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-Diffsrv_Video-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-Diffsrv_Video-DES-1_IP.loss.dat');
%pici=47;
%Ddata=load('/NARGES-DiffSrv_VoIP-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-DiffSrv_VoIP-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-DiffSrv_VoIP-DES-1_IP.loss.dat');
%pici=48;
%Ddata=load('/NARGES-FIFO-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-FIFO-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-FIFO-DES-1_IP.loss.dat');
%pici=49;
%Ddata=load('/NARGES-FIFO_Long-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-FIFO_Long-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-FIFO_Long-DES-1_IP.loss.dat');
%pici=50;
```

```
%Ddata=load('/NARGES-FIFO_VIdeo-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-FIFO_VIdeo-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-FIFO_VIdeo-DES-1_IP.loss.dat');
%pici=51;
%Ddata=load('/NARGES-FIFO_VoIP-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-FIFO_VoIP-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-FIFO_VoIP-DES-1_IP.loss.dat');
%pici=52;
%Ddata=load('/NARGES-PQ-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-PQ-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-PQ-DES-1_IP.loss.dat');
%pici=53;
%Ddata=load('/NARGES-PQ_Video-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-PQ_Video-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-PQ_Video-DES-1_IP.loss.dat');
%pici=54;
%Ddata=load('/NARGES-PQ_VoIP-DES-1_Voice.Delay.dat');
%Jdata=load('/NARGES-PQ_VoIP-DES-1_Voice.Jitter.dat');
%Pdata=load('/NARGES-PQ_VoIP-DES-1_IP.loss.dat');
%pici=55;

%% Loading, Initilisation & Transformation

    sd='/NARGES-PQ_VoIP-DES-1_Voice.Delay.dat';
    sj='/NARGES-PQ_VoIP-DES-1_Voice.Jitter.dat';
    sp='/NARGES-PQ_VoIP-DES-1_IP.loss.dat';
    Ddata=Ddata(:,2);Jdata=Jdata(:,2);Pdata=Pdata(:,2);

   winsize=10;
   rwin=sqrt(winsize);

%% Loading, Initilisation & Transformation


end % endif nargin
clc
%% Realtime Experiment Initializations

   win=winsize;
   rwin=sqrt(win);
   dsetn = pici;
   epochs=200; % for early stopping
   lr=0.0001; % Learning Rate Parameter
   mc=0.5; % Momentum Parameter
   lr_inc= 1.05; % MATLAB default
   lr_dec= 0.7; % MATLAB default
   goal=0;
   min_grad=1e-13;
   mu=0.005;
   mu_inc=10;
   mu_dec=0.3;
   mu_max=1e5;
   hn=10;
   time=inf;
   showCommandLine=1;
   max_fail= 30;

   show=1;

%end initializations

%% Transforming data and applying time windows of length "win" steps

%% Transforming Delay Tim-Series

TDdata = win_transform(Ddata,win);
TDsize = length(TDdata);
%MAXD = max(TDdata);

%% Transforming Jitter Tim-Series
TJdata = abs(win_transform(Jdata,win));
%TJsize = length(TJdata);
%MAXJ = max(TJdata);

%% Transforming Packet Loss Tim-Series
TPdata = win_transform(Pdata,win);
%maxAP=max(TPdata);
TPdataEnd = floor(2*length(TPdata)./3);
tlen_dfms2 = floor(length(TPdata)./3);
% end of packet loss transformation

TrainPdata=TPdata(tlen_dfms2+1:TPdataEnd);
```

```

close all;

%% Calling HDAX and calculating forcasted data of Delay and Jitter

FDdata = DFMS2(TDdata,sd,pici,dsetn,win);
pause;close;
FJdata = DFMS2(TJdata,sj,pici+276,dsetn,win);
pause;close;

%% Making MLP
Trainsize = floor(TDsize./3);
%Trainsize = floor(TDsize);
TestPdata=TPdata(TPdataEnd:length(TPdata));

TrainSet = [FDdata(1:Trainsize),FJdata(1:Trainsize),TrainPdata];
TestSet = [FDdata(Trainsize+1:length(FDdata)), ...
FJdata(Trainsize+1:length(FJdata)),TestPdata];
TrainOut = TrainPdata;

TrainSet = TrainSet';
TestSet = TestSet';
TrainOut = TrainOut';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NARGES = [TDdata,TJdata,TPdata];
data=NARGES';
Ranges = minmax(data);
Arch = [hn 1];
ActFunc = {'tansig','tansig'};

NARGESNet = newff(Ranges,TPdata' ,Arch, ActFunc, 'trainbr');
save NARGESNet;

 NARGESNet.trainParam.epochs = epochs;
 NARGESNet.trainParam.lr=lr;
 NARGESNet.trainParam.mc=mc;
 NARGESNet.trainParam.trainParam.goallr_inc=lr_inc;
 NARGESNet.trainParam.goal=goal;
 NARGESNet.trainParam.lr_dec=lr_dec;
 NARGESNet.trainParam.lr_inc=lr_inc;
 NARGESNet.trainParam.showCommandLine=showCommandLine;
 NARGESNet.trainParam.mu=mu;
 NARGESNet.trainParam.time=time;
 NARGESNet.trainParam.mu_inc=mu_inc;
 NARGESNet.trainParam.mu_dec=mu_dec;
 NARGESNet.trainParam.mu_max=mu_max;
 NARGESNet.trainParam.max_fail=max_fail;
 NARGESNet.trainParam.min_grad=min_grad;
 NARGESNet.trainParam.show=show;

tic
NARGESNet = train(NARGESNet, TrainSet,TrainOut);
Ellapsed = toc;
% end Making MLP

%% Start Simulation

out=sim(NARGESNet,TestSet);

out=out';

%% NRMSE and CCF for the Packet-Loss prediction
Error = Error1(out,TestPdata(:),'nrmse');
NRMSE = Error.NRMSE/rwin;
[r,lags]=xcorr(TestPdata(:),out,50,'coeff');
xc = max(r);


%% Calling ARMA Model
 phi=[1, - 0.05];
 theta= 0.005; %filter(1, phi, imp)';
tic
predseriesP=predarma2(TestPdata,phi,theta,1);
EllapsedP=toc;

%% NRMSE and CCF for ARMA (compared to NARGES)
ErrorP = Error1(predseriesP(1:end-1),TestPdata(:),'nrmse');
NRMSEP = ErrorP.NRMSE/rwin;
[rP,lags]=xcorr(TestPdata(:),predseriesP(1:end-1),50,'coeff');
xcP = max(rP);
```

```
%%end of Calling ARMA Model

%% start plotting %%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting NARGES, ARMA & Original traces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hold on
grid on
plot(TestPdata,'g.-','MarkerSize',18);
plot(out,'b+-','MarkerSize',18);
plot(predseriesP,'rx-','MarkerSize',18);
hold off
xlabel({'Step (s)'},'FontSize',18);
ylabel({'Average Packet-loss'},'FontSize',18);
title({['NARGES vs ARMA Prediction of Average ...
Packetloss: NARGES Error = ' num2str(NRMSE)   '; ARMA Error = ' ...
 num2str(NRMSEP)];['Data Set: ' num2str(dsetn) ...
 ' NARGES Correlation Coefficient= ' num2str(xc) ...
 ' ARMA Correlation Coefficient= ' num2str(xcP) ]},'FontSize',18);
    plotMeas = {'Original', 'NARGES', 'ARMA'};
    legend(plotMeas, 'Location', 'Northeast','FontSize',18);
    legend('boxoff');

%% make the figure bigger so it's easier to see

    set(gcf,'Units','normalized','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','NARGES Packetloss Prediction Model: copyright 2012, UTS');
    set(gcf,'PaperPositionMode','auto');
    set(gca, 'FontSize',18);

