Dynamic Routing Optimization Using Traffic Prediction

Dissertation

Zur

Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik

der Universität Rostock

vorgelegt von

Abutaleb Turky, geb. am 09.10.1978 in Kairo, Ägypten

aus Ilmenau

Rostock, 2014

urn:nbn:de:gbv:28-diss2015-0023-9

Betreuer:

- Prof. Dr. rer. nat. Clemens H. Cap
  Universität Rostock, Fakultät für Informatik und Elektrotechnik
  Institut für Informatik
  Lehrstuhl für Informations- und Kommunikationsdienste

Gutachter:

- Prof. Dr.-Ing. Dirk Timmermann
  Universität Rostock, Fakultät für Informatik und Elektrotechnik
  Institut für Angewandte Mikroelektronik und Datentechnik
  Lehrstuhl für Rechner in Technischen Systemen

- Prof. Dr.-Ing. Andreas Timm-Giel
  Technische Universität Hamburg-Harburg
  Institut für Kommunikationsnetze

Datum der Einreichung: 26. Mai 2014
Datum der Verteidigung: 17. November 2014

Zusammenfassung

Die Effizienz von Verfahren zum Traffic Engineering (TE) hängt maßgeblich von der Optimierung der Routen ab. Die meisten Routing-Algorithem nutzen Informationen über die verfügbare Bandbreite, um die günstigsten Pfade zwischen Sender und Empfänger zu bestimmen. Die Dienstgüte (Quality of Service, QoS) hängt zusätzlich von der Genauigkeit der Messungen der verfügbaren Brandbreite ab.

In aktuellen Verfahren zum dynamischen Routing wird der Zustand von Kanten im Netzwerk durch spezifische Gewichte repräsentiert. Diese Gewichte werden genutzt, um die günstigsten Routen zu berechnen. Die meisten Routing-Verfahren nutzen die aktuell gemessenen Bandbreiten, um die Gewichte der Kanten abzubilden. Auf Grund von häufigen Änderungen der verfügbaren Bandbreiten ist dies jedoch ineffizient für die Berechnung der Verbindungsauslastung. Zusätzlich basiert die Entscheidung des Routing-Verfahrens lediglich auf einer einmaligen Messung der Bandbreite und ist daher per se nicht akkurat.

Daher ist es eine sinnvolle Forschungsaktivität, die Verbindungsauslastung auf Basis des aktuellen Verkehrsaufkommen und der Nutzungsprofile für die nahe Zukunft abzuschätzen, um so die Leistung des Routing-Verfahrens zu verbessern. Es gibt nur wenige bekannte Forschungsanstrengungen in dieser Richtung, wie zum Beispiel den "Distribution-Free Prediction Interval (DF-PI) Algorithmus", der ein statistisches Modell der Verbindungsauslastung benutzt, oder den "Path Selection Algorithmus (PSA)", der Gleichungen in linearer Algebra nutzt, um die erwartete Verbindungsauslastung abzuschätzen.

In dieser Arbeit wird ein neuer, effizienter Routing-Mechanismus namens "Prediction of Future Loadbased Routing (PFLR)" vorgestellt, der die Leistung des Routing-Verfahrens verbessert. Der vorgeschlagene Ansatz funktioniert mit beliebigen Routing-Algorithmen, bei denen die Routen-Berechnung die Bandbreite berücksichtigt.

Mit Hilfe des PFLR-Algorithmus' wird die zukünftige Auslastung der Verbindungen im Netzwerk berücksichtigt. Diese Vorhersage trägt zu einer Reduzierung von Überlastungen im Netzwerk bei und ermöglicht insgesamt eine höhere Auslastung durch Nutzdaten.

Die Hauptidee des PFLR-Algorithmus' ist die gemeinsame Betrachtung der vorausgesagten und der aktuellen Verbindungsauslastung, um die Gewichte der Verbindungen im Netzwerkgraphen zu reduzieren. Der vorgeschlagene Ansatz

nutzt ein künstliches neuronales Netzwerke (Artificial Neural Network - ANN) als Prediktor für die zukünftige Verbindungsauslastung. Weiterhin hat der vorgeschlagene Algorithmus die Fähigkeit zur Adaption der Parameter im Vorhersagemodell, wie zum Beispiel "Gültigkeit der Vorhersage" und "Weite der Vorhersage". Dies dient zur effizienten Vorhersage des Verkehrsaufkommens und damit zur Verbesserung der Routing-Leistung.

Der Hauptgrund für die Nutzung von ANN ist die Tatsache, dass ANN eine der besten Möglichkeiten zum Modellieren und Vorhersagen der Verkehrsparameter sind. Ein ANN hat die Fähigkeit, verschiedene Funktionen zu approximieren, unabhängig von ihrem Grad der Nichtlinearität und ohne vorheriges Wissen der funktionalen Form. Daher bieten ANN eine akkurate Vorhersagemöglichkeit, für verschiedene Typen von Netzwerkverkehr.

Basierend auf verschiedenen Simulationsszenarios (mit verschiedenartigen Topologien, verschiedenen Typen von Netzwerkverkehr und unterschiedlichen Lastbedingungen) hat der entwickelte Routing-Algorithmus PFLR Vorzüge in Bezug auf Zurückweisung von Requests, Bandbreitenbegrenzung und Rerouting bei Link-Ausfällen.

Ein weiterer Forschungsgegenstand ist die Einführung eines neuen, effizienten TE-Algorithmus'. Dieser wird als Prediction-based Decentralized Routing (PDR) bezeichnet. Es handelt sich um einen vollständig dezentralen und selbstorganisierten Ansatz. PDR gehört zur Klasse der Ant Colony Routing (ACR)-Verfahren. Die Nutzung von Link-State-Informationen hilft dem Routing-Algorithmus bei der Erreichung einer Bandbreitengarantie. Die Betrachtung der zukünftigen, vorhergesagten Werte für die Link-Auslastung reduziert die Interferenz.

PDR-Algorithmen benutzen ähnliche Mechanismen wie der PFLR-Algorithmus, allerdings mit lokaler Implementierung. Weiterhin hat der PDR-Algorithmus die Fähigkeit zur lokalen Adaption der Gültigkeit der Vorhersagedauer abhängig von der Vorhersagegenauigkeit. Das dient der Vorhersage der Verbindungsauslastung.

Die Leistungsfähigkeit des vorgeschlagenen PDR-Algorithmus wird unter anderem mit verschiedenen zentral arbeitenden und dezentralen Routing-Verfahren und bei zwei verschiedenen Netzwerk-Topologien verglichen. Im Allgemeinen funktioniert der vorgeschlagene Algorithmus besser als die Vergleichsalgorithmen in Bezug auf verschiedene, zuvor genannte, Leistungsparameter.

Abstract

The efficiency of Traffic Engineering (TE) schemes mainly depends on routing optimization. Most routing algorithms use the information of available bandwidth (BW) to choose the paths between the source and destination nodes. Additionally, the provided Quality of Service (QoS) depends on the accurate measurement of the available BW.

In the current dynamic routing algorithms, the state of network links is represented by specific weights. These weights are used to compute the best paths between the source and destination pairs. Most routing approaches use the current measured BW information to represent the link weights. However, due to the varying nature of the available BW, updating the link state with the current measured BW is not an efficient approach to represent the link utilization. Also, the decisions of routing algorithm that depend on a single sample of measured available BW, which has not much significance due to the variable traffic nature, are not completely accurate.

Therefore, the new research direction is to perform the estimation of the link utilization in the future based on the actual traffic profile and use the estimated values of traffic to enhance the routing performance. There is a very small effort in this research direction, such as the Distribution-Free Prediction Interval (DF-PI) algorithm that uses the statistical model to estimate the link loads and the Path Selection Algorithm (PSA) algorithm that uses the linear algebra equations to estimate the link loads.

In this study, a new efficient routing maintenance approach, called Predicting of Future Load-based Routing (PFLR), is introduced for optimizing the routing performance in IP-based networks. The proposed approach runs with any routing algorithm whose computations depend on the residual BW in network links.

With the use of PFLR algorithm, the future status of the network link loads will be considered. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future and so reduces the occurrence of network congestions and at the same time leads to increase the network utilization.

The main idea of PFLR algorithm is combing the predicted link load with the current link load with an effective method in order to optimize the link weights and so reduce the occurrence of network congestions and increase the network utilization. The proposed approach uses the Artificial Neural Network (ANN) for building an adaptive traffic predictor in order to predict the future link loads. Furthermore, the proposed algorithm has the ability to adapt the parameters of the

prediction model, such as the length of prediction step and the prediction validity period, in order to efficiently estimate the link traffics and so effectively enhance the dynamic routing performance.

The main reason of using the ANN is that: the ANN is one of the best proposed tools for modeling and predicting the traffic parameters. The ANN has the ability to approximate too many functions regardless of their degree of nonlinearity and without prior knowledge of its functional form. Therefore, ANN can offer an accurate prediction capability (especially in our on-line forecasting case) with different types of network traffic whose nature is nonlinear and has the ability to be adaptive.

Based on different simulation scenarios (that have various network topologies, various traffic types and different network load conditions), the bundled routing algorithms with PFLR algorithm reduces the rejection ratio of requests, minimizes the bandwidth blocking rate and reroutes the requests upon link failure in an optimal way.

Another research objective is introducing a new efficient TE algorithm, called Prediction-based Decentralized Routing (PDR) algorithm, which is fully decentralized and self-organized approach. PDR algorithm is a new member of Ant Colony Routing (ACR) class. In this approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters. The use of link state information helps the routing algorithm to competently achieve the BW guarantee of the provided QoS. Moreover, the considering of future value of the network link loads leads to decrease the interference between the reserved requests in the future and so reduce the occurrence of network congestions and increases the network utilization.

PDR algorithm uses a similar prediction mechanism to the PFLR algorithm but with local-based implementation. Also, the PDR algorithm has the ability to locally adapt the prediction validity period depending on the prediction accuracy in order to make the prediction of link traffics more efficient and so effectively enhance the routing performance.

The performance of our proposed PDR algorithm is compared with various centralized and decentralized routing algorithms, within two different networks topologies, with different traffic types and under different network load conditions. In general, the proposed algorithm performs considerably better than the comparative algorithms with respect to different performance comparison criteria, such as the rejection ratio of requests, the bandwidth blocking rate.

Publications

- **2013**
  - Turky, A. A. & Cap, C. H. An effective QoS Providing Mechanism Using Prediction-based Decentralized Routing. IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA), 1089-1096, 2013.
- **2012**
  - Turky, A. A. & Cap, C. H. An Efficient Routing Maintenance Mechanism Using Adaptive Traffic Prediction. IEEE, IET Int. Symposium on Communication Systems, Networks and Digital Signal Processing (IEEE CSNDSP), 1-6, 2012.
- **2010**
  - Turky, A. A.; Liers, F. & Mitschele-Thiel, A. Self-optimizing Mechanism for Prediction-based Decentralized Routing. 7th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine) 2010, LNICST 74, 454-468, 2011.
- **2009**
  - Turky, A. A. & Mitschele-Thiel, A. Use of Load Prediction Mechanism for Dynamic Routing Optimization. IEEE Symposium on Computers and Communications ISCC, 782-786, 2009.
  - Turky, A. A. & Mitschele-Thiel, A. Prediction-based Decentralized Routing Algorithm. Electronic Communications of the EASST, 17, 2009.
- **2008**
  - Turky, A. A. & Mitschele-Thiel, A. MPLS Online Routing Optimization Using Prediction. Network Control and Optimization, Lecture Notes in Computer Science 5425, Springer, 45-52, 2008.

Acknowledgements

This thesis would not have been possible without the help and support of many people. Firstly, I would like to thank Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel for his knowledge, his belief and interest in the work. I have started my thesis and finished much work within his group.

I would like to thank my supervisor Prof. Dr. rer. nat. Clemens H. Cap for his excellent supervision , his encouragement and motivation throughout. Prof. Dr. Clemens H. Cap has played an excellent role for helping me to polish and finish up the work within my dissertation.

I would also like to thank all professors, college, and friends who encouraged me to accomplish this thesis. My thanks also go to everyone who has provided support or advice in one way or another. Last but not least, I would like to express my sincere, thanks and deepest gratitude to my family for their supports and encouragement.

Table of contents

List of figures

List of tables

CHAPTER 1 - Introduction

1.1 Study background

The rapid growth of Internet makes the Internet Service Providers (ISPs) demanding for a new technology which have the capability to maximize the network utilization. They hope to increase their gains by deploying the concept of service differentiation and offering the higher quality service at a premium. To support such capabilities, the conventional IP technologies should use the TE.

TE is defined as "the aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks" [1]. TE aims to cover different optimization issues that are related to the network performance such as providing the requested Quality of Service (QoS), minimizing the total delay and maximizing the network throughput, improving the network resources utilization by optimally distributing the traffic over the network topology and handling the quick recovery in cases of failure.

TE is an essential part in the network architecture for providing end-to-end QoS guarantees. QoS has many definitions. For example, in [2], QoS "is a set of service requirements to be met by the network while transporting a flow". There are many factors influencing the QoS which is expected by customers. Parts of these factors are objective metrics while others are subjective criteria.

The objective metrics are quantifiable factors such as the cost, the service reliability and the service level that meets the required throughput, loss and delay of the user applications. The subjective values are based on the opinions of the end-users, because users differ in their perception of what is good quality and what is not.

TE is an efficient management technique to optimize the utilization of network resources. With the help of TE, the network administrators have a precise control over the traffic within the network topology. They have the ability to make the balance of traffic loads among the network links and reduce the network congestions. Using TE technique, different categorized services can be offered by ISPs to their customers. ISPs can provide faster and more reliable service guarantees to their customers depending on the customer requirements and who

are able to pay more. The efficiency of TE schemes mainly depends on routing optimization, so the routing topic is discussed in more details in the next sections.

1.1.1 Routing in the Internet

Network routing refers to the ability of an electronic communication network to send a unit of information from point A to point B by determining a path through the network as efficiently and quickly [3] as possible. The sent data units of information are called packets. Such packets perhaps visit many cross-points. Cross-points are named as routers.

One of the router functions is to read the destination address of incoming packets, afterwards check an internal table for the best fitting outgoing link that the packets take towards its destination and then send the packets. Every link in the network has a specific limit of information which describes the maximum amount of bits which can be transferred per time unit, commonly named as link capacity.

The goals of a routing algorithm are determined by many factors such as the requirement of the communication network and the required traffic service. While the routing goals depend on the type of communication networks, it is classified into two different categories: user-oriented and network-oriented.

In the user-oriented type, the routing algorithm offers a good service to a specific user. However, the network-oriented routing algorithm offers an efficient and fair routing for most users, instead of providing the best service to a specific user.

The main routing functions are as follows [4]:

- Collecting the necessary information that is used to select and generate the paths and distributing it. This information includes service requirements and available resources within the network.
- Using the distributed information, the routing algorithm will optimally select and generate the paths according to a specific performance objective.
- Updating the router tables with the necessary information in order to forward the traffic along the selected paths.
- Forwarding the user traffic along the selected paths.

In addition, if any link or node failure occurs, the routing algorithm has to detect this exception and determine an alternate path as fast as possible.

1.1.2 Quality of service provision

Using the traditional best-effort service, there is no guarantee for the resulting transmission rate and the end-to-end delay. The processing of packets in this type of service provision is always the same within the network. Any network congestion causes an increasing end-to-end delay and leads to poor network performance.

The alternative mechanism is service provision with quality guarantees (QoS). The requested QoS depends on the application type [5]. Some applications demand for a guarantee of minimal useable bandwidth, another application needs a limit to the maximum end-to-end delay. The details of different QoS guarantees are discussed in the rest of this section.

- Delay guarantees: There is a class of applications which demands for a limitation of the end-to-end delay and the delay variance, like the voice over IP services and video conference systems. The end-to-end delay consists of three components. First is the propagation delay which depends on the distance between the source and destination pairs.

  Second is the queuing delay which represents the sum of waiting time at every intermediate node queue in the path. Third is the transmission delay which is determined by the minimum link capacity on the path.
- BW guarantees: Most multimedia applications require a specific level of BW guarantee. For this type of service, the routing algorithm reserves BW in all links of the path, which is equal to or higher than the traffic demand.