%% print the figure out

print('-depsc','-tiff','-r300',...
strcat('dataplot','_loss_',num2str(dsetn)));

    pause;close;
    grid off
%% end print the figure out

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% end of Plotting NARGES, ARMA & Original traces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting NARGES, ARMA & Original Boxplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    hold on
    notBoxPlot([TestPdata,out,predseriesP(1:end-1)]);
    title({['Data Set: ' num2str(dsetn) ...
    'Boxplot of Packet-Loss Data for Target data ...
    together with NARGES and ARMA Predictions']},'FontSize',18);

 % Add title and axis labels
    xlabel('Distributions','FontSize',18);
    ylabel(['Average Packet-Loss'],'FontSize',18);

    % Change the labels for the tick marks on the x-axis

    plotSpecies = {'Original', 'HDAX', 'ARMA'};
    set(gcf,'Units','normalised','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','NARGES Packetloss Prediction Model: copyright 2012, UTS');
    set(gca, 'XTick', 1:3, 'XTickLabel', plotSpecies,'FontSize',18);

    % Create labels for the legend
    plotMeas = {'SD', '95% Confidence Interval', 'Mean', 'Data Points'};
    legend(plotMeas, 'Location', 'Northeast','FontSize',18);
    legend('boxoff');
    set(gcf,'PaperPositionMode','auto');
    set(gca, 'FontSize',18);

%% print the figure out
   print('-depsc','-tiff','-r300',...
    strcat('notBoxPlot','_loss_',num2str(dsetn)));
%% end print the figure out

pause;close;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% end of Plotting NARGES, ARMA & Original Boxplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting NARGES, ARMA & Original CCF Stemplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


hold on
stem(lags,r,'g.-','MarkerSize',18);
stem(lags,rP,'rx-','MarkerSize',18);
title({['Data Set: ' num2str(dsetn)] ;
['Cross-Correlation Stemplot of predcition for ...
Packet-Loss Data']; [' NARGES Correlation Coefficient= ...
 ' num2str(xc) ' ARMA Correlation Coefficient= ' num2str(xcP)]},'FontSize',18);

    xlabel('Lags','FontSize',18);
    ylabel('Cross-Correlation Coefficient ','FontSize',18);

    % Change the labels for the tick marks on the x-axis

    %plotSpecies = {'Original', 'HDAX', 'ARMA'};
    set(gcf,'Units','normalised','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','NARGES Packetloss Prediction Model: copyright 2012, UTS');
    set(gca, 'FontSize',18);

    % Create labels for the legend
    plotMeas = {'ARMA', 'NARGES'};
    legend(plotMeas, 'Location', 'Northeastoutside','FontSize',18);
    legend('boxoff');
    set(gcf,'PaperPositionMode','auto');

print('-depsc','-tiff','-r300',...
strcat('xcorrstem','_loss_',num2str(dsetn)));
hold off
pause;close;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% end of Plotting NARGES, ARMA & Original CCF Stemplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% end Plotting %%%

%% filing output %%
    Sout1 = num2str(Ellapsed);
    Sout2 = num2str(NRMSE);
    Sout3 = num2str(EllapsedP);
    Sout4 = num2str(NRMSEP);
    fid = fopen('NARGESARMA.out', 'a');
    fprintf(fid, '%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n',...
    num2str(dsetn),Sout1,Sout2, Sout3, Sout4,xc,xcP, sp);
    fclose(fid);%pause(10);
%% end filing output %%
end
%% NARGES END %%
```

C.2 HDAX Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright and terms of use (DO NOT REMOVE):
% The code is made freely available for non-commercial
% uses only, provided that the copyright
% header in each file not be removed, and suitable
% citation(s) (see below) be made for papers
% published based on the code.
%
% Copyright (c) 2009-2012, Hooman Homayounfard,
%                           hoomanhm@it.uts.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% HDAX Main Function

function forecastedY= DFMS2(data,ss,pici,dsetn,win)
if nargin == 0
   win=10;
   data = win_transform(data,win);
   rwin=sqrt(win);
end

%initialization
   rwin=sqrt(win);
   tlen = floor(length(data)./4);  % %25 of data for training
   datat = data(1:tlen);
   datas = data(tlen:slen);
   stds = std(datat);
   maxY=max(data);

% for writing on plots
   outdatatype = 'delay';
   outdataunit = 'ms';

   if (pici-dsetn > 0)
 outdatatype = 'jitter';
 outdataunit = 'sec';
   end
% for writing on plots

% end of initialization

% Start of training Phase
freq_table = DFM(datat,maxY);
% End of training Phase

%% Start of estimation with HDAX
tic

lens=length(datas);
    shifted=datas;
     shifted(1:lens-1,1)=datas(2:lens,1);
        shifted=shifted(1:lens-1,1);% Shifted array of the input values
            Y=shifted-datas(1:lens-1);  % Calculating the difference between
   % two adjacent values in the time-series
lenY=lens-1;
scasenumber=zeros(lenY,1);
fcasenumber=zeros(lenY,1);

% finding the trend cases

for i=1:lenY
    scasenumber(i,1)=findcase(Y(i,1)/maxY);
end

% defining the Y values level 'fyt' function

FYt=zeros(lens,1);
for i=1:lens
```

```
    FYt(i,1)=findFYt(datas(i,1),maxY,4);
end


% Estimating the next delay value. For estimation
% we have used 1/10 of the mean value of each case group

forecastedY=zeros(lens,1);
%forecastedY(2)=datas(2);

for i=3:lenY
   fcasenumber(i+1,1)= forecase(freq_table,FYt(i,1), ...
    scasenumber(i-1,1),scasenumber(i,1));
   temp = Y(i,1)/maxY;

   if (fcasenumber(i+1,1) == 0)
    forecastedY(i+1)= datas(i);
   end
   if (fcasenumber(i+1,1) == 1)
    forecastedY(i+1)= datas(i)+(maxY/8)-stds;
   end
   if (fcasenumber(i+1,1) == 2)
    forecastedY(i+1)= datas(i)+(3*maxY/8)-3*stds;
   end
   if (fcasenumber(i+1,1) == 3)
    forecastedY(i+1)= datas(i)-(maxY/8)+stds;
   end
   if (fcasenumber(i+1,1) == 4)
    forecastedY(i+1)= datas(i)-(3*maxY/8)+3*stds;
   end
   if (fcasenumber(i+1,1) == 5)
      if (temp<-1)
   forecastedY(i+1)= datas(i)-maxY+stds;
  end
  if (temp>1)
   forecastedY(i+1)= datas(i)+maxY-stds;
  end
   end
% End of estimation with HDAX
end
Ellapsed1 = toc;
```

```

% adjustments of the two arrays of target data and estimated data

forecastedY(i-2)=forecastedY(i+1);
forecastedY = reshape(forecastedY,lenY+1,1);

%% NRMSE and CCF for the forecasted delay or jitter time-series
ErrorH = Error1(forecastedY(:),datas(:),'nrmse');
NRMSEH = ErrorH.NRMSE/rwin;
[rH,lags]=xcorr(datas(:),forecastedY(:),50,'coeff');
xcH = max(rH);


%% Calling ARMA for HDAX Predictor
 phi=[1, - 0.9]; %imp=[1;zeros(9,1)];
 theta= 0.02; %filter(1, phi, imp)';
tic
predseries=predarma2(datas,phi,theta,1);

%% NRMSE and CCF for ARMA (compared to HDAX) is calculated in here

ErrorA = Error1(predseries(1:end-1),datas(:),'nrmse');
NRMSEA = ErrorA.NRMSE/rwin;
[rA,lags]=xcorr(datas(:),predseries(1:end-1),50,'coeff');
xcA = max(rA);
Ellapsed2=toc;

%% end Calling ARMA for HDAX Predictor

%% start plotting %%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting HDAX, ARMA & Original traces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    hold on
    grid on
    plot(datas,'g.-','MarkerSize',18);
    plot(forecastedY,'b+-','MarkerSize',18);
    plot(predseries(1:end-1),'rx-','MarkerSize',18)
    xlabel({'Step (s)'},'FontSize',18);
    ylabel({['Average ' outdatatype ' (' outdataunit ')']},'FontSize',18);
    title({['Dataplot of ' outdatatype ' ...
    target dataset ' num2str(dsetn) ', ...
    together with HDAX and ARMA forecasts'];...
    ['HDAX vs ARMA - HDAX Fitness NRMSE= ' ...
    num2str(NRMSEH)  ' & ARMA Fitness NRMSE= ' ...
    num2str(NRMSEA)];['HDAX Correlation Coefficient= ' ...
    num2str(xcH) ' ARMA Correlation Coefficient= ' num2str(xcA)]},'FontSize',18);
    plotMeas = {'Original', 'HDAX','ARMA'};
    legend(plotMeas, 'Location', 'Northeast', 'FontSize',16);
    legend('boxoff');

    % make the figure bigger so it's easier to see
    set(gcf,'Units','normalized','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','HDAX Approximation: copyright 2012, UTS');
    set(gcf,'PaperPositionMode','auto');
    set(gca, 'FontSize',18);
%% Print the figure out
    print('-depsc','-tiff','-r300',...
    strcat('dataplot_',outdatatype,'_',num2str(dsetn)));
    %print('-depsc','-tiff','-r300',num2str(pici));
    pause;close;
    grid off
%% end of printing the figure

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting HDAX, ARMA & Original Boxplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    hold on
    notBoxPlot([datas,forecastedY,predseries(1:end-1)]);...
    title({['Data Set: ' num2str(dsetn) ' Boxplot of ...
    ' outdatatype ' data for Target data together ...
    with HDAX  and ARMA Forecasts']},'FontSize',18);
    % Add title and axis labels
    xlabel('Distributions','FontSize',18);
    ylabel(['Average ' outdatatype  ' (' outdataunit ')'],'FontSize',18);

    % Change the labels for the tick marks on the x-axis

    plotSpecies = {'Original', 'HDAX', 'ARMA'};
    set(gcf,'Units','normalized','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','HDAX Approximation: copyright 2012, UTS');
    set(gca, 'XTick', 1:3, 'XTickLabel', plotSpecies,'FontSize',18);

    % Create labels for the legend
    plotMeas = {'SD', '95% Confidence Interval', 'Mean Line', 'Data'};
    legend(plotMeas, 'Location', 'Northeast', 'FontSize',16);
    legend('boxoff');
    set(gcf,'PaperPositionMode','auto');
    set(gca, 'FontSize',18);

%% print the boxplots out
    print('-depsc','-tiff','-r300',...
    strcat('notBoxPlot_',outdatatype,'_',num2str(dsetn)));
%% end of Printing
    pause;close;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% end Plotting HDAX, ARMA & Original Boxplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plotting HDAX, ARMA & Original CCF Stemplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   hold on
   stem(lags,rA,'rx-','MarkerSize',18); ...
   title({['Data Set: ' num2str(dsetn) ' Cross-Correlation ...
   Stemplot of Forecasts for ' outdatatype ' Data'] ; ...
   [' HDAX Correlation Coefficient= ' num2str(xcH) ...
   ' ARMA Correlation Coefficient= ' num2str(xcA)]},'FontSize',18);
   stem(lags,rH,'g.-','MarkerSize',18);

    xlabel('Lags','FontSize',18);
    ylabel('Cross-Correlation Coefficient','FontSize',18);

    % Change the labels for the tick marks on the x-axis

    %plotSpecies = {'Original', 'HDAX', 'ARMA'};
    set(gcf,'Units','normalised','Position',[0.02  0.0467  0.95  0.85]);
    set(gcf,'name','Cross-Correlation of HDAX vs ARMA : copyright 2012, UTS');
    %set(gca, 'XTick', 1:3, 'XTickLabel', plotSpecies);

    % Create labels for the legend
    plotMeas = {'ARMA', 'HDAX'};
    legend(plotMeas, 'Location', 'Northeastoutside', 'FontSize',16);
    legend('boxoff');
    set(gcf,'PaperPositionMode','auto');
    set(gca, 'FontSize',18);
   hold off
   print('-depsc','-tiff','-r300',...
   strcat('xcorrstem_',outdatatype,'_',num2str(dsetn)));
   pause;close;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% end of Plotting HDAX, ARMA & Original CCF Stemplots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% end Plotting %%

% filing output for HDAX & ARMA
    Sout1 = num2str(Ellapsed1);
    Sout2 = num2str(NRMSEH);
    Sout3 = num2str(Ellapsed2);
    Sout4 = num2str(NRMSEA);
    fid = fopen('NARGESARMA.out', 'a');
    fprintf(fid, '%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n', ...
    num2str(dsetn),Sout1,Sout2, Sout3, Sout4,xcH, xcA, ss);
    fclose(fid);%pause(10);
% end filing output for HDAX & ARMA
end
%% HDAX END %%
```

C.3 HDAX Functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright and terms of use (DO NOT REMOVE):
% The code is made freely available for non-commercial
% uses only, provided that the copyright
% header in each file not be removed, and suitable
% citation(s) (see below) be made for papers
% published based on the code.
```

```
%
% Copyright (c) 2009-2012, Hooman Homayounfard,
%                          hoomanhm@it.uts.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Training of Look-up Table

function [outArray] = DFM(data,maxY)

   if nargin == 0
     clear all
      data = win_transform(temp);
      maxY=max(data);
   end

tic

% Initialization
alphabet_size=6;
    len=length(data);
        shifted=data;
            shifted(1:len-1,1)=data(2:len,1);
        shifted=shifted(1:len-1,1);
    Y=shifted-data(1:len-1);
lenY=len-1;
casenumber=zeros(lenY,1);

for i=1:lenY
    casenumber(i,1)=findcase(Y(i,1)/maxY);
end

FYt=zeros(len,1);

for i=1:len
    FYt(i,1)=findFYt(data(i,1),maxY,4);
end

outArray=zeros(alphabet_size^3*5,1);

for i=3:len-1
    index=FYt(i,1)*alphabet_size^3+casenumber(i-2,1)* ...
    alphabet_size^2+casenumber(i-1,1)*alphabet_size+casenumber(i,1)+1;
    outArray(index,1)=outArray(index,1)+1;
end
toc
end

%*************%

%% Finding Case Number of Time-Series Trend

function [casenumber]=findcase(y_max)
if (y_max==0)
    casenumber=0;
    return
end
if (y_max>0 && y_max<=0.5)
    casenumber=1;
    return
end
if (y_max>0.5 && y_max<=1)
    casenumber=2;
    return
end
if (y_max>-0.5 && y_max<0)
    casenumber=3;
    return
end
if (y_max>=-1 && y_max<-0.5)
    casenumber=4;
    return
end
if (y_max<-1 || y_max>1)
    casenumber=5;
    return
end

end
function [FYt]=findFYt(y,max,interval)
 FYt=floor(y/(max/interval));
    if (FYt>4)
      FYt=4;
 end
end

%*************%

%% Applying Moving Average Transformation on target Time-Series

function [ave]= win_transform(data,winsize)
%% if nurgin
if nargin == 0
   data = load ('/delay_adsl-to-wired-recv-const-1-udp-512.log.dat');
   data=data(:,3);
   winsize=10;
end

max = size(data);
    index=floor(max./winsize);
        ave(1:index+1)=0;
            sumdata = 0;

j=winsize;
t=1;

for i=1:max,
    if j > 0
     sumdata = data(i) + sumdata;
        j = j-1;
    elseif j == 0
     ave(t)=sumdata/winsize;
        t=t+1;
        sumdata=0;
    j=10;
end
end
ave=ave';
end

%*************%

%% Finding most frequent trend

```

```
function [fcasenumber] = forecase(freq_table,FYt,scasenumber1,scasenumber2)
max=0;
alphabet_size=6;
for i=0:5
    index=FYt*alphabet_size^3+scasenumber1*...
    alphabet_size^2+scasenumber2*alphabet_size+i+1;
    if (freq_table(index)> max)
   max = freq_table(index);
   fcasenumber = i;
    end

end
if (max == 0)
   fcasenumber = scasenumber2; %floor(rand*100/20);
   % if all the possible cases have 0 frequency we ...
   % make a random case number between 0 to 4
   % assigning the current case to the next one ...
   % in the absence of frequency for all possible cases
end
return
end