1.1.3 Routing classifications

The classification of routing algorithms depends on different point of views [6]. Here, a set of these classifications which are relevant for this work are presented:

- Static vs. dynamic: Within the static routing, the network administrators compute off-line all possible routes using static information and update manually the routing tables. In the dynamic routing approach, protocol uses the current state of network topology to compute the requested route

on demand. All routing decisions in the dynamic approach are done during runtime to adapt changes of network states. Most routing protocols are dynamic ones, like the Routing Information Protocol (RIP) [7] and the Enhanced Interior Gateway Routing Protocol (EIGRP) [8].

- Single-path vs. alternate and multi-path: Single-path routing algorithms determine the best available path according to the routing optimization goals and generate only one primary path between the source and destination pairs.

  Alternate path algorithms calculate in addition to the primary path another backup path to be used in case of any failure scenarios. The traditional routing example for this type is the Alternative Path Routing (APR) algorithm [9]. In the multi-path algorithms such as the Equal-Cost Multi-Path (ECMP) algorithm [10], there are multiple paths which are computed and maintained. In this case, the routing uses multiple paths to send the flows between the same source and destination nodes.
- Flat vs. hierarchical organization: Flat routing interprets every network node as separated peer and there exists one entry within the routing table for each node. In the routing algorithms which are based on the hierarchical organization such as the Fisheye State Routing (FSR) [11], all routers in the system are arranged in a hierarchical manner, where each hierarchy level is responsible for its own routing.
- Centralized vs. decentralized: In the centralized routing approach such as the Dijkstra's algorithm [27], each node has the complete link state information of a network's topology. All routing decisions are computed and taken in one place.

  On the other hand, the routing decisions in the decentralized approach are taken in all intermediate nodes between the source and destination pairs. All decisions are done, based on local state information only and there is no need for global information. The Optimal Routing Soft Computing Agents (ORSCA) algorithm [13] is an example of such decentralized routing algorithms.

- Constructive vs. destructive routing table making: In the constructive approach, the routing table is empty in the beginning. Afterwards, the routing algorithm adds routes one by one until the entities of routing tables are filled. The constructive routing is the common approach when the network topology is highly dynamic, like Mobile Ad hoc Networks (MANETs) [14].

  On the other hand, destructive algorithms suppose existing of all possible paths in the first. In other words, it assumes the network topology is a fully connected graph. Then, the routing algorithm collects some information to be able to delete the nonexistent paths in the physical network topology. The routing algorithm that uses random strategies to strongly explore the network topology is a destructive routing approach such as the reinforcement learning based routing algorithms [15].
- Proactive vs. reactive behavior: In the proactive routing approach such as the Optimized Link State Routing Protocol (OLSR) [16], the routes to all destinations are computed regardless of the route requests. While the reactive routing, like the Robust Secure Routing Protocol (RSRP) [17], searches for a route only as reaction to new routes or failure scenarios.

1.1.4 Dynamic routing protocol

A routing protocol is a set of processes, algorithms, and messages that are used to exchange routing information and populate the routing table with the routing protocol's choice of best paths [18]. The dynamic routing has not a fixed view of routing tables. Using the dynamic routing protocol, the routers are able to detect updates in the network topology. Each router communicates with its neighbors and sends the updates of the network environment. However, in the dynamic routing protocols, a lot of router resources are used and the overhead is increased. Additionally, it should reduce the complexity of update mechanism. The dynamic routing protocol can be classified according to different features. Two different classifications are discussed only, see Figure 1.1:

- Interior vs. Exterior Gateway Routing Protocols: The first classification depends on which level of network hierarchy that the protocol is to be

used. Part of routing protocols is used within a specific area, while the others are used among different areas.

- Interior Gateway Protocols (IGPs): IGPs are used within a single organization or domain such as the RIP Protocol [7] which uses the hop count information to select the best path and the Open Shortest Path First (OSPF) protocol [12] which uses the BW metric to select the shortest path. Another example is the Intermediate System–to–Intermediate System (IS-IS) protocol [19].
- Exterior Gateway Protocols (EGPs): In contrast to the IGPs, EGPs are used between the different organizations or domains such as the Border Gateway Protocol (BGP) [20] which is the most used exterior protocol between networks in the Internet.

- Distance vector vs. Link state vs. Path vectors protocols: The second classification depends on which information is distributed between the routers and how this information is used to update the routing tables.
  - Distance vector routing protocols: In distance vector protocols, like the RIP and the Interior Gateway Routing Protocol (IGRP) [21], each router distributes every period vectors of information to its neighbors. These vectors contain pairs of information about how to reach the destinations.

    First, the distance for a destination such as the hop count. The second is the direction for a destination or the next hop that should be used to reach a destination. Afterwards, each neighbor router includes its announcements and sends them again to its neighbors. Therefore, the routing table contains a cumulative distance to each destination in the network topology. This routing protocol uses the Bellman-Ford algorithm for selecting the best paths.
  - Link state routing protocols: In link state routing protocols, like the OSPF and IS-IS protocol, each router has a complete overview over the network topology. This includes a list of network links and their states. With the help of broadcast, this information is distributed among the routers in the network. The complete view of network link

states can be constructed in each router. In this way, each router can select the paths to all its destinations.

- Path vector protocols: In contrast to the previous two protocol classes, this protocol is used in the inter-domain such as the BGP [20]. The protocol behaves like a distance vector protocol but with advertising different kind of information. The advertised information contains the list of destination addresses and paths attributes to related destinations.

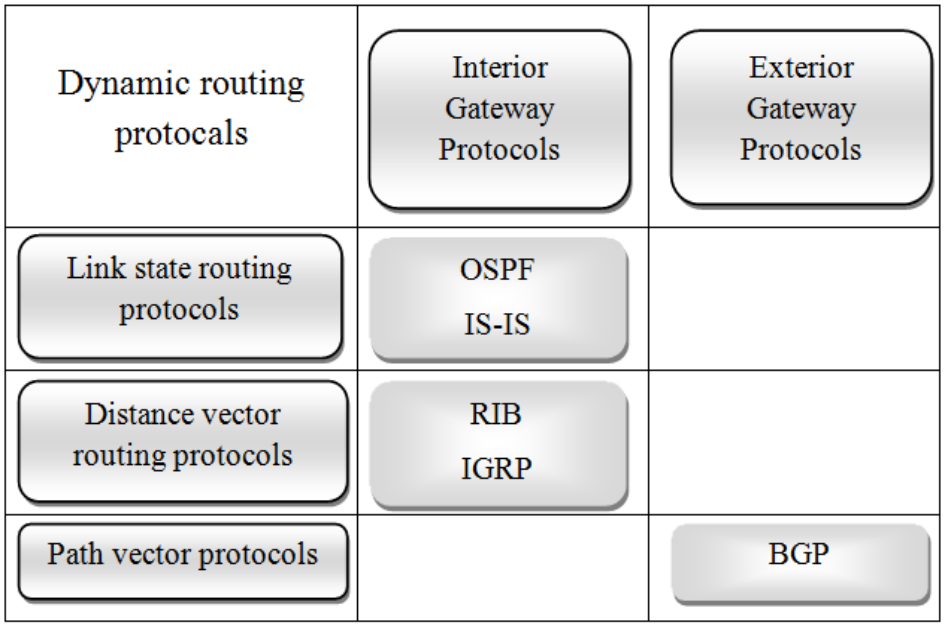

Figure 1.1 Dynamic routing protocol classifications.

1.2 Problem statement and motivation

During the current section, the statement of dynamic routing problem is described in more details in the first sub section. Additionally, the brief description of the main idea for the common solution for this problem is presented. After clarifying the statement of dynamic routing problem, the main motivation of the dissertation work is discussed in more details.

1.2.1 Problem statement

One of the main routing functions [4], which are described in section 1.1.1, is to optimally select the best paths between the various sources and destination pairs according to specific performance objective. The routing algorithm uses the current information of available resources within the network to select the routes.

Basically, the problem of finding the best paths between the source and destination pair is called the shortest path problem, which is defined in [28] as the following statements:

Given a weighted graph $G = (V, E)$, where V is the nodes set in the network and E is the network links set. Each link $l\ (u, v) \in E$ between nodes u and $v\ (u, v \in V)$ is specified by a single weight or cost $w\ (u, v) \geq 0$. The link weight can represent any measured value like the distance, available BW or delay.

The weight of a path $p = \{v_0, v_1, v_2, \ldots, v_k\}$ is the sum of link weights through this path:

$$w(p) = \sum_{i=0}^{k} w(v_{i-1} - v_i)$$

The object is finding a shortest path p^* between a given source node $s \in V$ and a given destination node $v \in V$.

$$p^* = \{min(\, w(p) : \ s \sim v\}$$

Figure 1.2 shows an illustrative example for a network weighted graph. In the example, $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $E = \{l_{12}, l_{13}, l_{18}, l_{23}, l_{27}, l_{28}, l_{34}, l_{35}, l_{46}, l_{56}, l_{57}, l_{67}, l_{78}\}$. There is two attached values for every network line. The first attached value (in the left side of network link) is the link weight, which represent any measured value like the distance, available BW or delay. The second attached value (the underlined value in the right side of network link) is the available BW (capacity units).

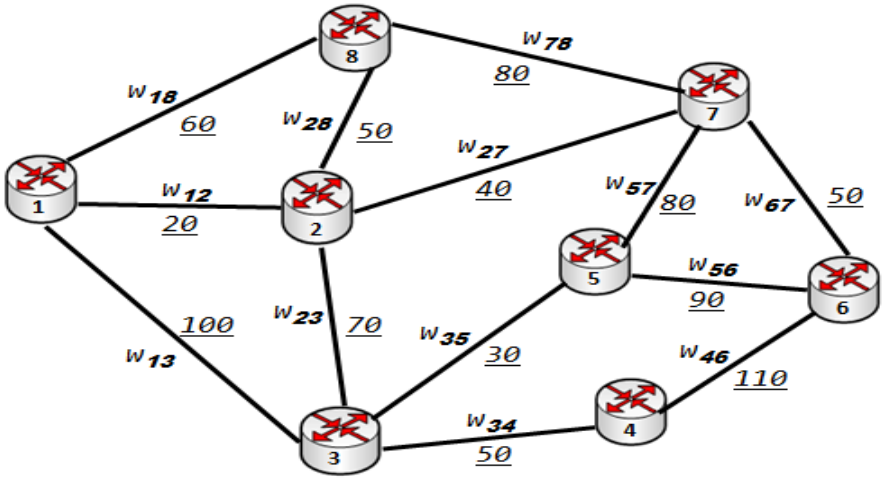

Figure 1.2 Illustrated example for a network weighted graph.

Every routing algorithm has its own method to represent the network link weights (or states). However, there are two common steps between most dynamic routing algorithms. These common steps are:

- Considering only the links that have available BW more than or equals the requested BW of requested flow. For example, Consider there is a route request between nods 1 and 6 that requires traffic demand equals 50 capacity units. In this case, all network links that have available BW less than 50 capacity units will be burnt and will not be considered. This means the network links l_{12}, l_{27} and l_{35} will not be considered (See Figure 1.3).
- Using Dijkstra's or Shortest Path First (SPF) algorithm [29], within reduced network, the shortest path is selected.

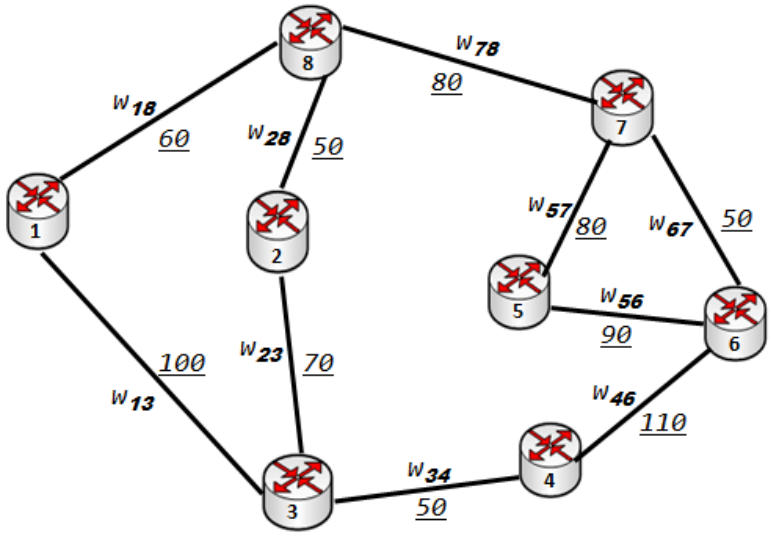

Figure 1.3 Reduced network after burning the unsuitable links.

1.2.2 Work motivation

According to the illustrated example in the last section, the routing algorithm has various alternative paths between the source and destination pair. While the routing algorithm selects the shortest path between all alternative paths. The desired method for determining the link weight values is an important factor that makes the routes selection differs from a routing algorithm to other routing algorithms.

The current dynamic routing algorithms update the link weights (states) with the current available BW. However, due to the varying nature of the available BW, updating the link state with the current measured BW is not an efficient approach to represent the link utilization. Also, the decisions of routing algorithm that depend on a single sample of measured available BW, which has not much significance due to the variable traffic nature, are not completely accurate.

Therefore, the new research direction is to perform the estimation of the link utilization in the future based on the actual traffic profile and incorporate the estimated values in the network link weights to enhance the routing performance. Of course, when the values of link weights will be changed, the shortest path between all alternative paths will be changed also. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future and so reduces the occurrence of network congestions and at the same time leads to increase the network utilization.
There is a very small effort in this research direction, such as the Distribution-Free Prediction Interval (DF-PI) [46] algorithm that uses the statistical model to estimate the link loads and the Path Selection Algorithm (PSA) [49] algorithm that uses the linear algebra equations to estimate the link loads. In the next section, our contribution in this research direction is discussed in more details.

1.3 Dissertation contributions

During this dissertation, an adaptive optimization mechanism is introduced which is based on a traffic prediction approach in order to improve the performance of dynamic routing algorithms. The work objective is reducing the network congestion, increasing the network utilization and efficiently re-routing the path requests upon link failure scenarios. During this dissertation, two different contributions are presented. The first contribution is introducing a new TE maintenance algorithm for centralized routing algorithms in order to enhance the performance of routing algorithms. The second contribution is developing a new TE prediction-based routing algorithm. The second approach is fully decentralized and self-organized algorithm.

1.3.1 Enhancing the performance of centralized routing algorithms

The first contribution is introducing a new TE routing maintenance algorithm for centralized routing algorithms in order to optimize the performance of routing algorithms. A new routing maintenance algorithm is proposed, called Predicting of Future Load-based Routing (PFLR) algorithm [22], [23], [24].
The PFLR algorithm runs beside any routing algorithm that depends on the available BW information of network links in order to select the best paths

between the source and destination pairs. The PFLR algorithm aims to efficiently represent the network link weights (states).

With the use of PFLR algorithm, the future status of the network link loads will be considered. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future and so reduces the occurrence of network congestions and at the same time leads to increase the network utilization. The most important feature of PFLR algorithm is the link weighs (states) representation. The proposed algorithm combines the predicted link load with the current link load with an effective method in order to optimize the link weights. The idea is to reduce the number of wrong and critical decisions in case of uncertain prediction accuracy.

In contrast to the DF-PI and PSA algorithms, the proposed approach uses the Artificial Neural Network (ANN) for building an adaptive traffic predictor in order to predict future link loads. The main reason of using the ANN is that: the ANN is one of the best proposed tools for modeling and predicting the traffic parameters. The ANN has the ability to approximate too many functions regardless of their degree of nonlinearity and without prior knowledge of its functional form. Therefore, ANN can offer an accurate prediction capability (especially in our on-line forecasting case) with different types of network traffic whose nature is nonlinear and has the ability to be adaptive. The up-to-date articles that propose and demonstrate the use of ANN for building the traffic predictor are published in [95], [96], [97], [98], [99] and [100].

Additionally, the advanced version of PFLR algorithm (PFLRv.2) has the ability to adapt the parameters of prediction model, such as the length of prediction step and the prediction validity period, in order to efficiently estimate the link traffics and so enhance the routing performance. The parameters adaptation process depends on the measurement of prediction accuracy in order to adapt the parameters.

Based on different simulation scenarios (that have various network topologies, various traffic types and different network load conditions), the bundled routing algorithms with PFLR algorithm reduces the rejection ratio of requests (In the best case, it rejects 17.45% less requests than the normal algorithms), minimizes the

bandwidth blocking rate (In the best case, it rejects 17.63% less BW than the normal algorithms) and reroutes the requests upon link failure in an optimal way.
Additionally, a comparative study between the PFLRv.2 algorithm and different estimation-based routing algorithms is presented. The objective is to prove the efficiency of PFRLv.2algorithm, based on some test scenarios and discuss the results. The experiment results show that, the proposed mechanism of PFLRv.2 algorithm enhances the performance of routing algorithms much more than statistical and linear prediction equations approaches.