%*************%
```

C.4 Error function

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright and terms of use (DO NOT REMOVE):
% The code is made freely available for non-commercial
% uses only, provided that the copyright
% header in each file not be removed, and suitable
% citation(s) (see below) be made for papers
% published based on the code.
%
% Copyright (c) 2009-2012, Hooman Homayounfard,
%                           hoomanhm@it.uts.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Error=Error1(P,T,Comment)
```

```

if isempty(Comment)
    Comment='missing data: Target and Predicted';
end

S=size(P);
if S(1)<S(2)
    P=P';
end

S=size(T);
if S(1)<S(2)
    T=T';
end

n=size(P,1);
% Measures of error
SSE=sum((P-T).^2);
RMSE=sqrt(SSE/size(P,1));
StdT=std(T,1);
StdP=std(P,1);
MaxT=max(T);
MaxP=max(P);
MAX=max(MaxP,MaxT);
MinT=min(T);
MinP=min(P);
MIN=min(MinP,MinT);
NRMSE=100*RMSE/(MAX - MIN); %sqrt(SSE/sum((T-mean(T)).^2));
NSC=1-SSE/sum((T-mean(T)).^2);
Cor=sum((P-mean(P)).*(T-mean(T)))/...
(sqrt(sum((P-mean(P)).^2))*sqrt(sum((T-mean(T)).^2)));

MAE=sum(abs(P-T))/size(P,1);
MARE=sum(abs((T-P)./T))/n;

MuT=mean(T);
MuP=mean(P);

%Calculating PERS
P2=T(1:end-1,:);
T2=T(2:end,:);
```

```
SSEN=sum((P2-T2).^2);
PERS=1-(SSE/SSEN);
RMSEN=sqrt(SSEN/(n-1));
NRMSEN=100*RMSEN/std(T2,1);

%Passing the output structure
Error.RMSE=RMSE;
Error.NSC=NSC;
Error.Cor=Cor;
Error.NRMSE=NRMSE;
Error.MAE=MAE;
Error.StdT=StdT;
Error.StdP=StdP;
Error.MuT=MuT;
Error.MuP=MuP;
Error.PERS=PERS;
Error.SSE=SSE;
Error.SSEN=SSEN;  %Sum Squared Error Naive
Error.RMSEN=RMSEN; %RMSE Naive
Error.NRMSEN=NRMSEN; %NRMSE Naive
Error.MARE=MARE;
Error.Er=T-P;
Io=find(Error.Er<=0);
Iu=find(Error.Er>0);

S1=size(Io,1);
S2=size(Iu,1);

Error.Po=S1/size(Error.Er,1);
Error.Pu=S2/size(Error.Er,1);
```

C.5 Running Experiments

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Copyright and terms of use (DO NOT REMOVE):
% The code is made freely available for non-commercial
% uses only, provided that the copyright
% header in each file not be removed, and suitable
% citation(s) (see below) be made for papers
% published based on the code.
```

```
%
% Copyright (c) 2009-2012, Hooman Homayounfard,
%                          hoomanhm@it.uts.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function NARGES_F

s = {'/delay_adsl-to-wired-recv-const-1-udp-64.log.dat';
    '/jitter_adsl-to-wired-recv-const-1-udp-64.log.dat';
    '/packetloss_adsl-to-wired-recv-const-1-udp-64.log.dat';

    '/delay_adsl-to-wired-recv-const-1-udp-512.log.dat';
    '/jitter_adsl-to-wired-recv-const-1-udp-512.log.dat';
    '/packetloss_adsl-to-wired-recv-const-1-udp-512.log.dat';

    '/delay_gprs-to-wired-winlin-tcp-1024.log.dat';
    '/jitter_gprs-to-wired-winlin-tcp-1024.log.dat';
    '/packetloss_gprs-to-wired-winlin-tcp-1024.log.dat';

    '/delay_gprs-to-wired-winlin-udp-512.log.dat';
    '/jitter_gprs-to-wired-winlin-udp-512.log.dat';
    '/packetloss_gprs-to-wired-winlin-udp-512.log.dat';

    '/delay_gprs-to-wired-winlin-udp-1024.log.dat';
    '/jitter_gprs-to-wired-winlin-udp-1024.log.dat';
    '/packetloss_gprs-to-wired-winlin-udp-1024.log.dat';

    '/delay_gprs-to-wired-winwin-tcp-512.log.dat';
    '/jitter_gprs-to-wired-winwin-tcp-512.log.dat';
    '/packetloss_gprs-to-wired-winwin-tcp-512.log.dat';

    '/delay_gprs-to-wired-winwin-tcp-1024.log.dat';
    '/jitter_gprs-to-wired-winwin-tcp-1024.log.dat';
    '/packetloss_gprs-to-wired-winwin-tcp-1024.log.dat';

    '/delay_gprs-to-wired-winwin-tcp-512.log.dat';
    '/jitter_gprs-to-wired-winwin-tcp-512.log.dat';
    '/packetloss_gprs-to-wired-winwin-tcp-512.log.dat';

    '/delay_gprs-to-wired-winwin-tcp-1024.log.dat';
    '/jitter_gprs-to-wired-winwin-tcp-1024.log.dat';
    '/packetloss_gprs-to-wired-winwin-tcp-1024.log.dat';

    '/delay_umts-to-wired-tcp-64.log.dat';
    '/jitter_umts-to-wired-tcp-64.log.dat';
    '/packetloss_umts-to-wired-tcp-64.log.dat';

    '/delay_umts-to-wired-tcp-512.log.dat';
    '/jitter_umts-to-wired-tcp-512.log.dat';
    '/packetloss_umts-to-wired-tcp-512.log.dat';

    '/delay_umts-to-wired-tcp-1024.log.dat';
    '/jitter_umts-to-wired-tcp-1024.log.dat';
    '/packetloss_umts-to-wired-tcp-1024.log.dat';

    '/delay_umts-to-wired-udp-64.log.dat';
    '/jitter_umts-to-wired-udp-64.log.dat';
    '/packetloss_umts-to-wired-udp-64.log.dat';

    '/delay_umts-to-wired-udp-512.log.dat';
    '/jitter_umts-to-wired-udp-512.log.dat';
    '/packetloss_umts-to-wired-udp-512.log.dat';


    '/delay_wired-to-adsl-recv_const-1-tcp-64.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-tcp-64.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-tcp-64.log.dat';

    '/delay_wired-to-adsl-recv_const-1-tcp-256.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-tcp-256.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-tcp-256.log.dat';

    '/delay_wired-to-adsl-recv_const-1-tcp-1024.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-tcp-1024.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-tcp-1024.log.dat';

    '/delay_wired-to-adsl-recv_const-1-udp-64.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-udp-64.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-udp-64.log.dat';

    '/delay_wired-to-adsl-recv_const-1-udp-512.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-udp-512.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-udp-512.log.dat';

    '/delay_wired-to-adsl-recv_const-1-udp-1024.log.dat';
    '/jitter_wired-to-adsl-recv_const-1-udp-1024.log.dat';
    '/packetloss_wired-to-adsl-recv_const-1-udp-1024.log.dat';

    '/delay_wired-to-gprs-linwin-tcp-64.log.dat';
    '/jitter_wired-to-gprs-linwin-tcp-64.log.dat';
    '/packetloss_wired-to-gprs-linwin-tcp-64.log.dat';

    '/delay_wired-to-gprs-linwin-tcp-512.log.dat';
    '/jitter_wired-to-gprs-linwin-tcp-512.log.dat';
    '/packetloss_wired-to-gprs-linwin-tcp-512.log.dat';

    '/delay_wired-to-gprs-linwin-tcp-1024.log.dat';
    '/jitter_wired-to-gprs-linwin-tcp-1024.log.dat';
    '/packetloss_wired-to-gprs-linwin-tcp-1024.log.dat';

    '/delay_wired-to-gprs-linwin-udp-64.log.dat';
    '/jitter_wired-to-gprs-linwin-udp-64.log.dat';
    '/packetloss_wired-to-gprs-linwin-udp-64.log.dat';

    '/delay_wired-to-gprs-linwin-udp-512.log.dat';
    '/jitter_wired-to-gprs-linwin-udp-512.log.dat';
    '/packetloss_wired-to-gprs-linwin-udp-512.log.dat';

    '/delay_wired-to-gprs-linwin-udp-1024.log.dat';
    '/jitter_wired-to-gprs-linwin-udp-1024.log.dat';
    '/packetloss_wired-to-gprs-linwin-udp-1024.log.dat';


    '/delay_wired-to-gprs-winwin-udp-64.log.dat';
    '/jitter_wired-to-gprs-winwin-udp-64.log.dat';
    '/packetloss_wired-to-gprs-winwin-udp-64.log.dat';


    '/delay_wired-to-umts-tcp-64.log.dat';
    '/jitter_wired-to-umts-tcp-64.log.dat';
    '/packetloss_wired-to-umts-tcp-64.log.dat';

    '/jitter_wired-to-umts-tcp-512.log.dat';
    '/delay_wired-to-umts-tcp-512.log.dat';
    '/packetloss_wired-to-umts-tcp-512.log.dat';


    '/delay_wired-to-wireless-sctp-64.log.dat';
    '/jitter_wired-to-wireless-sctp-64.log.dat';
    '/packetloss_wired-to-wireless-sctp-64.log.dat';

    '/delay_wired-to-wireless-sctp-512.log.dat';
    '/jitter_wired-to-wireless-sctp-512.log.dat';
    '/packetloss_wired-to-wireless-sctp-512.log.dat';

    '/delay_wired-to-wireless-sctp-1024.log.dat';
    '/jitter_wired-to-wireless-sctp-1024.log.dat';
    '/packetloss_wired-to-wireless-sctp-1024.log.dat';

    '/delay_wired-to-wireless-tcp-64.log.dat';
    '/jitter_wired-to-wireless-tcp-64.log.dat';
    '/packetloss_wired-to-wireless-tcp-64.log.dat';

    '/delay_wired-to-wireless-tcp-512.log.dat';
    '/jitter_wired-to-wireless-tcp-512.log.dat';
    '/packetloss_wired-to-wireless-tcp-512.log.dat';


    '/delay_wired-to-wireless-udp-64.log.dat';
    '/jitter_wired-to-wireless-udp-64.log.dat';
    '/packetloss_wired-to-wireless-udp-64.log.dat';

    '/delay_wired-to-wireless-udp-512.log.dat';
    '/jitter_wired-to-wireless-udp-512.log.dat';
    '/packetloss_wired-to-wireless-udp-512.log.dat';


    '/NARGES-DiffSrv-DES-1_Voice.Delay.dat';
    '/NARGES-DiffSrv-DES-1_Voice.Jitter.dat';
    '/NARGES-DiffSrv-DES-1_IP.loss.dat';

    '/NARGES-Diffsrv_Video-DES-1_Voice.Delay.dat';
    '/NARGES-Diffsrv_Video-DES-1_Voice.Jitter.dat';
    '/NARGES-Diffsrv_Video-DES-1_IP.loss.dat';

    '/NARGES-DiffSrv_VoIP-DES-1_Voice.Delay.dat';
    '/NARGES-DiffSrv_VoIP-DES-1_Voice.Jitter.dat';
    '/NARGES-DiffSrv_VoIP-DES-1_IP.loss.dat';

    '/NARGES-FIFO-DES-1_Voice.Delay.dat';
    '/NARGES-FIFO-DES-1_Voice.Jitter.dat';
    '/NARGES-FIFO-DES-1_IP.loss.dat';

    '/NARGES-FIFO_Long-DES-1_Voice.Delay.dat';
    '/NARGES-FIFO_Long-DES-1_Voice.Jitter.dat';
    '/NARGES-FIFO_Long-DES-1_IP.loss.dat';

    '/NARGES-FIFO_VIdeo-DES-1_Voice.Delay.dat';
    '/NARGES-FIFO_VIdeo-DES-1_Voice.Jitter.dat';
    '/NARGES-FIFO_VIdeo-DES-1_IP.loss.dat';

    '/NARGES-FIFO_VoIP-DES-1_Voice.Delay.dat';
    '/NARGES-FIFO_VoIP-DES-1_Voice.Jitter.dat';
    '/NARGES-FIFO_VoIP-DES-1_IP.loss.dat';

    '/NARGES-PQ-DES-1_Voice.Delay.dat';
    '/NARGES-PQ-DES-1_Voice.Jitter.dat';
    '/NARGES-PQ-DES-1_IP.loss.dat';

    '/NARGES-PQ_Video-DES-1_Voice.Delay.dat';
    '/NARGES-PQ_Video-DES-1_Voice.Jitter.dat';
    '/NARGES-PQ_Video-DES-1_IP.loss.dat';

    '/NARGES-PQ_VoIP-DES-1_Voice.Delay.dat';
    '/NARGES-PQ_VoIP-DES-1_Voice.Jitter.dat';
    '/NARGES-PQ_VoIP-DES-1_IP.loss.dat'};


        % experiment with data from D-ITG


count=1;

for i = 1:36

```

```
        Ddata = load (s{count});
            Ddata=Ddata(:,3);
                Jdata = load (s{count+1});
                Jdata=Jdata(:,3);
            Pdata = load (s{count+2});
        Pdata=Pdata(:,3);

    winsize=10;

NARGES130(Ddata,Jdata,Pdata,winsize,s{count},s{count+1},s{count+2},i);
       count = count+3;

end

        %more experiment with data from OPNET 2012
count = 109;

for i = 37:46

         Ddata = load (s{count});
        Ddata=Ddata(:,2);
        Jdata = load (s{count+1});
        Jdata=Jdata(:,2);
        Pdata = load (s{count+2});
        Pdata=Pdata(:,2);
        winsize=24;
        NARGES130(Ddata,Jdata,Pdata,winsize,s{count},s{count+1},s{count+2},i);
        count = count+3;
end

end
```

C.6 ARMA Implementation

The following ARMA implementation adapted from Brockwell and Davis (2006).

```
function [predseries]=predarma2(time-series,phi,theta,yesmean)
% PREDARMA One step prediction of an ARMA time-series
%
```

```
%           PREDARMA(A,P,Q,YM) where A,P,Q are vectors generates
%           the one step ahead prediction of the ARMA time-series A. The
%           vectors P and Q are the coefficients of the ARMA time-series.
%           If YM==1, the mean is substracted from the time-series
%           before prediction, but if YM==0 it is not.  The default
%           if YM is not supplied is YM=1
%
% Usage: [wssq,predseries,errors,av,r]=predarma(timeseries,phi,theta,yesmean)
%           wssq is the weighted sum of squares of errors, predseries is
%           the predicted series, from the supplied time-series, errors
%           is the differences (timeseries-predseries), av is the average
%           value of timeseries, and r is a vector of relative variances
%           of the first few errors (since r->1, there is no point in r
%           being as long as timeseries).
%
%           See also ACF, XCOV, DURBLEV, PACF, ARMAESTIMATE, XMAESTIMATE,
%           PREDAR.