1.3.2 Developing a new prediction-based decentralized routing algorithm

According to the great benefits of using the self-organizing systems, the second contribution for this dissertation is developing a fully decentralized and self-organized algorithm. A new efficient TE algorithm is proposed, called Prediction-based Decentralized Routing (PDR) [25], [26], [27].
The PDR algorithm is a member of traffic-aware routing algorithms. At the same time, this algorithm is considered as a new member of Ant Colony Routing (ACR) class. The ACR algorithms are inspired from real ants' behaviors which have the ability of discovering the shortest path to a food source and their nest without any knowledge of geometry but with a keen sense of smell.
According of the new aspects within the PDR approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time, which is used within the traditional ACR algorithms, to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters. Using the information of link state helps the routing algorithm to efficiently achieve the BW guarantee of the provided QoS. Additionally, considering the future value of the network link loads leads to reduce the interference between the reserved requests in the future and so reduce the occurrence of network congestions and increases the network utilization.
One of the good features within the PDR algorithm is the use of advanced and effective Ant-based framework, which is tested and evaluated in [45], to build the PDR approach. In addition to that, the link state information is represented by an efficient formula which is used in an advanced routing algorithm, called Least Interference Optimization Algorithm (LIOA) [73]. LIOA algorithm proves its

efficiency when compared to other routing approaches. Additionally, the PDR algorithm uses a similar prediction mechanism to the PFLR algorithm but with local-based implementation. Finally, PDR algorithm has the ability to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics and so enhance the performance of PDR algorithm.

The advanced version of PDR algorithm (PDRv.2) algorithm uses an efficient ant's selection methods (for the intermediate nodes) which consider the predicted link load to better estimate for the congestion within network links. This feature gives the ability to efficiently distribute the ants on the network topology and accurately discover the best paths.

The performance of our proposed PDR algorithm is compared with various decentralized algorithms and within two different networks topologies and with different traffic types. In general, the proposed algorithm performs considerably better than the comparative algorithms with respect to different performance comparison criteria, such as the rejection ratio of requests (In the best case, it rejects 63.40% less requests than the comparative algorithm) and the bandwidth blocking rate (It rejects 62.37% less BW than the comparative algorithms).

Additionally, based on a comparative study between various versions of PDR algorithm and various centralized routing algorithms, the PDRv.2 algorithm (and In contrast to the PDRv.1 algorithm) performs considerably better than various centralized algorithms with respect to different performance comparison criteria, such as the rejection ratio of requests (In the best case, it rejects 32.89% less requests than the traditional centralized algorithm) and the bandwidth blocking rate (It rejects 35.86% less BW than the comparative algorithms).

1.4 Dissertation Structure

In the next chapter, a literature review for three different topics is introduced. The first part within the second chapter covers different centralized routing algorithms, the second part within the second chapter demonstrates different estimation-based routing algorithms and the third part within the second chapter represents various ACR algorithms.

Chapter three give in the first part an overview about the biological neuron model and the artificial neuron model. After that, the architecture of various ANN models is presented. The third part within chapter three demonstrates the details of different learning processes for the ANN. The forth part list the various applications of ANN in the different research area. The last part in this chapter discusses the traffic prediction capability using the ANN model. This part first defines the traffic prediction problem and presents the main characteristics of network traffic, which lead to the possibility of traffic prediction, are listed. Finally the survey of using the ANN within the traffic prediction field is presented. Additionally, the main features of ANN, that gives it the ability to be the best selected tool to predict the network traffic, are presented.

Chapter four represents in more details the design of proposed model for both PFLR and PDR routing algorithms. For every proposed algorithm, the characteristics of innovative idea, the model architecture, the details of internal predictor processes and the pseudo code of algorithm are presented. Additionally, the complexity analysis for each algorithm is discussed in more details in the last section of their related algorithm.

Chapter five presents the performance study of all proposed algorithms compared to the traditional and advanced routing algorithms and with respect to different performance criteria. The simulation sceneries consider four different network topologies, two different generated traffic types, various network load conditions and two different real traffic data sets. Additionally, a comparative study of PFLR algorithm and different estimation-based routing algorithms is introduced. Finally, Chapter six presents the final conclusion for this dissertation and shows the future work.

Appendix A presents the analysis studies of PFLR and PDR algorithms that are performed in order to select the best values of parameters. Appendix B presents analysis studies for the PFLRv.2 and PDR algorithms in order to test the prediction accuracy of proposed algorithms for different load scenarios and with respect to different routing algorithms. Finally, Appendix C presents the stabilization of statistic results.

CHAPTER 2 - Routing literature review

This chapter includes a literature review for different classes of routing algorithms. The first part presents related work in centralized routing approaches. The second part gives an overview of the estimation-based routing approaches. The last part focuses on the decentralized routing approach and demonstrates the basic concepts of ACR algorithms.

2.1 Centralized routing algorithms

In centralized routing algorithms, as described before in chapter one, the source node has all current state of network links and the route selection is done in one place. In this part, some conventional routing algorithms that don't consider the future route requests are presented such as Dijkstra's [29], Widest Shortest Path (WSP) [34] and Constraint Shortest Path First (CSPF) [35] algorithms. Additionally, some advanced routing algorithms are presented, which consider the future route requests and plan to reduce or avoid the interference that may happen in the future between the route requests, such as Minimum Interference Routing Algorithm (MIRA) [36], Dynamic Online Routing Algorithm (DPRA) [43] and Least Interference Optimization Algorithm (LIOA) [45].

2.1.1 Shortest path routing algorithms

There are two particular shortest path routing algorithms, Dijkstra and Bellman–Ford algorithms [32]. Both of them target to find the shortest path between the source and destination pairs.

2.1.1.1 Dijkstra's algorithm

Dijkstra's or Shortest Path First (SPF) algorithm is proposed by E. Dijkstra [29] to give a solution for the shortest path problem. The basic idea of SPF algorithm is used in many routing protocols such as OSPF and IS-IS.

Dijkstra's algorithm depends on the fact that states, the subsections of shortest paths are also shortest paths. Also it does not compute the shortest path to a specific destination only, but it computes the shortest paths to all possible destinations in the network. Dijkstra's algorithm divides all nodes into two

groups: group *A* that contains all visited nodes during the algorithm to search for the shortest path from the source node *s* to destination node *v*, group *B* that contains the remaining nodes.

Algorithm 2.1 Dijkstra's algorithm.

1) Put the source node s in the set *A*, put all other nodes in the set *B*.
2) Assign *Dist* variable to every node which represent the minimum distance to the source node. The source node has zero value, the directly connected nodes have the distance value between them and the source node and the others have infinity.
3) Do the following steps:
   a) Select the node *u* that is not in the current set *A* and has the minimum *Dist* value.
   b) Add node *u* to set *A* and remove it from set *B*.
   c) Stop the algorithm if set *B* is empty.
4) Do the following steps:
   a) Indentify all neighbor nodes for node *u* that are not in the current set *A*.
   b) Check the improvement of *Dist* for every neighbor k, *Dist (k)* is equal to *min (Dist (k), Dist (u) + w (u, k))*.
5) Go to step 3.

Dijkstra's algorithm starts with initialization of the *Dist* variable to each node which represents the minimum distance to the source node. In the next step, the algorithm describes how to extend the set *A*. In each an iteration of the algorithm, it selects the unvisited node that has the minimum *Dist* value. Then, the algorithm tries to check if the *Dist* variable for every neighbor is reduced or not. If it is reduced then it updates the *Dist* variable with the new value, this step is called "relaxing" step. Finally, it returns to step number three until set *B* becomes empty. In the context of algorithms comparisons, it is helpful to be aware of the computational complexity for every algorithm. Any algorithm contains various

operations which should be considered when determining the computational complexity for this algorithm. One of the famous methods that represent the computational complexity is big-O notation which is defined as "a theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size" [30].

In order to determine the computational complexity of Dijkstra's algorithm, the worst case scenario should be considered. The worst case scenario is considering the complete graph of network topology. In this type of network topology, every node is connected to any node in the network topology. Assume now Dijkstra's algorithm is trying to find the shortest paths from the source node *s* to $|V|$ -1 destination nodes, where $|V|$ is the number of vertices in the graph. In other words, $|V|$ -1 comparison operations are made on $|V|$ -1 network node. This means that the complexity of Dijkstra's algorithm is $O(|V|^2)$. But using a good data structure [31], it can be improved to $O(|E| + |V| \log |V|)$, where $|E|$ is the number of links in the network topology.

2.1.1.2 Bellman–Ford algorithm

The Bellman–Ford algorithm was proposed by R. Bellman and L. Ford [32], [33].In contrast to Dijkstra's algorithm, this algorithm allows using of negative edge weights but it does not allow negative weigh cycle occurrence. The main core of Bellman–Ford is very similar to Dijkstra's algorithm, but instead of selecting only one node from the unvisited nodes that has the minimum distance to the source node *s*, it "relaxes" all the links and repeats this for $|V| - 1$ times. With these repetitions, the algorithm propagates the shortest distance to all destination nodes. In general, the Bellman–Ford algorithm is slower than Dijkstra's algorithm. The distributed version of Bellman–Ford algorithm is used in the RIP protocol.

The algorithm starts with the initialization of *Dist* variable to each node which represents the minimum distance to the source node*s*. It sets zero *Dist* value to the source node*s* and infinity to the others. The algorithm in the second step tries to "relax" every link in the network and updates the *Dist* variable for every node depending on the improvement of the *Dist* variable. It repeat the previous step $|V|$

− 1 times. In this way, the algorithm continues through the graph by first checking the 1-hop paths, then the 2-hop paths up to paths with $|V| - 1$ hops.

Algorithm 2.2 Bellman–Ford algorithm.

1) Assign *Dist* variable to every node which represent the minimum distance to the source node. The source node has zero value and the others have infinity.
2) Repeat the next step $|V|$-1 times.
   a) Repeat the next step for every link $l\,(u, v) \in E$.

      if $Dist\,(v) > Dist\,(u) + w(u, v)$ then

      $Dist\,(v) = Dist\,(u) + w(u, v)$

      end if
3) Repeat the next step for every link $l\,(u, v) \in E$.
   a) if $Dist\,(u) + w(u, v) < Dist\,(v)$ then

      Return error "Graph contains a negative-weight cycle"

      end if

As described before, the Bellman–Ford algorithm can be implemented with negative values for edge weights. In this case, the algorithm should have a validation step to verify that the final graph result does not contain any negative weight cycle. Thus, the algorithm in the final step checks if there is any negative weight cycle exist and returns an error message in this case. If the graph does not contain any negative weight, the final step will be useless and can be omitted.
The Bellman-Ford algorithm takes in the worst case $|V| - 1$ iterations. In each an iteration of the algorithm, $|E|$ comparison operations are required to relax the network links. Therefore, the computational complexity of Bellman-Ford algorithm is $O\,(|V||E|)$.

2.1.2 Widest shortest path algorithm

Roch A. Guerin [34] has introduced a modification to the shortest path algorithm, called Widest Shortest Path (WSP), which is based on the computation of the shortest paths in the first stage and if there is more than one of these shortest

paths, it chooses the path whose available BW (i.e., the smallest value on any of the links in the path) is maximal. This work is introduced to provide extensions to the OSPF protocol in order to support QoS routing in IP-based networks.

WSP algorithm aims to select a path that satisfies the required BW of flow and minimize the use of network resources. WSP has different implementations based on either pre-computations or on-demand computations. The associated link cost or weight is a combination of hop count information and available BW in this link. WSP algorithm searches for a path with the minimum number of hops which can support the requested BW. If there is more than one, it prefers the path with maximum BW. The focus here is on the implementation of WSP algorithm for on-demand computation of QoS paths which depends on the modification of Dijkstra's algorithm.

Algorithm 2.3 WSP algorithm.

1) Consider only the links that have available BW more than the requested BW of flow.
2) Give every link l $(u, v) \in E$ equal weights.
3) Put the source node s in the set A, put all other nodes in the set B.
4) Assign *Dist* variable to every node which represent the minimum distance to the source node. The source node has zero value, the directly connected nodes have an equal value and the others have infinity.
5) Do the following steps:
   a) Select the node u that is not in the current set A and has the minimum *Dist* value.
   b) If there are more than nodes have the same minimum *Dist* value, select the node that belongs to a path that contains maximum BW.
   c) Add node u to set A and remove it from set B.
   d) Stop the algorithm if set B is empty.
6) Do the following steps:

a) Indentify all neighbor nodes for node u that are not in the current set *A*.

b) Check the improvement of *Dist* for every neighbor k, *Dist (k)* is equal to *min (Dist (k), Dist (u) + w (u, k))*.

7) Go to step 5.

WSP starts with neglecting the links that have not sufficient BW for the requested BW of flow. Another modification to Dijkstra's algorithm is to assume all network links have an equal weight or cost. In this way, the shortest path contains the minimums numbers of hops. Additionally, Dijkstra's algorithm is modified to keep only the maximum BW path among the equal hop paths. So, WSP algorithm selects the node that belongs to a path that contains maximum BW in case of existing of more than node with the same minimum *Dist* value. The computational complexity of WSP algorithm is similar to Dijkstra's algorithm, it is $O(|V|^2)$ but using a good data structure, it can be improved to $O(|E| + |V| \log |V|$

2.1.3 Constraint shortest path first algorithm

E. Crawley [35] proposed the Constraint Shortest Path First (CSPF) algorithm to overcome the problem of load balancing in OSPF which modifies the link cost to reflect the current resource availability. The cost of links is inversely proportional to the residual link capacities. The CSPF algorithm is introduced as an efficient component in the QoS-based routing framework in the Internet. The main objectives of QoS-based routing are:

- Dynamic selection of requested paths: QoS-based routing should provide the requested QoS guarantee such as the end to end delay and BW.
- Optimization of network resources: QoS-based routing target to improve the total network throughput in order to enhance the network resource utilization.
- Graceful performance degradation: Another objective is giving an efficient reaction in the case of overload conditions.

The link weights in QoS-based routing must reflect the QoS requirements. Of course, a combination of metrics can be considered, but it should be carefully

handled to reduce the complexity of computing paths. CSPF algorithm is used in the second version of OSPF protocol.

CSPF can be considered as an advanced version of shortest path algorithm. The algorithm starts with pruning all the links that do not have sufficient BW for the requested BW of flow. As mentioned above, the algorithm aims to reflect the current resource availability; therefore CSPF algorithm sets all link weights equal to the inverse of their available BW. The rest of the CSPF algorithm is the same as Dijkstra's algorithm.

Algorithm 2.4 CSPF algorithm.

1) Consider only the links that have available BW more than the requested BW of flow.
2) Set the link weights w (u, v) equal to the inverse of their available BW.
3) Put the source node s in the set A, put all other nodes in the set B.
4) Assign *Dist* variable to every node which represent the minimum distance to the source node. The source node has zero value, the directly connected nodes have the inverse value of available BW for their link and the others have infinity.
5) Do the following steps:
   a) Select the node u that is not in the set A and has the minimum *Dist* value.
   b) Add node u to set A and remove it from set B.
   c) Stop the algorithm if set B is empty.
6) Do the following steps:
   a) Indentify all neighbor nodes for node u that are not in the current set A.
   b) Check the improvement of *Dist* for every neighbor k, *Dist (k)* is equal to *min (Dist (k), Dist (u) + w (u, k))*.
7) Go to step 5.

2.1.4 Minimum interference routing algorithm

Minimum Interference Routing Algorithm (MIRA) is an example of advanced routing algorithms [36]. The idea of MIRA is avoiding the routing over links that may interfere with another path requests in the future. This algorithm is proposed to define dynamic Label Switched Paths (LSPs) in Multi-Protocol Label Switched (MPLS) networks. In MPLS networks [37], the packets are assigned with labels at the ingress router of the network. These labels are used to forward the packets according to specific LSPs. Service providers can use LSPs to implement Virtual Private Networks (VPNs) or to satisfy other QoS agreements [38].

Signaling protocols like Resource Reservation Protocol (RSVP-TE) [39] and Label Distribution Protocol (LDP) [40] are used to setup the paths. In other words, the route of LSPs is explicitly defined between the ingress and egress routers with the help of these signaling protocols. With the explicit LSPs routing in MPLS networks, the service providers have an important feature to engineer how their traffic will be routed, and have the ability to improve the network utilization, by minimizing the rejection ratio of requests.