%           Author :P Shelton :10:02:95
%                   R. G. Addie: March 1995, August 1996.
%           References
%                P.J. Brockwell and R.A. Davies,
%                Time Series: Theory and Methods, Spriger-Verlag, 2006,
%
%           Last Changes made by: Hooman Homayounfard, UTS, Sydney, 2009
tic
fprintf('.');
tslen=length(timeseries);
if (nargin<=3) yesmean=1; end;
if (yesmean)
av=mean(timeseries);              % removing the mean
timeseries=timeseries-av;
end;
predseries=zeros(tslen,1);
errors=zeros(tslen,1);
p=length(phi);                    % the order of the AR part of the ARMA model
q=length(theta);                 % the order of the MA part
                         % note that the first coeff=1 is implied
m=max(p,q);

gamma=acvf(phi,theta,1,5*m); % get the acvf of the time-series model
```

```
maacvf = acvf([],theta,1,2*m);  % the autocov of a MA series with coeffs theta
            % the autocovariance matrix kappa
Nthetas = 4*m;
Nkappas = Nthetas+1;
kappa=zeros(Nkappas);
for i=1:Nkappas,     % the upper limit depends on the length of the time-series
                % Nkappas should be enough
    for j=1:Nkappas,
        if (i<=m)&(j<=m),
            kappa(i,j)=gamma(1+abs(i-j));
        else
            if (min(i,j)<=m)&(m<max(i,j))&(max(i,j)<=(2*m)),
                partgamma=zeros(1,p);
                partgamma(1:abs(i-j))=gamma(abs(i-j):-1:1);
                partgamma(abs(i-j)+1:p)=gamma(2:p-abs(i-j)+1);
                kappa(i,j)=gamma(1+abs(i-j));
                        if (p>=1)
                kappa(i,j)=kappa(i,j)-phi*partgamma(1:p)';
                        end;
            else
                 if (min(i,j)>m & abs(i-j)+1<=length(maacvf)),
                      kappa(i,j)=maacvf(abs(i-j)+1);
                 else
                      kappa(i,j)=0;
                 end
            end
        end
    end
end
            % the rest of this function finds the one step prediction
            % of the ARMA(p,q) model recursively.
[thetapre,v]=inn(kappa);
errors(1)=timeseries(1);
ssq=errors(1)^2;
%
%       for the first 4*m predictions we use a slow method
%       which is capable of handling the initial transient
%       See Brockwell and Davis 2nd Ed., page 176.
for n=1:m
        predseries(n+1) = thetapre(n,1:n)*errors(n:-1:1);
        errors(n+1) = timeseries(n+1)-predseries(n+1);
```

```
end;
for n=m+1:min(Nthetas-1,tslen)
        if (q>0)
                predseries(n+1) = thetapre(n,1:q)*errors(n:-1:n-q+1);
        end;
        if (p>0)
                predseries(n+1) = predseries(n+1) + phi*timeseries(n:-1:n-p+1);
        end;
        if (n<tslen)
                errors(n+1) = timeseries(n+1)-predseries(n+1);
        end;
end;

if (tslen>Nthetas)
e = errors(Nthetas:-1:Nthetas+1-q);
x = timeseries(Nthetas:-1:Nthetas+1-p);

if (0)          % this version replaced by something better
if (p>0 & q>0)
        % fprintf('p = %d, q = %d\n', p, q);
        for k=Nthetas+1:tslen
                predseries(k) = theta*e + phi*x;
                errors(k) = timeseries(k)-predseries(k);
                x = [timeseries(k);x(1:p-1)];
                e=[errors(k);e(1:q-1)];
        end;
elseif (p>0)
        % this should be especially fast because matlab handles the loop
        preds = conv(phi,timeseries')';
        predseries(m+2:tslen) = preds(m+1:tslen-1);
        errors = timeseries-predseries;
elseif (q>0)
        for k=Nthetas:tslen
                predseries(k) = theta*e;
                errors(k) = timeseries(k)-predseries(k);
                e=[errors(k);e(1:q-1)];
        end;
else
        predseries = zeros(size(timeseries));
        errors = timeseries-predseries;
end;
```

```

else            % simpler version
        if (q>0)
                % fprintf('p = %d, q = %d\n', p, q);
                for n=Nthetas+1:tslen
                  predseries(n+1) = theta*errors(n:-1:n-q+1);
                  if (p>0)
                    predseries(n+1)=predseries(n+1)+phi*timeseries(n:-1:n-p+1);
                  end;
                  if (n<tslen)
                        errors(n+1) = timeseries(n+1)-predseries(n+1);
                  end;
                end;
        elseif (p>0)
                % this should be especially fast because matlab handles the loop
                preds = conv(phi,timeseries')';
                predseries(m+2:tslen) = preds(m+1:tslen-1);
                errors = timeseries-predseries;
        else
                predseries = zeros(size(timeseries));
                errors = timeseries-predseries;
        end;
end;

wssq = errors(Nthetas+1:tslen)'*errors(Nthetas+1:tslen) ...
        + sum((errors(1:Nthetas).*errors(1:Nthetas))./v(1:Nthetas));


if (yesmean)
        predseries = predseries+av;
end;
Ellapsed=toc;
end;
```

Bibliography

E. Aboelela. *Network simulation experiments manual*. Morgan Kaufmann, 2011.

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, June 1993.

J. Antari and A. Zeroual. Modelling video packet transmission in IP networks using Hammerstein series and higher order cumulants. *AEU-International Journal of Electronics and Communications*, 63(5):406–411, 2009.

J. Antoniou, C. Chrysostomou, and N. Jacovides. Specification of Simulation Environment. Technical Report D4.1, SEACORN, 2002.

R. G. Ash. *Dynamic Routing in Telecommunications Networks*. McGraw-Hill Professional, 1997.

I. Batyrshin and L. Sheremetov. Perception based time series data mining with map transform. In *MICAI 2005: Advances in Artificial Intelligence*, volume 3789 of *LNCS*, pages 514–523. Springer Berlin / Heidelberg, 2005.

I. Z. Batyrshin and L. B. Sheremetov. Perception–based approach to time series data mining. *Applied Soft Computing Journal*, 8(3):1211–1221, 2008.

D. P. Bertsekas and R. G. Gallager. *Data networks*. Prentice Hall, Englewood Cliffs, N.J, 1992.

S. Biaz and N. H. Vaidya. Distinguishing congestion losses from wireless transmission losses: A negative result. In *7th Int. Conf. Computer Communications and Networks*, pages 722–731, Lafayette, LA, 1998.

V. Borges, M. Curado, and E. Monteiro. The impact of interference-aware routing metrics on video streaming in wireless mesh networks. *Ad Hoc Networks*, 9(4):652–661, 2011.

A. Botta, A. Dainotti, and A. Pescapé. Multi-protocol and multi-platform traffic generation and measurement. In *Infocom '07 DEMO Session*, volume 45, pages 526–532, Anchorage, Alaska, USA, 2007.

A. Botta, A. Pescapé, and G. Ventre. Quality of service statistics over heterogeneous networks: Analysis and applications. *European Journal of Operational Research*, 191 (3):1075–1088, 2008.

A. Botta, A. Dainotti, and A. Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531 – 3547, 2012.

G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley series in probability and statistics. John Wiley, Hoboken, N.J., 4th edition, 2011.

L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: new techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24:24–35, October 1994.

P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, New York, 2006.

H. Cao, N. Mamoulis, and D.W. Cheung. Mining frequent spatio-temporal sequential patterns. In *Fifth IEEE International Conference on Data Mining*, page 8289, 2005.

G. D. Caro. *Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks*. PhD thesis, Université libre de Bruxelles, 2004.

G. D. Caro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In A.E. Eiben et al, editor, *Parallel Problem Solving from Nature PPSN V, LNCS*, volume 1498, pages 673–682. Springer Berlin / Heidelberg, Berlin, 1998.

R. L. Carter and M. E. Crovella. *Measuring bottleneck link speed in packet-switched networks*, volume 27-28. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 1996.

C. Chatfield. *Time-series forecasting*. Chapman and Hall, 2001.

C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall/CRC, 2004.

Y. Chi, H. Wang, P.S. Yu, and R.R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Fourth IEEE International Conference on Data Mining, ICDM'04.*, pages 59–66. IEEE, 2004.

R. G. Cole and J. H. Rosenbluth. Voice over IP performance monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, 2001.

J. Debenham and S. Simoff. Informed agents: Integrating data mining and agency. In C. Boukis, L. Pnevmatikakis, and L. Polymenakos, editors, *International Federation for Information Processing–IFIP*, volume 247, pages 165–173. Springer–Verlag, Boston, 2007.

J. Debenham, S. Simoff, J. Leaney, and V. Mirchandani. Smart communications network management through a synthesis of distributed intelligence and information. *Artificial Intelligence in Theory and Practice II*, pages 415–419, 2008.

A. D. Dongare, R. R. Kharde, and A. D. Kachare. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2, 2012.

C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *IEEE INFOCOM'01*, volume 2, pages 905–914, 2001.

H. Dutta. Empowering Scientific Discovery by Distributed Data Mining on the Grid Infrastructure. In A. Hanemann, B. Kratz, M. Mukhi, and T. Dumitras, editors, *ICSOC 2006*, volume RC24118, pages 25–30. IBM Press, 2006.

C. Estan, S. Savage, and G. Varghese. Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM'03, pages 137–148, Karlsruhe, Germany, 2003.

U. Fayyad, D. Haussler, and P. Stolorz. Mining scientific data. *Communications of the ACM*, 39(11):51–57, 1996.

U. M. Fayyad. *Advances in knowledge discovery and data mining*. AAAI: MIT Press, Menlo Park, Calif., 1996.

D. Feng, Z. Shuai, W. Heng, S. Huang, and T. Jun. Cooperative agent-based QoS framework for heterogeneous networks. In *Proceedings of Computer Engineering and Management Sciences (ICM) 2011, International Conference on Information Technology*, volume 2, pages 214–217. IEEE, 2011.

F. D. Foresee and M. T. Hagan. Gauss-Newton approximation to Bayesian regularization. In *Proceedings of the 1997 International Joint Conference on Neural Networks*, volume 3, pages 1930–1936. Houston, Texas, USA, 1997.

I. Foster. Efficient computation control in concurrent logic languages. *New generation computing*, 10(1):1–21, 1991.

T.C. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

K. Fujita. Extending OPNET modeler with client profiles for selecting data sources in WAN. *Nebula Project: final report*, pages 1–19, 2003.

R. S. Gray, D. Kotz, G. Cybenko, and D. Rus. Mobile agents: Motivations and state-of-the-art systems. In Bradshaw, editor, *Handbook of Agent Technology*. AAAI/MIT Press, 2000.

L. Green. *Automated, Ubiquitous delivery of Generalised Services in an Open Market*. PhD thesis, University of Technology, Sydney, 2007.

R. A. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE*, volume 3, pages 1903–1908 vol.3, 1997.

S. R. Gulliver and G. Ghinea. The perceptual and attentive impact of delay and jitter in multimedia delivery. *IEEE Transactions on Broadcasting*, 53(2):449–458, 2007.

J. Han and M. Kamber. *Data mining : concepts and techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2006.

J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

E.J. Hannan and J. Rissanen. Recursive estimation of mixed autoregressive-moving average order. *Biometrika*, 69(1):81–94, 1982.

A. L. G. Hayzelden and J. Bigham. Agent technology in communications systems: an overview. *Knowl. Eng. Rev.*, 14(4):341–375, 1999.

O. Hermanns and M. Schuba. Performance investigations of the IP multicast architecture. *Computer Networks and ISDN Systems*, 28(4):429–439, 1996.

H. Homayounfard and P. J. Kennedy. HDAX: Historical symbolic modelling of delay time series in a communications network. In P. J. Kennedy, K. Ong, and P. Christen, editors, *AusDM'09*, volume 101 of *CRPIT*, pages 129–138, Melbourne, Australia, 2009.

H. Homayounfard, P. J. Kennedy, and R. Braun. NARGES: Prediction model for informed routing in a communications network. In J. Pei et al., editor, *LNAI*, volume 7818, pages 327–338. Springer Berlin Heidelberg, 2013.

P. L. Hsu and H. Robbins. Complete convergence and the law of large numbers. *Proceedings of the National Academy of Sciences of the United States of America*, 33(2): 25–31, 1947.

IST-INTERMON. Advanced architecture for INTER-domain quality of service MONitoring, modelling and visualisation, Last Modified 6 April 2012. URL `http://www.ist-intermon.org/`. European IST research projects — for more information visit: www.cordis.lu/ist/.

ITU-T. One-way transmission time. *ITU-T Recommendation*, G.114, 1993.

ITU-T. One-way transmission time. *ITU-T Recommendation*, G.114, 2003. URL `http://www.itu.int/en/ITU-T/`.

M. Jain and C. Dovrolis. End–to–end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. *ACM SIGCOMM Computer Communication Review*, 32(4):295–308, 2002.

R. Jain. A delay–based approach for congestion avoidance in interconnected heterogeneous computer networks. *SIGCOMM Comput. Commun. Rev.*, 19(5):56–71, 1989.

B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *Communications Magazine, IEEE*, 45(10):112–121, 2007.

W. Jiang and H. Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *NOSSDAV'2000*, June 2000.

R. Jin and G. Agrawal. An algorithm for in-core frequent itemset mining on streaming data. In *Fifth IEEE International Conference on Data Mining, ICDM 2005*, New Orleans, USA, 2005. IEEE.

R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28 (1):51–55, 2003.

D. Katz, K. Kompella, and D. Yeung. Traffic Engineering TE Extensions to OSPF Version 2. *RFC*, 3630, 2003.

E. Keogh, J. Lin, and A. Fu. Hot SAX: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 1–8, Houston, 2005.

A. N. Kolmogorov. On the representations of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Dokl. Akad. Nauk USSR*, 114(5):953–956, 1957.

K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Advances in spatial databases*, pages 47–66. Springer, 1995.

D. Kotz and R. S. Gray. Mobile agents and the future of the internet. *SIGOPS Oper. Syst. Rev.*, 33(3):7–13, 1999.

A. Küpper and A. S. Park. Stationary vs. mobile user agents in future mobile telecommunication networks. In K. Rothermel and F. Hohl, editors, *Mobile Agents*, LNCS 1477, pages 112–123. Springer Berlin / Heidelberg, Stuttgart, 1998.

V. Kurkova. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5 (3):501 – 506, 1992.

A. Kusiak. A data mining approach for generation of control signatures. *Journal of manufacturing science and engineering*, 124(4):923–926, 2002.