The key idea of MIRA algorithm is selecting the paths that have a little chance to interfere with future paths which are among other source and destination pairs. The definition of interference in MIRA depends on computing the maximum flow (max-flow) value between a given ingress and egress pair. The max-flow $MV\ (s_1, v_1)$ value is defined as "An upper bound on the total amount of BW that can be routed between a given ingress and egress pair (s_1, v_1)" [41]. When the routing algorithm routes LSP with D units of BW between s_1 and v_1, the $MV\ (s_1, v_1)$ value will be decreased by D value. However, the value of $MV\ (s_1, v_1)$ may be also decreased when other LSPs are routed between other ingress and egress pairs.

So, the amount of interference for a given (s_1, v_1) pair is defined as the decrease of $MV\ (s_1,v_1)$ value because of routing LSPs between some other ingress and egress pairs .The links positions in the network topology have the great effect on reducing the max-flow values between different ingress and egress pairs. The algorithm defines the links that have a high priority to decrease the max-flow values of one or more ingress and egress pairs as critical links.

Therefore, the MIRA algorithm aims to pick up the minimum interference path for a given LSP that maximizes the minimum max-flow between all other source and destination pairs. It targets to reduce the routing over critical links. Thus, the algorithm represents the criticality properties by giving the critical links small weight values. The explicit route is commuted using a modified version of Dijkstra's shortest path algorithm.

The problem of maximizing the minimum max-flow (MAX-MIN-MAX) between all source and destination pairs can be formalized as: For a given a path between *a* and *b* nodes, the objective function of MAX-MIN-MAX is

$$\max \sum_{(s,v)\epsilon V \setminus (a,b)} \alpha_{sv}\, MV(s,v)$$

Since (α_{sv}) parameter is the weight of the ingress and egress pair which represents the relative importance of the ingress and egress pair to the network administrator. To compute the link weight *w (u, v)* for every link *l* (*u, v*) $\in E$, the set of critical links C_{sv} for the ingress and egress pair (*s, v*) is determined in the first, so

$$w\,(u,v) = \sum_{(s,v)\epsilon V:\, l(u,v)\epsilon C_{sv}} \alpha_{sv}$$

If all ingress and egress pairs have the same priorities, *w (u, v)* represents the number of ingress and egress pairs for which link is critical.

The flow residual graph theory [42] is used to determine the critical links set. After computing the max-flow between the node *s* and node *v*, let *R* be the set of nodes that are reachable from the source node *s* and *T* be the set of nodes that are reachable from the destination node *v*. A link *l (i, j)* $\in C_{sv}$ if:

- Link *l (i, j)* is filled to capacity.
- $j \notin R$ and $i \notin T$.
- There no path between *i* and *j* in the flow residual graph

MIRA algorithm in the first step computes the max-flow for all possible source and destination pairs excluding the *(a, b)* pair. Then it computes the set of critical links C_{sv}. Depending on the previous step, the algorithm computes the weights that will be associated to every network link. In the fourth step, all the links that do not have sufficient BW to route *D* units are pruned. The rest of MIRA algorithm is using Dijkstra's shortest to find the shortest path between *a, b* nodes.

Algorithm 2.5 MIRA algorithm.

Input:	A graph $G = (V, E)$ and a set B of all residual link capacities. An ingress node a and an egress node b between which a flow of D units have to be routed.
Output:	A route between a and b having a capacity of D units of BW.

Algorithm:

1) Compute the maximum flow values for all $(s, v) \in V \setminus (a, b)$.
2) Compute the set of critical links C_{sv} for all $(s, v) \in V \setminus (a, b)$.
3) Compute the weights

$$w(u, v) = \sum_{(s,v)\in V:\, l(u,v)\in C_{sv}} \alpha_{sv}$$

4) Eliminate all links which have residual BW less than D and form a reduced network.
5) Using Dijkstra's algorithm, compute the shortest path in reduced network using $w(u, v)$ as the weight on link $l(u, v)$.
6) Route the demand of D units from a to b along this shortest path and update the residual capacities.

In general, MIRA is proposed for routing of BW guaranteed LSPs in MPLS networks. Also, it has the capability to include the hop-count constraints by using the Bellman-Ford algorithm instead of using Dijkstra's algorithm. MIRA algorithm outperforms the WSP algorithm and (Minimum Hop Algorithm) MHA with respect to many performance metrics such as the rejection ratio of path requests and the successful re-routing of demands upon link failure scenario.

On the other hand, MIRA algorithm has a higher computational complexity than the shortest path algorithm. In the worst case, every node can be a source node for all other nodes. Therefore, there are $|V|^2$max-flow calculations are required. Since

each maximum flow calculation takes $O(|V||E| \log(|V|^2/|E|))$ [42], so the worst case runtime of MIRA is $O(|V|^3|E| \log(|V|^2/|E|))$.

2.1.5 Dynamic online routing algorithm

R. Boutaba [43] introduced the Dynamic Online Routing Algorithm (DORA) that selects the BW guaranteed LSPs in MPLS-based networks. DORA aims to avoid routing over links that have a high potential to be part of any other paths in the future. Also it targets to balance the traffic load by preferring the paths that have more residual BW. DORA depends on computing the Path Potential Value (*PPV*) array associated with a source-destination pair to minimize the expected interference between the different paths.

Both of MIRA, DORA and Profile-Based Routing (PBR) [44] algorithms have discussed the requirements and design issues for path selection algorithms within MPLS-based network. A brief description for these requirements is presented here:

- Routing constraints: the routing algorithm should compute the optimal paths according to different constraint types such as delay, jitter, loss ratio and BW.
- Online routing: Due to TE requirements, the routing algorithm should have the capability to compute the LSPs on demand. It has not any prior knowledge of the arrival time and required BW of LSPs.
- Computational complexity: In the case of online routing, the time of routing Computation should be short enough to offer fast LSPs selection in MPLS-based network.
- Re-routing performance: Another important routing objective is efficiently rerouting the requests upon node or link failure scenario. Also, it is important to spread the traffic onto the network in an optimal way to reduce the LSPs numbers that are affected by the link or node failures.
- Link state distribution: All link state information must be present at the ingress node to compute the best path. Therefore, a new distribution technique is required to distribute all required routing information such as the network topology and the residual BW of links.

- Use knowledge of the ingress and egress of LSPs: Sometime, it is useful to know all possible combination of the ingress and egress of LSPs. The routing algorithm may use this knowledge to reflect the interference effect on the link weights.

DORA algorithm has two stages. In the first stage, it computes the *PPV* for every source and destination pair. *PPV* represents how a specific link has the chance to be included in a path more than the other links. For every source and destination pair, the algorithm associates every link a *PPV* variable and initializes it by zero.

Algorithm 2.6 DORA algorithm.

Stage 1:

1) For each source–destination pair *(s, v)*, determine the set of all Disjointed Paths $DP_{(s,\ v)}$. One possible way is to use Dijsktra's algorithm to find a shortest path (in terms of number of hops) for *(s, v)*, add this path to $DP_{(s,\ v)}$, and then remove all links that are part of the resulting path, and repeat these steps until *v* is no longer reachable from *s*.
2) For each source–destination pair *(s, v)*, construct the $PPV_{(s,\ v)}$ array, and initialize all entries to zero. The array size equals the No. of network links.
3) For the source–destination pair *(s1, v1),*
   a) For each link *l* in the network, if *l* is part of any path in $DP_{(s1,v1)}$, subtract 1 from $PPV_{(s1,v1)}(l)$.
   b) For all the source–destination pairs other than *(s1, v1)*, inspect each link *l* and determine the number of times *n* that *l* appears in $DP_{(s,\ v)}$, where *(s, v)* is not equal to *(s1, v1)*. Increment $PPV_{(s1,\ v1)}(l)$ by *n*.
4) Repeat Step 3 for all the other source–destination pairs.
5) For each source–destination pair (s, v), normalize all entries in $PPV_{(s1,\ v1)}$, with the smallest *PPV* element over all source–destination pairs equal to 0 and the largest *PPV* element over all source–destination pairs equal to 100. Let $NPPV_{(s1,\ v1)}$ (l) be equal to the normalized value of $PPV_{(s1,\ v1)}$ (l).

Stage 2: (Suppose a request arrives to set up a path between *s1* and *v1* with *D* amount of BW)

1) Remove links with a residual BW less than the requested BW *D*.
2) For each network link *l*, determine its residual BW*RB (l)*, take the reciprocal of *RB (l)*, and normalize $(RB\ (l))^{-1}$ to the range 0–100, with the smallest $(RB\ (l))^{-1}$ value equal to 0 and the largest value equal to 100.Let *NRB (l)* be equal to the normalized value of $(RB\ (l))^{-1}$.
3) For the source–destination pair *(s1, v1)*, construct a link weight table $LWT_{(s1,v1)}$, using the following equation:

$$LWT_{(s1,v1)}(l) = NPPV_{(s1,v1)}(l) * (1\text{-}BWP) + NRB(l) * (BWP)$$

   Where *BWP* is the BW proportion parameter.
4) Run Dijsktra's algorithm to compute a link-weight optimized path between *(s1, v1)*.

Between the same source and destination pair, there is more than available Disjoint Path *DP*. When link *l* is included in a path between the source and destination *(s1, v)* pair, $PPV_{(s1, v1)}(l)$ will be decremented by one. When link *l* is included in a path between a different source and destination pair, $PPV_{(s1, v1)}(l)$ will be incremented by one. In the second stage, DORA prunes all links that do not have sufficient BW to route *D* units of BW. Then the algorithm combines the *PPV* value with residual link BW to form a weight value for each link that is used to compute a weight-optimized network path.

If all nodes can be a source node for all other nodes, then the first stage computes paths for different $|V|^2$ source and destination pairs. In each source and destination pair, the algorithm takes $O(|E|)$ to compute different *DPs*. Since Dijsktra's algorithm requires $O\ (|E| + |V|\ log\ |V|)$, then the first stage of DORA requires $O\ (|V|^2|E|^2 + |V|^3|E|\ log\ |V|)$. The overall computational complexity of second stage is similar to Dijsktra's algorithm which requires $O\ (|E| + |V|\ log\ |V|)$.

2.1.6 Least interference optimization algorithm

A.B. Bagula [45] introduced a Least Interference Optimization Algorithm (LIOA) which reduces the interference among competing flows by balancing the number and quantity of flows that are carried by a link to achieve efficient routing of MPLS BW guaranteed LSPs. In general, the simulation study demonstrates that LIOA outperforms many routing algorithms such as MHA, CSPF and MIRA algorithms. The comparative study of these algorithms is done with respect to different performance metrics including the rejection ratio of LSPs and the successful re-routing of LSPs upon single link failure [45].

According to the interference reduction technique, each algorithm has its own definition for the interference among competing paths. As mentioned before, MIRA aims to reduce the maximum flow in order to minimize the interference, while DORA computes the path potential value for every link in path to avoid routing over links that have a high potential to be part of any other paths in the future. In the LIOA algorithm, there is a big relationship between the interference and the number of flows which travel through a link.

The main objective of LIOA is selecting the shortest path between the source and destination pair *(s, v)* which contains links with minimum number and magnitude of flows. Consider a link cost function:

$$w(u,v) = \frac{{I_{uv}}^{lc}}{{S_{uv}}^{(1-lc)}}$$

Where I_{uv} is the number of flows carried on the link, *lc* is the least interference control parameter which represents a trade-off between the number and the magnitude of the flows traversing a link and the link slack $S_{uv} = R_{uv} - r_{uv}$, where R_{uv} is the maximum link capacity, r_{uv} is the total reserved BW. Depending on the desired link cost function, minimizing the number of flows I_{uv} and maximizing the link slake S_{uv} leads to minimizing the link cost function. Therefore, there is a minimum number of LSPs that are required to be rerouted again in case of single link failure. Note that the range of *lc* is *(0, 1)* and the special cases are:

$$w(u,v) = \begin{cases} \dfrac{1}{R_{uv} - r_{uv}} & lc = 0 \qquad (CSPF\ routing) \\ \dfrac{1}{R_{uv}} & lc = r_{uv} = 0 \quad (OSPF\ routing) \\ I_{uv} & lc = 1 \end{cases}$$

Algorithm 2.7 LIOA algorithm.

Input:	A graph *G = (V, E)* and a set *B* of all residual link capacities. An ingress node *s* and an egress node *v* between which a flow of *D* units have to be routed.
Output:	A route between *s*and *v* having a capacity of *D* units of BW.

Algorithm:

1) Prune the network. Set *w (u, v) =∞* for each link *l (u, v)* whose slake

$$(R_{uv} - r_{uv}) \leq D.$$

2) Find the new least cost path. Apply Dijkstra's algorithm to find the least cost path.

$$p^* = \{min(\, w(p) : \; u \sim v\}$$

3) Route the traffic demand. Assign the traffic demand *D* to the best path *p**.

4) Update the link reserved BW, interference and costs. For each link,

$$l\ (u, v)\ \in p^* : r_{uv} = r_{uv} + D;$$

$$I_{uv} = I_{uv} + 1;$$

$$w(u,v) = \frac{I_{uv}{}^{lc}}{S_{uv}{}^{(1-lc)}}$$

In the first step of LIOA, all links that do not have sufficient BW to route *D* units are pruned. Then Dijkstra's algorithm is used to find the shortest path between the source and destination pair. When the path is selected, the required *D* units of BW will be reserved in all links that belong to the best path. Finally, LIOA updates the link information within the best path. It decrements the available BW by *D* and increments the number of flows by one and then updates the link weight formula

with the new updates. In the context of commotional complexity, LIOA is similar to Dijkstra's algorithm which requires in the worst case $O(|V|^2)$ to compute the shortest path.

2.2 Estimation-based routing algorithms

The provided QoS depends on the accurate measurement of the available BW. Due to the varying nature of the available BW, globally updating the link state of all nodes with the current measured BW is not an efficient approach to represent the link utilization. Therefore, new approaches perform an estimation of the link utilization in the future depending on the actual traffic profile.

In the next sections, two different estimation-based routing algorithms are presented. In this type of routing algorithm, the future of link loads are estimated from the link load histories. These algorithms aim to reduce the interference between the paths in the future by incorporating the estimated available BW into the link weights formula. The first algorithm uses a statistical approach to estimate the available BW in the links. However the second algorithm depends on solving the linear algebra equations to estimate the available BW in the network links.

2.2.1 Distribution-free prediction interval routing algorithm

C. S. Yin and M. Yaacob introduced a statistical approach named Distribution-Free Prediction Interval (DF-PI) [46] which uses the maximum, the minimum and the average of last recent samples of available BW during a past period to estimate the future of available BW. DF-PI uses the estimated available BW in the link weights to enhance the performance of the WSP algorithm. DF-PI is proposed to work in the hop by hop QoS routing approach, but it can also be used in the explicit QoS routing. In general, DF-PI outperforms WSP algorithm in terms of the link utilization, the packet loss and the average end-to-end delay. Additionally, it has less overhead in updating messages.

Instead of estimating the future of available BW at random period, DF-PI proposes to use prediction intervals method. A prediction Interval (PI) is defined as "an interval that will, with a specified degree of confidence or prediction level, contain the next randomly selected observation(s) from a population" [47]. To

determine precisely PIs, the type of population distribution for a given random samples should be considered. In the available BW case, there are many factors which control these distribution characteristics such as the network topology, the arrival rates and the traffic distributions [48]. Due to the complexity of considering these distribution characteristics, DF-PI does not consider any kind of probability distribution to determine the available BW. In other words, the proposed mechanism uses the Distribution-Free (DF) approach. The general method of distribution-free PI construction is described using the following steps [46]:

1. Let order statistics of the sample be $x_{(1)} \leq x_{(2)} \leq ..\leq x_{(n)}$ whereby $x_1, x_2,..,x_n$ are independent and each with any continuous distribution function $F(x)$.
2. Specify the desired prediction level for the interval.
3. Determine (from tabulations or calculations) the order statistics that provide the PI with at least the desired prediction level. If no such order statistics exist, use the extreme order statistics to obtain the interval end points and determine the associate prediction level.
4. Use the selected order statistics as the endpoints of the distribution-free PI.