M. Last, Y. Klein, and A. Kandel. Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 31(1):160 –169, Feb 2001.

H. Y. K. Lau and S. O. Woo. An agent–based dynamic routing strategy for automated material handling systems. *International Journal of Computer Integrated Manufacturing*, 21(3):269–288, 2007.

C.H. Lin, D.Y. Chiu, Y.H. Wu, and A. L. P. Chen. Mining frequent itemsets from data streams with a time-sensitive sliding window. In *SIAM international conference on data mining, SDM05*, volume 119, pages 68–79, Newport Beach, 2005.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2 – 11. ACM, San Diego, California, 2003.

R. Lopez, E. Balsa-Canto, and E. Oate. Neural networks for variational problems in engineering. *International Journal for Numerical Methods in Engineering*, 75(11): 1341–1360, 2008.

D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):415–447, 1992.

E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Electrical Engineers, Proceedings of the Institution of*, 121(12):1585–1588, 1974.

G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 346–357, 2002.

A. Markopoulou, F. Tobagi, and M. Karam. Loss and delay measurements of internet backbones. *Computer Communications*, 29(10):1590–1604, 2006.

A.P. Markopoulou, F.A. Tobagi, and M.J. Karam. Assessment of VoIP quality over internet backbones. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 150–159, 2002.

J. L. Marzo, E. Calle, C. Scoglio, and T. Anjah. QoS online routing and MPLS multilevel protection: a survey. *Communications Magazine, IEEE*, 41(10):126–132, 2003.

F. Mateo, J. J. Carrasco, A. Sellami, M. Milln-Giraldo, M. Domnguez, and E. Soria-Olivas. Machine learning methods to forecast temperature in buildings. *Expert Systems with Applications*, 40(4):1061 – 1068, 2013.

N. F. Maxemchuk and M. El Zarki. Routing and flow control in high-speed wide-area networks. *Proceedings of the IEEE*, 78(1):204–221, 1990.

A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. *Database Theory-ICDT 2005*, pages 398–412, 2005.

I. Miloucheva, U. Hofmann, and P. Gutirrez. Spatio-temporal QoS pattern analysis in large scale internet environment. In Giorgio Ventre and Roberto Canonico, editors, *Interactive Multimedia on Next Generation Networks*, volume 2899 of *LNCS*, pages 282–293. Springer Berlin / Heidelberg, 2003.

D.C. Montgomery, C.L. Jennings, and M. Kulahci. *Introduction to time series analysis and forecasting*. John Wiley, 2008.

S. B. Moon. *Measurement and analysis of end–to–end delay and loss in the Internet*. PhD thesis, University of Massachusetts, Amherst, 2000.

B. Moulin. The social dimension of interactions in multiagent systems. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications, LNCS*, volume 1441 of *Proceedings of the Workshops on Commonsense Reasoning, Intelligent Agents, and Distributed Artificial Intelligence*, pages 109 –123. Springer-Verlag, London, 1998.

R. S. Naoum and M. Maswady. Performance evaluation for VOIP over IP and MPLS. *Performance Evaluation*, 2(3):110–114, 2012.

S.J. Orfanidis. *Optimum signal processing: An introduction*. Macmillan New York, 1985.

V. Paxson. *Measurements and analysis of end–to–end Internet dynamics*. PhD thesis, University of California at Berkeley, Berkeley, CA, April 1997.

J. Postel. User datagram protocol. Technical report, IETF, 1980.

J. Postel. Transmission control protocol. Technical report, IETF, 1981.

J. Potemans, B.V. Broeck, G. Ye, J. Theunis, P. Leys, E.V. Lil, and A.V. Capelle. Implementation of an advanced traffic model in OPNET Modeler. *Department of Electrical Engineering – ESAT-TELEMIC Division, Belgium*, 2003.

J. Rankin, G. Christie, and Irina Kondratova. Mobile multimodal solutions for project closeout. In *Proceedings of the 6th Construction Specialty Conference, Canadian Society for Civil Engineering*, Toronto, Ontario, Canada, 2005.

M. Roccetti, V. Ghini, G. Pau, P. Salomoni, and M. E. Bonfigli. Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the internet. *Multimedia Tools and Applications*, 14(1):23–53, 2001.

L. E. Rocha-Mier, L. Sheremetov, and I. Batyrshin. Intelligent agents for real time data mining in telecommunications networks. In *Proceedings of the 2nd international conference on Autonomous intelligent systems: agents and data mining*, AIS-ADM'07, pages 138–152, Berlin, Heidelberg, 2007. Springer-Verlag.

M. Rose and D. Cass. *ISO Transport Service on top of the TCP Version: 3*, volume 1006 of *RFC*. RFC Editor, 1987.

L. Roychoudhuri and E. S. Al-Shaer. Real-time packet loss prediction based on end-to-end delay variation. *IEEE Transactions on Network and Service Management*, 2(1): 29–38, 2005.

H. Sandick and E. Crawley. QoS routing (qosr) working group report. Technical report, Internet Draft, Internet Engineering Task Force (IEFT), 1997.

R. Sasisekharan, V. Seshadri, and S. M. Weiss. Data mining and forecasting in large-scale telecommunication networks. *IEEE Expert: Intelligent Systems and Their Applications*, 11(1):37–43, 1996.

B. Schilling. Qualitative comparison of network simulation tools. *Institute of Parallel and Distributed Systems (IPVS), University of Stuttgart*, 2005.

R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169, 1997.

M. Schwartz and T. Stern. Routing techniques used in computer communication networks. *IEEE Transactions on Communications*, 28(4):539–552, 1980.

M. Shiblee, P. Kalra, and B. Chandra. Time Series Prediction with Multilayer Perceptron (MLP): A New Generalised Error Based Approach. *Advances in Neuro-Information Processing*, pages 37–44, 2009.

Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

J. L. Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet. *IEEE Transactions on Networking*, 10(4):541–550, 2002.

W. Stallings. *Data and computer communications*. Pearson Prentice-Hall, Pearson Education Inc., NJ, USA, 8th edition, 2007.

M. Steenstrup. *Routing in communications networks*. Prentice Hall, Englewood Cliffs, N.J., 1995.

P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *Communications Magazine, IEEE*, 35(1):80–86, 1997.

R.A. Tintin and D.I. Lee. Intelligent and mobile agents over legacy, present and future telecommunication networks. In A. Karmouch and R. Impey, editors, *MATA'99: First International Workshop on Mobile Agent for Telecommunication Applications*, pages 109–126. World Scientific Publishing, Ottawa, 1999.

Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda. Achieving moderate fairness for UDP flows by path–status classification. In *25th Annual IEEE Conference on Local Computer Networks*, pages 252–261, 2000.

V. Tresp and R. Hofmann. Nonlinear time-series prediction with missing and noisy data. *Neural computation*, 10(3):731–747, 1998.

J. J. Van Wijk and E. R. Van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of IEEE Symposium on Information Visualization (Info Vis '99)*, pages 4–9, Washington, DC, USA, 1999.

A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM2001)*, volume 9, pages 6–9, Prague, Czech Republic, June 2001.

Z. Wang and J. Crowcroft. A new congestion control scheme: slow start and search (Tri-S). *SIGCOMM Comput. Commun. Rev.*, 21(1):32–43, 1991.

G. Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 2000.

G. Weiss, J. Eddy, S. Weiss, and R. Dube. Intelligent telecommunication technologies. In L. C. Jain, R.P. Johnson, Y. Takefuji, and L.A. Zadeh, editors, *Knowledge–Based Intelligent Techniques in Industry*, pages 249–275. CRC Press, Boca Raton, 1998.

R. Wilcox. *Introduction to Robust Estimation and Hypothesis Testing*. Academic Press, Elsevier, Oxford, UK, 3rd edition, 2012.

Q. Wu and O. Mencer. Evaluating sampling based hotspot detection. In Mladen Berekovic, Christian Muller-Schloer, Christian Hochberger, and Stephan Wong, editors, *Architecture of Computing Systems - ARCS 2009*, volume 5455 of *Lecture Notes in Computer Science*, pages 28–39. Springer Berlin Heidelberg, 2009.

H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, and J.S. Yoo. A framework for discovering co-location patterns in data sets with extended spatial objects. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, volume 89. Florida, USA, 2004.

J.X. Yu, Z. Chong, H. Lu, and A. Zhou. False positive or false negative: mining frequent itemsets from high speed transactional data streams. In *Proceedings of the ThirtiethIinternational Conference on Very Large Databases*, volume 30, pages 204–215. VLDB Endowment, 2004.

L. A. Zadeh. From computing with numbers to computing with words from manipulation of measurements to manipulation of perceptions. *Annals of the New York Academy of Sciences*, 929(1):221–252, 2001.

LA Zadeh. From computing with numbers to computing with words: From manipulation of measurements to manipulation of perceptions. *IEEE transactions on circuits and systems. 1, Fundamental theory and applications*, 46(1):105–119, 1999.

X. Zhang, N. Mamoulis, D.W. Cheung, and Y. Shou. Fast mining of spatial collocations. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 384–393. ACM, 2004.

X. Zhang, Y. Pang, and Z. Guo. Rate-distortion based path selection for video streaming over wireless ad-hoc networks. In *2009 IEEE International Conference on Multimedia and Expo, ICME 2009*, pages 754–757. IEEE, 2009.

H. Zheng and J. Boyce. An improved UDP protocol for video transmission over internet-to-wireless networks. *IEEE Transactions on Multimedia*, 3(3):356–365, 2001.

B. Zhou, D. He, and Z. Sun. Traffic modeling and prediction using ARIMA/GARCH model. In A. Nejat Ince and E. Topuz, editors, *Modeling and Simulation Tools for Emerging Telecommunication Networks*, pages 101–121. Springer US, 2006.