Given a sample $b(T)$ of size n which represent the observations of available BW, $b(t1), b(t2), b(t3), ..., b(tn)$ at related time $t1, t2, t3, ..., tn$. The statistical estimated available BW, δ, for the previous period that has $[b_l, b_n, b_u]$ values is proposed as:

$$\delta = \frac{(w_1 * b_n) + (w_2 * b_l) + (w_3 * b_u)}{w_1 + w_2 + w_3}$$

Where b_l and b_u are the minimum and maximum values in $b(T)$ and b_n is the current available BW that can be represented by the average of last five elements of sample $b(T)$.While the control parameters $\{w_1, w_2, w_3\}$are used to represent the effect of past and current states on δ calculation. In case of considering the importance of least future available BW, the control parameters should take values as $w_1 > w_2 > w_3$.

In the first step of the DF_PI algorithm, all the links that do not have sufficient BW to route D units are pruned. Then, the statistical available BW, δ, is calculated for every network link. In the rest of algorithm, the WSP algorithm is executed with a little modification. Instead of selecting the path with the

maximum available BW in case of existing equal shortest paths, DF-PI selects the path with the highest statistical available BW δ.

Algorithm 2.8 DF-PI algorithm.

Input:	A graph $G = (V, E)$ and a set B of all residual link capacities. An ingress node s and an egress node v between which a flow of D units have to be routed.
Output:	A route between s and v having a capacity of D units of BW.

Algorithm:

1) Consider only the links that have available BW more than the requested BW of flow.
2) For each link $l\ (u, v) \in E$, calculate The statistical estimated available BW δ_{uv}:

$$\delta_{uv} = \frac{(w_1 * b_n) + (w_2 * b_l) + (w_3 * b_u)}{w_1 + w_2 + w_3}$$

3) Run the WSP algorithm to find the shortest path between s and v nodes with the following modification:-
   - If there are more than shortest path exist:
   a) Calculate the statistical available BW of a path p,

      $\delta_p = \min(\delta_{uv}, \forall\ l\ (u, v) \in p)$.

   b) Select the one with the highest δ_p.

2.2.2 Path selection routing algorithm

T. Anjali and C. Scoglio [49] have proposed a linear prediction approach, named Path Selection Algorithm (PSA), which solves the linear prediction equations to estimate the available BW and also tells the duration for which the estimate is valid with a high degree of confidence. PSA utilizes the estimated available BW as weights in the shortest widest path algorithm and achieve some improvement in the routing performance.

PSA algorithm considers that all network information is available at the source node. This information includes the network topology and the link state information such as the available BW. The available BW information can be measured using active and passive approaches. In the active approach such as Path load tool [50], test packets are sent to measure a specific service. Although the active approach produces additional traffic into the network, it has the capability to measure the desired quantities at the desired time. On the other hand, the passive approach like NetFlow tool [51] is able to observe the network traffic without sending any packets.

PSA proposes to use the Simple Network Management Protocol (SNMP) [52] that gives the managed devices a means to store the management information and provides it to the management system on demand. With the help of SNMP framework, a modified version of the Multi Router Traffic Grapher (MRTG) [53] tool can be used to measure the average of available BW in a specific network link over 10 second durations.

PSA starts with running the available BW estimation procedure. Then it uses the estimated available BW to select the best path using the shortest widest path algorithm that prefers the minimum hop path in case of existing equal widest paths.

Algorithm 2.9 PSA algorithm.

1) At time instant k, a request with D units arrives between nodes $s$and v.
2) Run the available BW estimation procedure for links that has not estimated available BW.
3) Compute the best path using the shortest widest path algorithm with weights as calculated in step 2.
4) Obtain the available BW, A, on the bottleneck link of the path.
5) If $D > A * threshold$, reject this path and return to step 3. Else, path is selected for the request.
6) If no path available, request rejected and network is congested.

The PSA algorithm accepts the selected path if a certain fraction of the minimum available BW in this path more than the requested *D* of BW. If it is not more than the requested *D* of BW, then the algorithm rejects this path and tries to select another path between the source and destination pair.

Using the modified version of MRTG tool, the PSA algorithm is able to obtain the link utilization statistics for 10 second intervals. For a given link *l (u,v)* $\in$ *E*, the variables of available BW estimation algorithm are defined in Table 2.1.

Table 2.1 The variables of available BW estimation algorithm.

Variable	Mean
C	Capacity of link in bits per sec.
L (t)	Traffic load at time *t* in bits per sec
A (t)	Available capacity at time *t* in bits per sec that is equal to *C – L(t)*
τ	Length of the averaging interval of MRTG.
L_τ *[k]*	Average load in $[(k-1)\tau, k\tau]$.
pm	The number of past measurements in prediction.
h	The number of future samples reliably predicted.
$A_h[k]$	The estimate at kτ valid in $[(k+1)\tau, (k+h)\tau]$.

The problem of traffic load estimation after samples $a \in [1, h]$ can be formulated as linear prediction:

$$L_\tau[k+a] = \sum_{n=0}^{pm-1} L_\tau[k-n]\, w_a(n) \qquad \text{Equation (2.1)}$$

Where $w_a\ (n)$ represents the prediction coefficients. In general, Wiener-Hopf equations [54] are used to solve this kind of problems. They are given in a matrix form as $R_L\, w_a = r_a$, *for* $a \in [1, h]$:

$$\begin{bmatrix} rL(0) & \cdots & rL(pm-1) \\ \vdots & \ddots & \vdots \\ rL(pm-1) & \cdots & rL(0) \end{bmatrix} \begin{bmatrix} w_a(0) \\ \vdots \\ w_a(pm-1) \end{bmatrix} = \begin{bmatrix} rL(a) \\ \vdots \\ rL(a+pm-1) \end{bmatrix}$$

Where *rL(n)* is the autocorrelation of the sequence from the available measurements:

$$rL(n) = 1/N \sum_{i=0}^{N} L_\tau[k-i]\, L_\tau\,[k-i-n]$$

Since N is the requested estimation accuracy. If you need more precise, you should give N high value. To compute the autocorrelation value, the algorithm needs $(n + N)$ measurement samples. PSA algorithm uses Levinson recursion method [55] to solve the Wiener-Hopf equations and compute w_a vector.

Algorithm 2.9.1 Available BW estimation procedure.

1) At time instant k, available bandwidth measurement is desired.
2) Find the vectors w_a, $a \in [1, h]$ using Wiener-Hopf equations given pm and the previous measurements.
3) Predict $[\widehat{L}_\tau [k+1], ..., \widehat{L}_\tau [k + h]]^T$ from Equation (2.1).
4) Compute $A_h[k] = C - max \{\widehat{L}_\tau [k+1], ..., \widehat{L}_\tau [k + h]\}$.
5) At time $(k + h)\,\tau$, get $[L_\tau [k+1], ..., L_\tau [k+h]]^T$.
6) Find the error vector get $[e_\tau [k+1], ..., e_\tau [k+h]]^T$ since,

$$e_\tau [k+a] = (L_\tau [k+a] - \widehat{L}_\tau [k+a])^2 \text{ for } a \in [1, h].$$

7) Set $k = k + h$.
8) Run the interval variation procedure to obtain new values for pm and h.
9) Go to step 1.

The available BW estimation procedure solves the Wiener-Hopf equations and determines the w_a vector. Then it substitutes with the w_a and past measurement values in Equation (2.1) to get on the predicted load values. Depending on the predicted load values and the link capacity, the procedure computes the predicted available BW, where $A_h[k] = C - max \{\widehat{L}_\tau [k+1], ..., \widehat{L}_\tau [k + h]\}$.

In the fifth step, all the actual values of traffic load within the links are maintained. In step six, the error vector is calculated to maintain the prediction accuracy for the last prediction processes, where the error value is:

$$e_\tau [k+a] = (L\tau [k+a] - \widehat{L}_\tau [k+a])^2 \text{ for } a \in [1, h].$$

Finally, the procedure sends the error vector to the interval variation procedure to obtain new values for pm and h and increments the k parameter with h value.

Firstly, the interval variation procedure calculates the mean (μ), standard deviation (σ) and the maximum value (M^2_E) of error e_τ. Additionally, the procedure sets boundaries to both of *pm* and *h* values by defining pm_{max}, pm_{min}, h_{min} variables. Also it introduced the thresholds *Th1* to *Th4* to determine how the changes of *pm* and *h* happened. Depending on the comparison between the statistics values of error and the thresholds parameters, the new values of *pm* and *h*are determined and returned back again to be used in the available BW estimation procedure.

Algorithm 2.9.2 Interval variation procedure.

1) If $\sigma/\mu > Th1$, decrease *h* till h_{min} and increase *pm* till pm_{max} multiplicatively.
2) If $Th1 > \sigma/\mu > Th2$, decrease *h* till h_{min} and increase *pm* till pm_{max} additively.
3) If $\sigma/\mu < Th2$, then:
   a) If $\mu > Th3 * M^2_E$, decrease *h* till h_{min} and increase *pm* till pm_{max} additively.
   b) If $Th3 * M^2_E > \mu > Th4 * M^2_E$, keep h_{min} and *pm* constant.
   c) If $\mu < Th4 * M^2_E$, increase *h* and decrease *pm* till pm_{min} additively.

2.3 Decentralized routing algorithms

In contrast to the centralized routing, in the decentralized routing, every node has only a local view of network status. Therefore, every node takes its own routing decision to forward the packets to an outgoing link. This means that multiple decisions are taken in all intermediate nodes between the source and destination pairs. While the decentralized routing approach is more tolerant in case of link or node failure and does not require advertising the changes to all network nodes, it requires an efficient framework to select the near optimal path because all routers are burdened with the route calculation.

This part focuses on a self-organized and decentralized routing approach called Ant Colony Routing (ACR). ACR algorithms are inspired from real ants' behavior. The ants are able to discover the shortest way to a food source and their nest without any knowledge of geometry but with a keen sense of smell.

ACR algorithms are considered as a member of Swarm Intelligence (SI) algorithms. SI is defined as "a computational technique for solving distributed problems inspired from biological examples provided by social insects such as ants and by swarm, flock and shoal phenomena such as fish shoals and bird flocks" [56]. In the SI system, there are many distributed agents with local views that can communicate with each other to solve different kind of problems. In general, the SI has the following features:

- Autonomy: Each individual controls its own behavior without outside management or maintenance.
- Adaptability: The individuals communicate with each other directly or indirectly via the local environment. Using the indirect communication method, the individuals are able to detect any change in the surrounding environment and have the ability to adapt their behavior to face the new changes.
- Scalability: There is no degradation for the overall system performance when the number of individuals is increased.
- Flexibility: Each individual in SI has the same priority and can be added, deleted or replaced.

- Robustness: In case of any failure, the overall system has the ability to be fast robust because there is no central control.
- Massively parallel: Each individual is performing the same task.
- Self-organization: SI system has all self-organization capabilities.
- Cost effectiveness: As mentioned before, each individual has the autonomy feature and there is no central control. Thus, these features lead to reduce the cost of the design and implementation task.

Many applications use the SI principles. For example, the solutions for discrete optimization problems can be found using the Ant Colony Optimization (ACO) meta-heuristic [57]. Another application of SI is the Particle Swarm Optimization (PSO) algorithm which is used to solve nonlinear optimization problems with constraints [58].

One of the important SI feature is the self-organization. Self-organization is defined as "a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of a system without any external or centralized control" [59].The principles of self-organization in social insects are interpreted through the following methods [60]:

- Positive feedback: The ants aim to reinforce the good alternative solution for a problem more than the bad alternative solution. Therefore, it deposits an amount of pheromone depending on the quality of solution.
- Negative feedback: In contrast to the positive feedback, an ant reduces the amount of pheromone for a solution. The negative feedback concept is always used to interpret the satiability among the collective pattern. The ants use the negative feedback concept when the food source is exhausted or during the competition among different sources.
- Interactions among individuals and with the environment: The ants can not only communicate directly with each other, but can also communicate indirectly via the environment. In the indirect communication type, which is called "stigmergy", an ant deposits a certain amount of pheromone in a path as a good sign for the food to let other ants follow this path?

- Probabilistic techniques: The randomness can be observed form the ant live. But in the artificial ant process, the ants use the stochastic process to move from one state to another state.

ACR algorithms are based on the ACO meta-heuristic. The ACO meta-heuristic depends on a multi-agent structure to solve discrete optimization problems. Each autonomous agent has its own memory that is used to memorize the solution states and uses the stochastic process to take its decision. In general, ACO meta-heuristic can be considered as a distributed learning approach. In addition to all inspired characteristics form the real ants, the artificial ants of ACO meta-heuristic have other characteristics such as the look ahead [61], local optimization [62] and backtracking [63] capabilities. The ACO algorithm can be used to search for the near-optimal solution for different kind of problems such as scheduling [64], routing [65] and assignment [66] problems. The first ACO algorithm is the Ant System (AS) algorithm [67] which is proposed to solve the Travelling Salesman Problem (TSP).

In the following sections, the AntNet algorithm, which is the oldest algorithm in the ACR class, is represented in more details. Then, an advanced algorithm from the same class, called Trail Blazer algorithm, is presented.

2.3.1 AntNet routing algorithm

AntNet [63], [68] is an ACO algorithm for distributed and adaptive best-effort routing in IP networks. AntNet is considered the first algorithm, which is inspired from the ant colony behavior, addressed for the use in routing. The behavior of AntNet depends on the mobile agents "ants'" framework. The main characteristics of AntNet algorithm comprise the following steps [69]:

- At regular intervals, mobile agents or ants are forwarded form the source node s to the destination node d. During the forwarding phase, these mobile agents are called forward ants $F_{s \to d}$.
- Each ant constructs a path by taking a sequence of decisions based on a stochastic policy parameterized by local pheromones and heuristic information.

- During the trip between the source and destination pair, the ants collect information about the trip time and identify the visited nodes.
- Once it has arrived at the destination, the backward phase starts. The backward ants $B_{d\to s}$ retrace the route $p_{s\to d} = \{s, v_1, v_2, \ldots, d\}$ that was followed by their forward ants.
- The backward ants update the local routing information in all the intermediate nodes $v_k \in p_{s\to d}$. The local routing information consists of two tables: The statistical model table M^{vk} that contains the expected end-to-end delays, the pheromone or routing table $\mathcal{T}^{vk}$ that is used to route the data packets.
- The backward ants are removed from the network when they have reached the source node s.
- When the regular data packets arrive at anode, they are forward according to the information in the $\mathcal{T}^k$ table.

Figure 2.1 shows the node data structure in AntNet algorithm. The pheromone or routing table $\mathcal{T}^k$ contains N columns, one for each destination node. Each column has L entries, one for each outgoing link of the node. The entry τ_{ij} contains the probability of sending a packet to destination j on the outgoing link i. Each entry has the same meaning of pheromone variable in the ACO meta-heuristic which represents the goodness of selecting an outgoing link to forward the date packets through it.

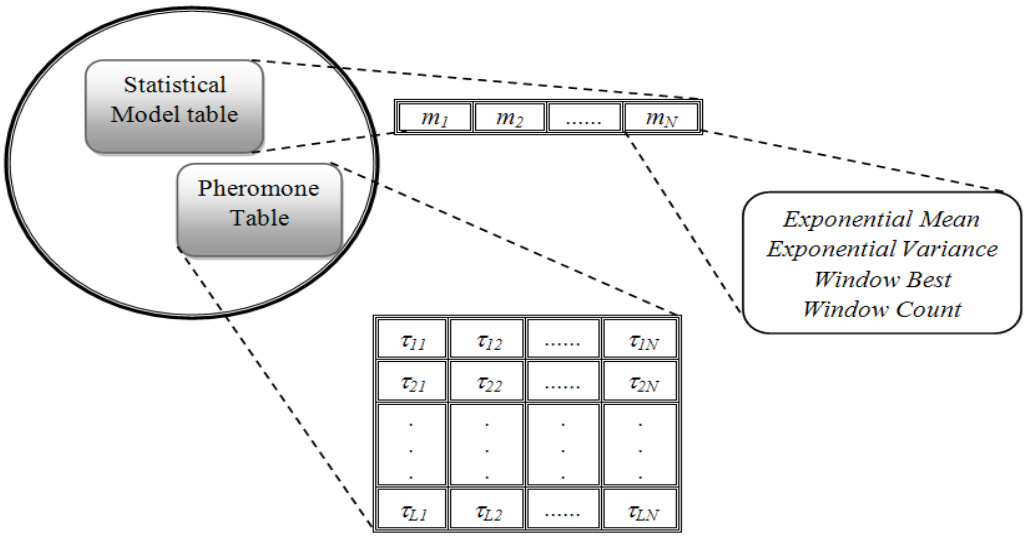

Figure 2.1 The node data structure in AntNet algorithm.

The AntNet algorithm takes the optimal local decision by adapting the pheromone values during the continual search. The range of τ_{ij} entry is [0, 1] and the sum of destination column is one:

$$\sum_{n=N_k} \tau_{nd} = 1, \quad d \in [1, N],\ N_k = \{neighbors(k)\}.$$

The statistical model table M^k contains only one row which has an entry to each destination node. M^k contains the following statistical information:

- μ_d is the mean of the traveling time to reach destination d from the current node,
- σ^2_d is the variance of this traveling time and
- W_d is the best traveling time to reach destination d during the window of the last w observations.

The following exponential model is used to estimate the new μ_d and σ^2_d values:

$$\mu_d \leftarrow \mu_d + \eta\,(o_{k\to d} - \mu_d) \qquad \text{Equation (2.2)}$$

$$\sigma^2_d \leftarrow \sigma^2_d + \eta\,((o_{k\to d} - \mu_d)^2 - \sigma^2_d) \qquad \text{Equation (2.3)}$$

While $o_{k\to d}$ is the new observation of the travelling time between node k and destination node d. The factor η controls the number of effective samples that is used to find the best traveling time W_d during it. With the experimental result, the effective relationship between w and η is: $w=5(c/\eta),\ c \in [0, 1]$.

The AntNet algorithm proposes to use an adaptive reinforcement approach in order to compute the reinforcement parameter r which is used as a function of the goodness for the observed trip time. AntNet algorithm depends on the ant's trip time T and the statistical variables of M^k to compute the value of r parameter:

$$r = c_1\left(\frac{W_d}{T}\right) + c_2\left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (\mathrm{T} - I_{inf})}\right) \qquad \text{Equation (2.4)}$$

The c_1 and c_2 parameters represent the importance of each term. I_{sup} and I_{inf} are used to estimate the limits of an approximate confidence interval for μ:

$$I_{inf} = W_d \qquad \text{Equation (2.5)}$$

$$I_{sup} = \mu d + z\left(\frac{\sigma^2{}_d}{\sqrt{w}}\right) \qquad \text{Equation (2.6)}$$

While $z = \left(\frac{1}{\sqrt{(1-\gamma)}}\right)$, where γ is the confidence coefficient.

Using the calculated reinforcement parameter r, AntNet algorithm adjusts the probability entries of the pheromone table $\mathcal{T}^k$. the probability of selecting the outgoing link f when the destination is d, τ_{fd}, will be increased by a value proportional to the reinforcement parameter r:

$$\tau_{fd} \leftarrow \tau_{fd} + r\,(1 - \tau_{fd}) \qquad \text{Equation (2.7)}$$

The probabilities of selection other outgoing links that are connected to N_k, τ_{nd}, will be adjusted according to the following formula:

$$\tau_{nd} \leftarrow \tau_{nd} - r\,\tau_{nd}, \quad \forall n \neq f \qquad \text{Equation (2.8)}$$

Algorithm 2.10 AntNet algorithm.

1) At regular intervals Δt, a source node s sends the forward ant $F_{s \rightarrow d}$ *to* a destination node d. The forward ants are created at node s with the following probability: $p_d = \frac{f_{sd}}{\sum_{n=1}^{N} f_{sn}}$, where f_{sd} is the amount of traffic between s and d nodes.
2) Each ant has an internal stack memory $S_{s \rightarrow d}\,(k)$ which maintains the node identifier for node k that belong to the visited path and the time elapsed until arriving this *k-th* node.
3) At each intermediate node k, the ant aims to select an outgoing link based on a stochastic policy parameterized by local pheromones and heuristic information.
4) The forward ants aim to avoid the cycle occurrence by selecting the next hop that does not make any loop. If there is no any other choice, then the ant is destroyed.
5) Once the forward ant arrives at the destination node, another backward ant $B_{d \rightarrow s}$ is generated and the stack memory of the forward ant is transferred to its memory.
6) Using the stack memory $S_{s \rightarrow d}$, the backward ant takes the reversed path towards the source node.

7) At a node k that belongs to the reserved path, both of the statistical model table M^k and the pheromone table $\mathcal{T}^k$ are updated according to the following steps:
   a) The backward ants use the information in the stack memory $S_{s\to d}(k)$ to compute the mean μ_d and variance σ^2_d using Equations (2.2) and (2.3) and update W_d variable with the best traveling time value.
   b) Compute the reinforcement parameter r using Equations (2.4), (2.5) and (2.6).
   c) Update the pheromone table $\mathcal{T}^k$ entries by substituting with r value in Equations (2.7) and (2.8).

In general, the AntNet algorithm shows superior performance under different experiment conditions compared to different routing algorithms from the machine learning fields such as Q-Routing [70] and Predictive Q-Routing [71] algorithms. A lot of algorithms have been proposed based on AntNet algorithm to overcome some limitations such as the adaptive routing algorithm that has been proposed by Yun and Zincir in order to reduce the information in the routing table [72].

2.3.2 The Trail Blazer routing algorithm

The Trail Blazer (TB) algorithm is an intra-domain routing algorithm that aims to minimize the network congestion through local decisions, based on latency measurements collected by scout packets [73]. The TB algorithm is another example of ACR algorithms which maintains a probability table in each node to determine the probability of selecting an outgoing link in order to forward the data packets to a given destination. The TB algorithm is meant to be an extension of existing link-state protocols such as OSPF, which provides shortest path information to initialize the probability table. Therefore, TB does not require a learning period to discover the network topology. TB is also simpler than the AntNet algorithm.

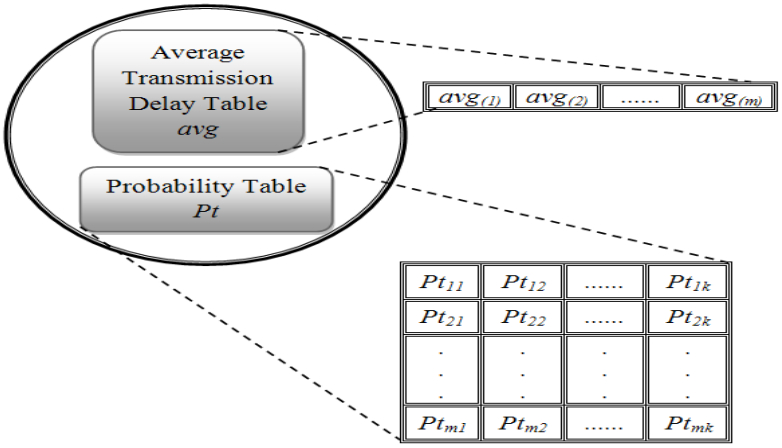

Figure 2.2 The node data structure in TB algorithm.

Figure 2.2 shows the node data structure in TB algorithm. In the TB design, each router has two tables: a link probability table *Pt* and an average transmission delay table *avg*. *Pt* contains *m* rows, one for each destination node. Each row has *K* entries, one for each outgoing link of the router. The entry *pt[d,i]* is the probability of sending a packet to destination d on the outgoing link *i*. The table *avg* has *m* entries, one for each destination node. The entry *avg(d)* is the average transmission delay from the current node to the destination *d*, which is computed from the last *M* scout packets that arrived from *d*.

The TB algorithm sends an amount of scout packets depending on the amount of regular data traffic between the source and destination pair. Scout packets maintain the list of visited nodes to detect loops. TB algorithm can operate in two different modes. The first mode is One-way mode. In this mode the scout packets are sent from the destination toward its source. During the trip from the destination to the source, the latency is accumulated by the scout packets in order to update the probability tables in all intermediate nodes. The scout packets update their accumulated latency *td* in every intermediate node by $td = td + t(i)$, where *t(i)* is the current latency of the outgoing link *i*. Once the scout packets arrive at the source, they will be destroyed.

In the second mode which is called Two-way, the scout packets are sent from the source towards its destination in order to keep the local time stamp of all visited nodes. Once the scout packets arrive at the destination, they will take the reverse path back to the source. The latency from the current node to the destination on the return trip is computed by subtracting the saved time stamp at this node from

the saved time stamp at the destination node. Once the scout packets arrive at the source, they will be destroyed also.

The *Pt* table is initialized with the collected information from the computed shortest paths by the underlying link-state protocol. Let *SP (d)* represents the set of outgoing links that belong to one or more shortest paths from the current node to destination. The initial value of *pt* is:

$$pt[d,i] = \begin{cases} \frac{1}{|SP(d)|} & i \in SP(d) \\ 0 & otherwise \end{cases}$$

After computing the accumulated latency *td*, the scout packets use the *td* value to update the *pt* as follows:

$$f(td) = max\left(min\left(\frac{avg(d)}{td}, 10\right), 0.1\right) \quad \text{Equation (2.9)}$$

$$\Delta p = \delta \times f(td) \quad \text{Equation (2.10)}$$

$$pt[d,i] = \frac{(pt[d,i] + \Delta p)}{(1 + \Delta p)} \quad \text{Equation (2.11)}$$

$$pt[d,j]_{j \neq i} = \frac{(pt[d,j])}{(1 + \Delta p)} \quad \text{Equation (2.12)}$$

The average latency *avg(td)* is used to scale the positive reinforcement value of the scout packet. A larger value of *f (td)* indicates a better (shorter) path. *f(td)* is limited to the range *[0.1,10]* to prevent wide fluctuations in Δp, which is the reinforcement value of *pt[d,j]*.δ defines the learning rate of the algorithm. All entries in *Pt* of the same destination *d* are scaled by *1*+Δp to ensure that their sum remains is equal to one.

Algorithm 2.11 TB algorithm (Two-way mode).

1) Initialize the *Pt* table with the collected information from the computed shortest paths by the underlying link-state protocol.
2) Send a scout packet every *N* regular data packets from the source node *s* to the destination node *d*.
3) Each scout packet maintains the node identifier for node *k* that belongs to the visited path and the local time stamp of this node.

4) At each intermediate node *k*, the outgoing link is selected based on uses type of scout packets. There is two types of scout packets, Best-path scout packets and exploratory scout packets.
5) The scout packet aim to avoid the cycle occurrence by selecting the next hop that does not make any loop. If there is no any other choice, then the scout packet is destroyed.
6) Once the scout packet arrive the destination node, it will take the reserved path toward the source node *s*.
7) At a node *k* that belongs to the reserved path, both of the average transmission delay table *avg* and the probability table *Pt* are updated according to the following steps:
   a) Compute the latency from the node *k* to the destination *d, t(i),* by subtracting the saved time stamp at this node from the saved time stamp at the destination node.
   b) Update the average transmission delay table *avg.*
   c) Compute the reinforcement parameter Δp using Equations (2.9) and (2.10).
   d) Update the probability table *Pt* entries by substituting with Δp value in Equations (2.11) and (2.12).
8) On the other hand, when a node receives the regular data packets, which needs to be forwarded, data packets will be routed according to the probabilities in *pt* table.

In general, the communication overhead of TB algorithm is low because the number of scout packets is too small compared to the number of regular data packets. Also the TB algorithm aims to reduce the path oscillations by updating the probability table with a combination of old and new information. With the experimental result, TB algorithm has the least packet drop rate due to network congestion compared to the SPF algorithm.

CHAPTER 3 - Artificial neural network: Model selection and traffic prediction

The objective of this chapter is to introduce an overview about the Artificial Neural Network (ANN) and present the qualification and efficiency of ANNs within the network traffic prediction field. In the first section, the ANN is defined and its properties are presented. The second section demonstrates the neuron biological model and the design of the emulated artificial model.
Different methodologies of constructing ANN are discussed in section three. In section four, the differences between various learning processes are described. In section five, a lot of ANN applications are presented. The last section focuses on the use of ANNs as a traffic prediction model. Additionally, it presents the qualifications of ANNs that let it one of the most used traffic prediction models.

3.1 Background

The ANN is defined as "an information processing paradigm that is inspired by biological nervous systems, such as the brain, to process the information" [74]. ANN is constructed by interconnecting many information processing elements called neurons to solve different kinds of problems. The internal structure of ANN is changed during a learning phase based on a history of information called training samples.
The first artificial neuron was proposed in 1943 by W. McCulloch and W. Pitts [75] to emulate the structure of a biological neuron. The first ANN model which was called Perceptron has been proposed by F. Rosenblatt in 1958 [76]. The perceptron model aims to emulate the processing of visual data in the human brain and the ability to recognize the objects. In general, ANNs is preferred to be used due to having the following features:

- Adaptive learning: The ability to change the internal structure based on the training samples.
- Self-organization: Each neuron works on its local information and all neurons communicate with each other to achieve a global object.

- Real time operation: ANN aims to decrease the computation complexity. Therefore, the computation is done in a parallel manner.
- Non-linear model: ANN can be represented as anon-linear model. It is the suitable technique when the linear approximation is not valid.
- Fault tolerance via redundant Information: There is no degradation of ANN performance in case of any neuron failure.
- VLSI implementation: the massively parallel nature of ANN makes it easy to implement it using the Very Large Scale Integrated (VLSI) technology.

3.2 The neuron model

This section focuses on the main structure and functions of a neuron element. The design of the artificial model for the neuron element is based on the biological neuron model. Therefore, the biological neuron model is firstly discussed in details during the next section.

3.2.1 The biological model

The human brain contains approximately 10 billion interconnected neurons [77]. The main objective of a neuron is processing the signal information and transmitting it to its neighbors. Figure 3.1 shows the biological neuron. The main core of neuron is the soma or the cell body which processes the signals and transmits it to other neurons through an extension called axon. The soma receives the signals from other neurons through many small extensions called dendrites. A neuron's dendrites tree is connected to very large numbers of neighboring neurons. The strengths of received signals are added through the processes of spatial and temporal summation.

The neuron has only an axon which is connected to the soma at a point called axon hillock and the axon end is separated into several branches. At the end of branches, there are many connection points called synapses which in turns are connected to dendrites for other neurons to transmit the processed signal to them. The nature of transmitted signals is electrical which is called action potentials.

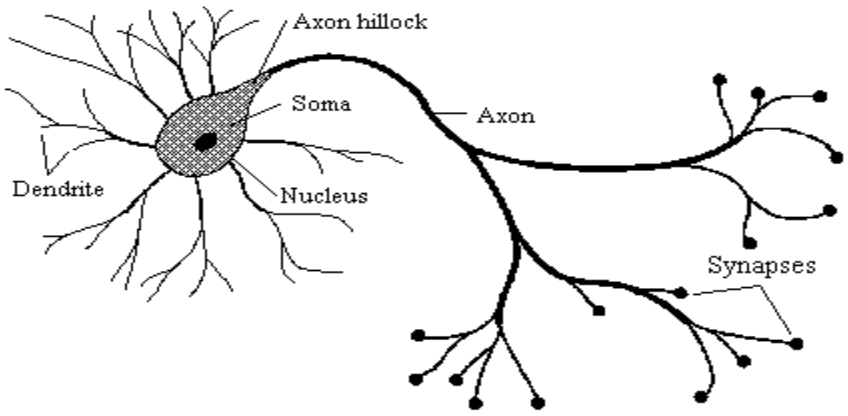

Figure 3.1 The biological neuron.

3.2.2 The artificial model

Figure 3.2 shows the neuron model that is widely used within the ANN field [78]. The artificial neuron has usually n inputs, named x_1, x_2, x_3... x_n and only one output y. The processing of information within the artificial neuron can be interpreted by the following mathematical operations:

- Synaptic operation: The incoming signals to the neuron are assigned weights, named w_1, w_2, w_3 ….w_n. The weights role is similar to the synapses role within the biological model which represents the significance of corresponding incoming signals. The weighted signals, corresponding to the dendrites in the biological model, are forwarded directly to the next artificial neuron section.
- Somatic operation: There are two different somatic functions, the aggregation and output (activation) functions. Firstly, the weighted signals are aggregated in a specific manner. The output or activation function controls the value of the output signal depending on a transfer function. The range of an output signal is usually between 0 and 1 or -1 and 1.

Both of inputs and weights can represent real values. If the connection has a positive weight value, this indicates an excitatory effect. On the other hand, if the connection has a negative weight value, this indicates an inhibitory effect.

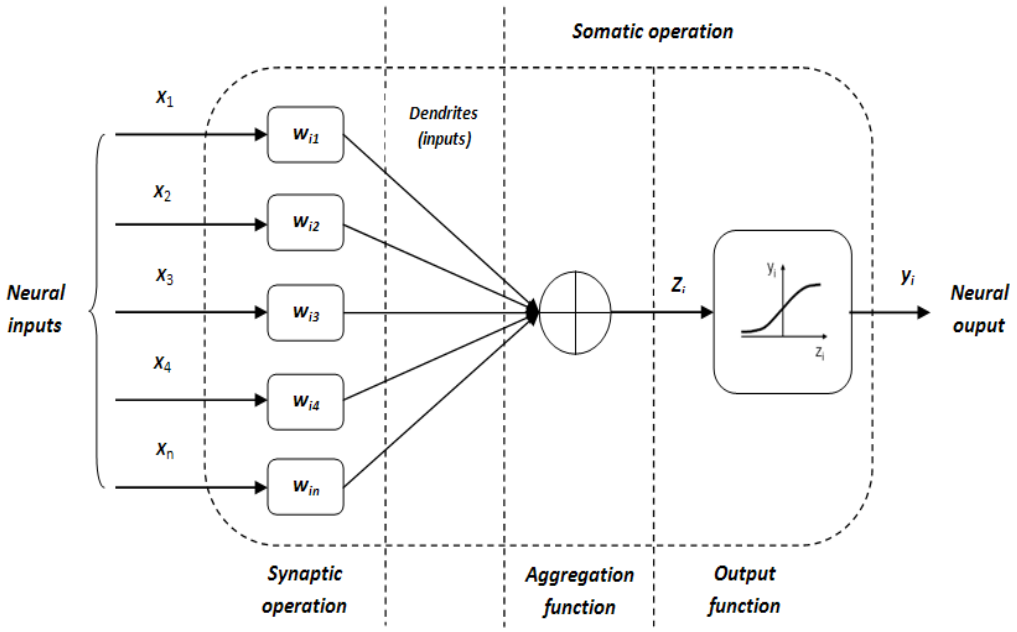

Figure 3.2 The artificial neuron.

There are many aggregation functions. The most common one is the linear combination method. In the linear combination, the dot product concept is used to combine the inputs and weights values according to the following equation:

$$z_i = X \cdot W_i = \sum_{j=1}^{n} x_j \, w_{ij}.$$

There are a lot of output functions and the selection of the suitable output function depends on the application [79]. Figure 3.3 shows different output functions and the description of them is summarized as follow:

- Step function: The oldest output function is the Step function which is proposed by W. McCulloch [75]. The output of artificial neuron, $y_i = \begin{cases} 1 & if\ z_i \geq 1 \\ 0 & if\ z_i < 1 \end{cases}$, where z_i is the aggregation of weighted signals.
- Linear function: The neuron output value is proportional to the aggregation of weighted signals, $y_i = k*z_i$, where k is constant. If k is equal to 1, the function is called identity function. This type of function is used in the linear neural network [80].
- Ramp function: This function is a combination of the step and linear functions. The period between T and $-T$ is linear. The function approximation is one of these function applications [81]. The ramp function equation is:

$$y_i = \begin{cases} T & if\ z_i \geq T \\ z_i & if\ z_i < |T| \\ -T & if\ z_i \leq -T \end{cases}$$

- Sigmoid function: The sigmoid function can be considered as a continuous version of the ramp function. Additionally, it provides a non-linear response. This function is usually used in the Multi Layered Perceptron (MLP) for classification tasks [82]. A common example of the sigmoid function is the logistic function which is represented by the following equation:

$$y_i = \frac{1}{1 + e^{-az_i}}$$

- The Gauss function: The gauss function forms a sympatric shape around the origin. This function is controlled by a parameter called σ which forms the wide of function curve. The Radial Basis Function (RBF) neural network uses this function [83]. The gauss function is represented by the following equation:

$$y_i = e^{-\frac{z_i^2}{2\sigma^2}}$$

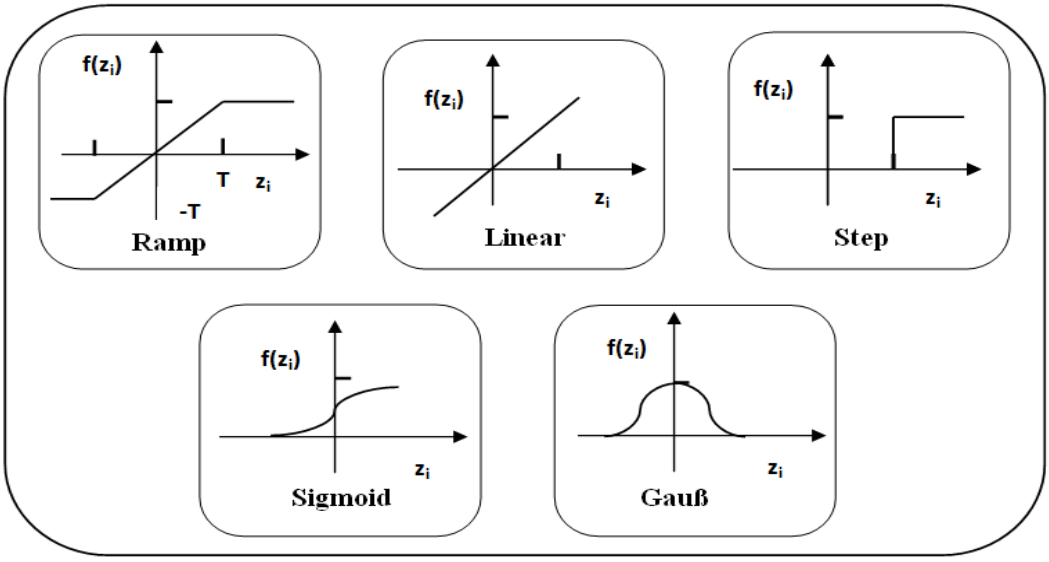

Figure 3.3 The output functions.

3.3 Neural network models

The next sections show the different methods that describe how the neurons are connected in order to form various models of neural networks.

3.3.1 Homogeneous vs. structured networks

Most neural network models are homogeneous. The structure of a homogeneous neural network is very formal and regular such as the feed forward neural network that will be described in details later. This model does not consider the knowledge about local connections or the relationship between the internal neurons. In the context of comparison between the homogeneous neural network and biological neural network, the architecture of them is not very similar.

In contrast to the homogeneous neural network, the structured model allows the integration of structure knowledge by considering specific local connections or relations between the neurons in the network. This model is more similar to the biological neural network than the homogeneous neural network. An example of a structured neural network is the structure of local connections of the pyramidal cells in the visual cortex (see Figure 3.4).

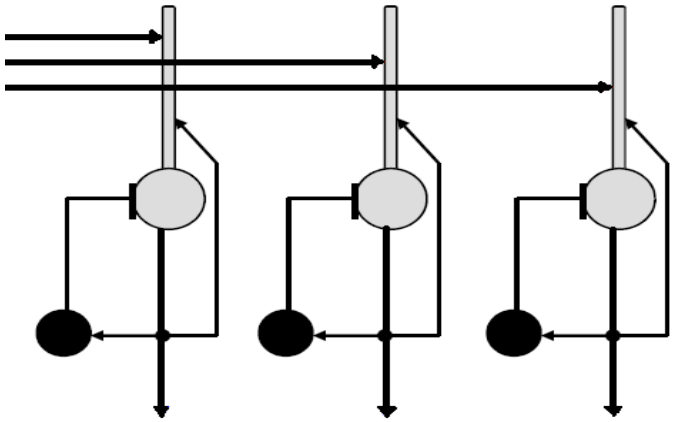

Figure 3.4 The pyramidal cells in the visual cortex.

3.3.2 Feed forward vs. recurrent networks

ANN can be categorized into Feed Forward Neural Network (FFNN) and Recurrent Neural Network (RNN), depending on the pattern of connections between the neurons and the propagation of signals inside the neural network.

In the FFNN, the neurons are ordered in layers and each layer is connected to the following layer only. By default, each neuron is connected to all neurons in the following layer. There are no feedback connections are present in this kind of topology (see Figure 3.5). The data is forwarded from the input layer through the hidden layers to the output layer. In other words the propagation of data is done in

a unidirectional way. During the design of FFNN, the number of hidden layers and the number of neurons within it represent the ability of the neural network. The MLP is a classical example of FFNN [82].

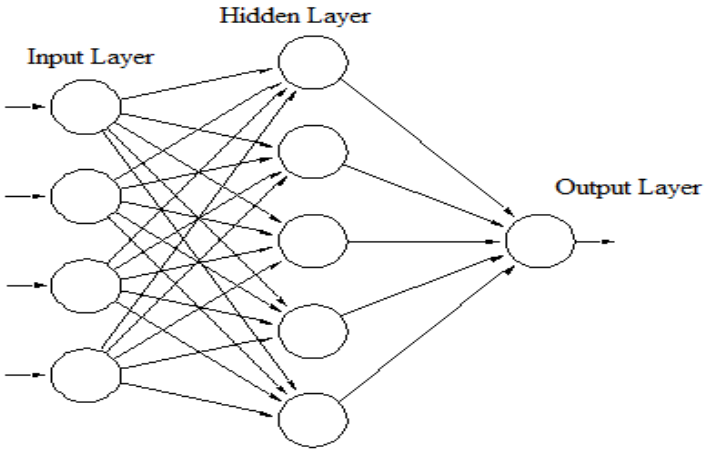

Figure 3.5 Feed forward neural network.

In contrast to the FFNN, each neuron in the RNN can be connected to any neuron regardless its position in the network (see Figure 3.6). In other words, this kind of neural network contains a feedback mechanism which allows it to exhibit a dynamic temporal behavior. The feedback mechanism gives the ability to create an internal state which can be used as internal memory to process arbitrary sequences of inputs. The famous RNN model was invented by John Hopfield [84].

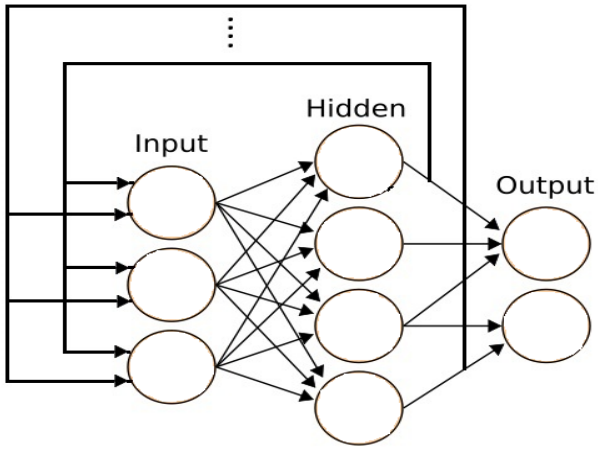

Figure 3.6 Recurrent neural network.

3.3.3 Fully vs. partially connected networks

In the fully connected neural network (see Figure 3.7), each neuron is connected to all other neurons in the network. This type of architecture is the most general

neural network architecture. However, it requires a considerable time to determine the internal weights values. An example for this neural network model is the fully connected RNN that is used as auto-associative memory for the pattern memorization task.

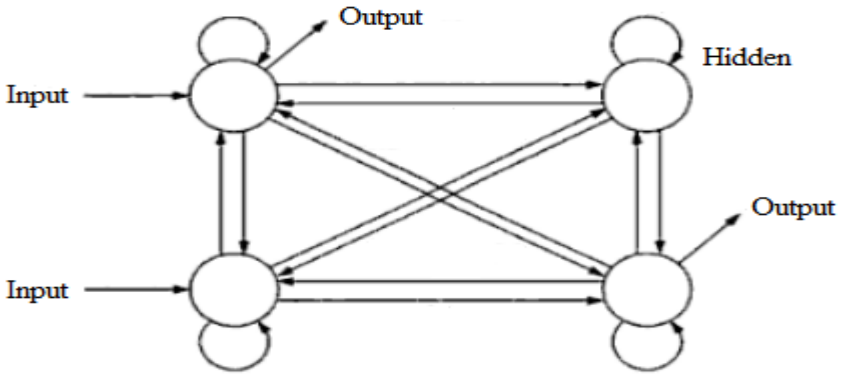

Figure 3.7 Fully connected neural network.

On the other hand, the partially connected neural network contains only a sub-set out of all possible connections between the neurons (see Figure 3.8). The main purpose of pruning some connections is to reduce the training time of the neural network and to simplify the implementation process in order to reduce the overall cost. An example of this neural model is the Partially Connected Feed Forward Neural Network (PCFNN) [85].

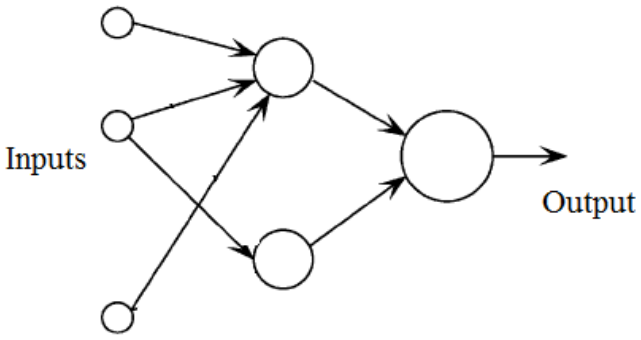

Figure 3.8 Partially connected neural network.

3.4 The learning process

The internal structure of ANN is constructed during the learning or training process. There are different methods to set the weight values of the neural

network in order to achieve the desired function. Three different learning methods are described in the next sections.

3.4.1 Error-driven learning

The error-driven learning is supervised learning. In this learning method, the ANN has a priori set of output classes into which the input patterns will be classified. The ANN is trained on data samples called a training data set. A part of this data is formed as inputs for the neural network and another part is formed as desire output named target.

The standard training process uses the back-propagation method [86]. During the training process, each weight in the network takes a specific value depending on the desire target. At the beginning of the training process, each weight takes an initial random value and then the errors of weights are computed. To minimize the total error, the gradient decent method is used to adjust the weight values depending on the errors of weights.

The last step of the training process is to validate the weight settings on another data set called validation data set in order to check the correctness of the desired neural network function. The traditional examples of feed forward topologies, which use the supervised learning method, are the MLP and the Time Delay Neural Nets (TDNN) [87]. The examples of recurrent topologies are the Jordan network [88] and the Elman network [89].

3.4.2 Unsupervised learning

In contrast to the error-driven learning, the ANN, that uses the unsupervised learning, has not any priori set of output classes into which the input patterns will be classified. During the unsupervised learning process, all input patterns, which are similar in the statistics, are grouped into a specific cluster. The clustering is based on the property distribution function of input data.

Typical examples of the unsupervised learning algorithms are the Hebbian learning algorithm, and the competitive learning algorithm. The main idea of the Hebbian algorithm bases on increasing the weight between two neurons when the two neurons have highly correlated outputs at the same time. The main objective

of the competitive learning algorithm is to give the weights specific values in order to let some neurons in the network respond to a subset of input data.
The traditional examples of feed forward topologies, which use the unsupervised learning method, are the Self-Organizing Feature Maps (SOFM) [90] and the Neural Gas (NG) [91]. The examples of recurrent topologies are the Hopfield networks and the Boltzmann-Machine [92].

3.4.3 Reinforcement learning

The reinforcement learning can be considered a specific learning type in between the supervised and unsupervised learning. In this type, there is a feedback signal from the environment after a sequence of inputs that decides if the output is right or wrong. However the feedback signal nature is only evaluative, not instructive.
The reinforcement learning has three components: the agent, the environment and the actions. The agent is the decision maker that takes action according to specific responses from the environment. After each action, the agent receives a reward. The main target of an agent is to select the nearly optimal actions in order to maximize the received reward during a specific period of time. In general, ANNs are usually used in the reinforcement learning as part of the overall algorithm such as the Q-learning Neural Network (QNN) that is used to approximate value functions in the Q-learning approach [93].

3.5 Neural network applications

The ANNs have a lot of real life applications within different fields. The applications of ANNs can be categorized depending on the desired task. Some of these applications are presented in the following:

- Classification: the typical ANN application is the classification including the pattern recognition.
- Data compression: ANN has the ability to receive a big amount of data, such as image data, and process it in parallel in order to represent an efficient compression schema.
- Prediction: ANNs have introduced high prediction accuracy within linear and non-linear data sets.
- Clustering: The SOFM is usually used in the clustering task.

- Function approximation: As described before, the QNN is used to approximate value functions in the Q-learning approach.
- Filtering: One of the traditional applications is using the ANNs as data filter in the control field.

The main idea of this thesis depends on the network traffic prediction, therefore, in the next section; the use of ANN in the traffic prediction application will be discussed in more details.

3.6 Traffic Prediction Using ANN Model

The network traffic prediction is classified as a Time Series Forecasting (TSF) problem. In particular, the TSF problem concerns with the systems that have a strong correlation between their chronologically ordered values [94], such as the network traffic. The TSF objective is to model these systems and predict their behavior based in a historical data without the awareness of their internal structure and main functions.

The network traffic has some remarkable characteristics which make the prediction of their future values possible. The main characteristics of network traffic are listed in the following points:

- Highly non-linear nature: The authors of scientific work in [101] demonstrate the evidence of nonlinear nature of network traffic.
- Strong correlation: Network traffic presents a strong correlation between its chronologically ordered values.
- Self-Similarity: This property is defined in [102] as "It is a specific phenomenon where a certain property of an object is preserved with respect to scaling in space and/or time".
- Long Range Dependence: This property is defined in [103] as "It is a behavior of a time-dependent process that shows statistically significant correlations across large time scales".
- Burstiness: This property is defined in [104] as "The intermittent increases and decreases in activity or frequency of an event". This feature can be measured by the ratio of peak rate to mean rate. Also, Bursty events are characterized by heavy tailed distributions.

There are several efforts have been introduced in the traffic prediction research direction. These can be classified into two categories: linear prediction and nonlinear prediction [95]. The previously discussed prediction techniques within the DF-PI [46] and PSA [49] algorithms (See chapter Two) are classified as linear prediction models. The most common example of nonlinear prediction model is the ANN model, that is used to predict the real network traffic in the following recently research works [95], [96], [97], [98], [99] and [100].

The comparative study between the linear and nonlinear predictions in [95], shows that traffic prediction using the ANN model outperforms the traffic prediction using the linear forecasting models. Additionally, based on experiment result in chapter five, our proposed prediction model (ANN-based) outperforms the linear forecasting models within DF-PI and PSA algorithms [24].

On the whole, the ANN model is one of the best proposed tools for modeling and predicting the traffic parameter whereas the ANN has the following properties:

- Simple architecture: ANN achieves an efficient performance with a simple architecture in several research fields.
- Flexibility: ANN is not restricted on a specific system with predefined type of relationship between their parameters.
- Learning capability: The most interested property of ANN over the traditional modeling techniques is the learning capability. Based on the provided historical data, the underlying relationship between system inputs and outputs can be efficiently recognized by ANN without prior knowledge of the system functional form.
- Non-linear modeling capability: ANN has the ability to approximate too many functions regardless of their degree of nonlinearity.

CHAPTER 4 - System model

In the first chapter, the model details of proposed algorithms are presented and discussed. The main objective of this dissertation is to introduce a new optimization mechanism which uses the predicted traffic in order to improve the performance of dynamic routing algorithms. Two different contributions are introduced. The first contribution is enhancing the performance of centralized routing algorithms [22], [23], [24] and the second contribution is developing a new prediction-based decentralized routing algorithm [25], [26], [27].

4.1 Predicting of Future Load-based Routing (PFLR)

This section provides a detailed description for the Predicting of Future Load-based Routing (PFLR) algorithm. The first part of this section discusses the main characteristics of the new innovative idea of PFLR algorithm. The second part describes the model structure, the proposed prediction models and pseudo code for the first version of PFLR algorithm (PFLRv.1). The third part of this section describes the model structure, the new added features and pseudo code for the second version of PFLR algorithm (PFLRv.2). The last part discusses the complexity analysis of PFLR algorithm.

4.1.1 The characteristics of innovative idea

Before going ahead to describe the PFLR algorithm in more details, this section summarizes the main characteristics of the new innovative idea and outlines the new features for this proposed algorithm.

- The proposed approach is routing maintenance algorithm, which can run with any routing algorithm whose computations depend on the residual BW in network links.
- With the use of PFLR algorithm, the future status of the network link loads will be considered. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future and so reduces the occurrence of network congestions and at the same time leads to increase the network utilization.

- The most important feature of PFLR algorithm is the link state (weight) representation. The proposed algorithm combines the predicted link load with the current link load with an effective method in order to optimize the link weights. The idea is to reduce the number of wrong and critical decisions in case of uncertain prediction accuracy.
- The proposed approach uses the Artificial Neural Network (ANN) for building an adaptive traffic predictor in order to predict future link loads. The main reason of using the ANN is that: the ANN is one of the best proposed tools for modeling and predicting the traffic parameters. The ANN has the ability to approximate too many functions regardless of their degree of nonlinearity and without prior knowledge of its functional form. Therefore, ANN can offer an accurate prediction capability (especially in our on-line forecasting case) with different types of network traffic and has the ability to be adaptive. The up-to-date scientific researches that propose and demonstrate the use of ANN for building the traffic predictor are presented in [95], [96], [97], [98], [99] and [100].
- The PFLR algorithm has the ability to adapt the length of prediction step depending on the prediction accuracy in order to efficiently estimate the link traffics and so enhance the routing performance.

4.1.2 The PFLRv.1 approach

Figure 4.1 outlines the operations of PFLRv.1. The white color boxes represent the typical routing components. In the traditional dynamic routing, there are requests for routes among different source and destination pairs. The routing algorithm takes the current information of the network links and computes the best path based on a pre-defined method. After selecting the best path, the routing algorithm forwards the packets through the network and updates the reserved BW of each link that belongs to the best path between the source and the destination.

The dark color boxes are the components of the proposed algorithm. The idea behind design of PFLR algorithm is to consider the future link load to enhance the performance of dynamic routing algorithms. Therefore, a traffic predictor is

proposed in order to accurately predict the traffic behavior. The ANN is used for building an adaptive traffic predictor in order to predict future link loads.

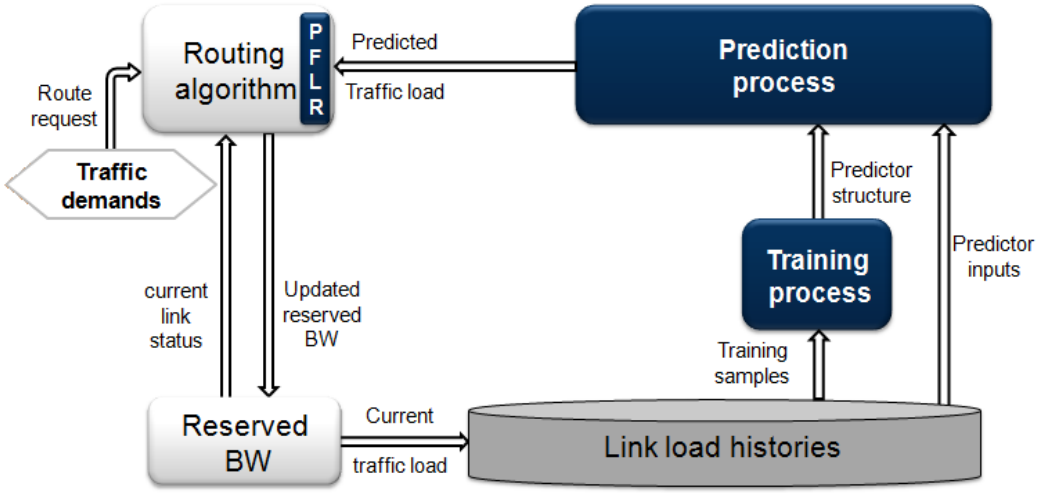

Figure 4.1 Predicting of Future Load-based Routing (PFLRv.1) model.

In the detailed implementation, every link has a predictor which is placed on one of the directly connected nodes. Each predictor works on its link history and has its own parameter values. In other words, the predictions are made decentralized to achieve a fast prediction and to conquer the complexity of prediction.

The proposed predictor has two different processes: the training and prediction processes. In the training process, the internal structure of dynamic FFNN is constructed by a learning process based on the history of link loads. During the prediction process, the future link load on every link is estimated after (and during) a specified period of time, named Window Size (*WS*). During this dissertation, the *WS* parameter (or length of prediction step) is defined as "the number of representative sample steps used to determine the next prediction point".

4.1.2.1 Proposed prediction models

There are two different prediction models are proposed within the PFLRv.1 algorithm. The first is the single step-ahead prediction model and the second is the multi steps-ahead prediction model. Each prediction model has different training and prediction processes. In the next sub sections, the description of each model will be introduced in more details.

4.1.2.1.1 The single step-ahead prediction model

The single step-ahead model is considered as an event-based approach. In the event-based approach, if a new route is requested in the network, a new event is generated. In general, the network traffic is sampled at every new route request. Therefore, the lengths of sample steps are different and the length of each step equals to the time difference between the occurrences of current and previous route requests.

In the single step-ahead prediction model, the structure of used dynamic FFNN is shown in Figure 4.2. It consists of three layers: The input layer which contains sixteen neurons; the hidden layer which contains 20 neurons and only one neuron in the output. The Levenberg-Marquardt [105] training algorithm is used because it is the fastest and most accurate one in this case. A lot of experiments are made with various numbers of input neurons, various hidden layer numbers and various neurons in the hidden layer. After that, the best structure of FFNN, which achieves the best training results, is selected. Additionally, different values of training period size are tested to achieve an efficient predictor.

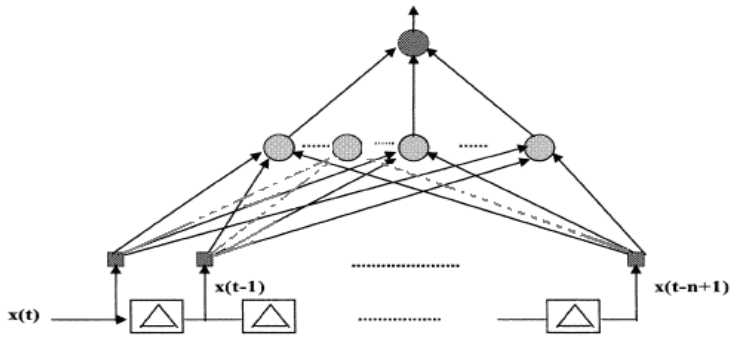

Figure 4.2 Dynamic FFNN architecture (Single step-ahead model).

During the training process, a history of the last thousand (plus *WS*) of sampled link traffic are used for training purpose. One training pattern contains sixteen sampled traffic values from the history in row as input values and one expected output value. The expected output value is a history value *WS* time after the input values. By shifting, one thousand training patterns are generated (see Figure 4.3). Additionally, the training process is triggered every a specific fixed period (e.g. a hundred traffic samples).

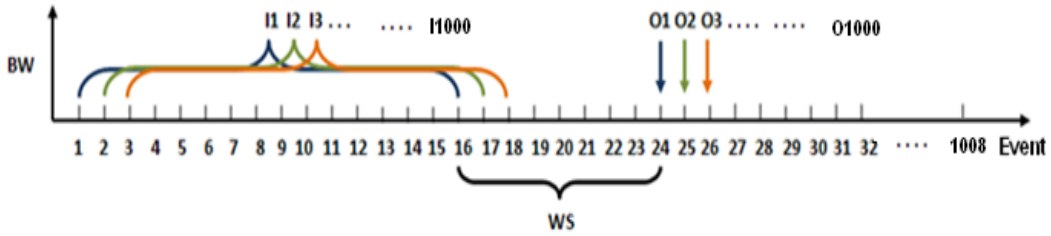

Figure 4.3 The training process (PFLR algorithm - Single step-ahead model).

Figure 4.4 demonstrates the details of prediction process. In the prediction process within the single step-ahead prediction model, the last sixteen traffic values are used as inputs for the FFNN. Then, the FFNN predicts a value for the link load after a *WS* period. In this model, the prediction process is triggered every *WS* period. In the first version of PFLR, an analysis study is done to select the best value of *WS*. In other words, *WS* period has fixed value during the whole simulation time.

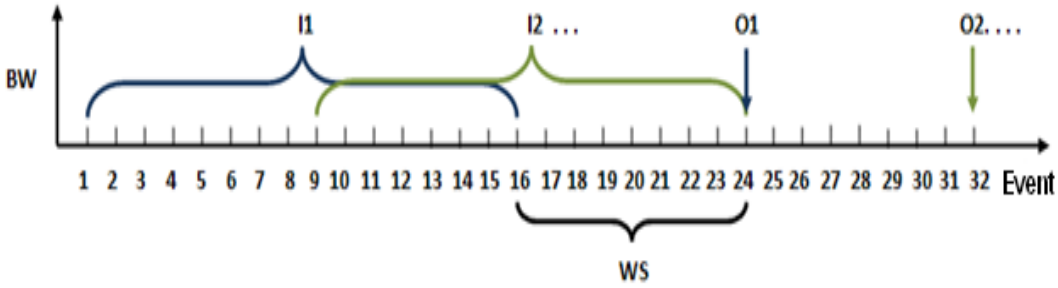

Figure 4.4 The prediction process (PFLR algorithm).

4.1.2.1.2 The multi steps-ahead prediction model

In the single step-ahead prediction model, the network traffic is sampled at every new route request. However, the new route request changes only the reserved BW of the links within selected path between the source and destination pair. This means that, if the traffic values within each network link at every new route request are considered, it will cause a lot of data redundancy.

In order to clarify the previous situation, an illustrative example, that shows the changes of reserved BW within a specific link, is considered. Figure 4.5 shows the reserved BW graph within a specific link. In this figure, the X axis represents the time units (the red labels). At the same time, it shows the occurrences time of consequence route requests (the black labels). Additionally, the Y axis represents the reserved BW values.

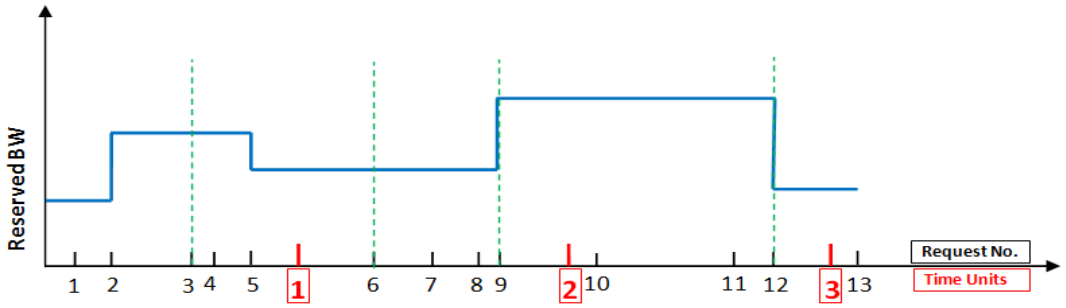

Figure 4.5 The sample step representation.

In this illustrative example, the considered link only was a part of the route requests no. 2 and 9. Therefore, the reserved BW within this link is increased at the time of route requests no. 2 and 9. Additionally, at the occurrence time of route request no. 5 and 12, the holding time for other rout requests are finished. Therefore, the reserved BW within this link is decreased at the occurrence time of route request no. 5 and 12. However, the reserved BW values are fixed at the occurrence time of all other route requests, because the considered link is not part of their selected paths.

In single step-ahead prediction model, the network traffic is sampled at every new route request. Thus, the sampled traffic data contains a lot of repeated data. To overcome the data redundancy problem, there is another method is proposed to represent the effective sample step. The new proposed method is to use the mean of inter-update intervals to represent the effective sample step.

The inter-update interval term represents the number of consequence requests that occurred between two successive changes (updates) of the reserved BW. Considering the data within Figure 4.5, there are four changes at the occurrence time of requests no. 2, 5, 9 and 12. Thus, the inter-update intervals for the four changes in the reserved BW are two, three, four and three requests. This means that, the mean of inter-update intervals equals to three requests. The green ditched lines within Figure 4.5 represent the proposed effective sample steps.

The second proposed feature in the multi steps-ahead prediction model is the prediction of link load values at multiple points in the future. In the single step-ahead prediction model, the future link load information is represented by the predicted link load value at the end of *WS* period. However, in the multi steps-

ahead prediction model, the future link load information is represented by the average of all predicted link load values during the *WS* period.

In multi steps-ahead prediction model, the structure of the used dynamic FFNN is shown in Figure 4.6. It consists of three layers: The input layer which contains seventy six neurons; the hidden layer which contains thirty five neurons. The number of neurons within the output layer equals to the used *WS* value in the considered performance test.

Also, the Levenberg-Marquardt [105] training algorithm is used. A lot of experiments are made with various numbers of input neurons, various hidden layer numbers and various neurons in the hidden layer. After that, the best structure of FFNN, which achieves the best training results, is selected. Additionally, different values of training period size are tested to achieve an efficient predictor.

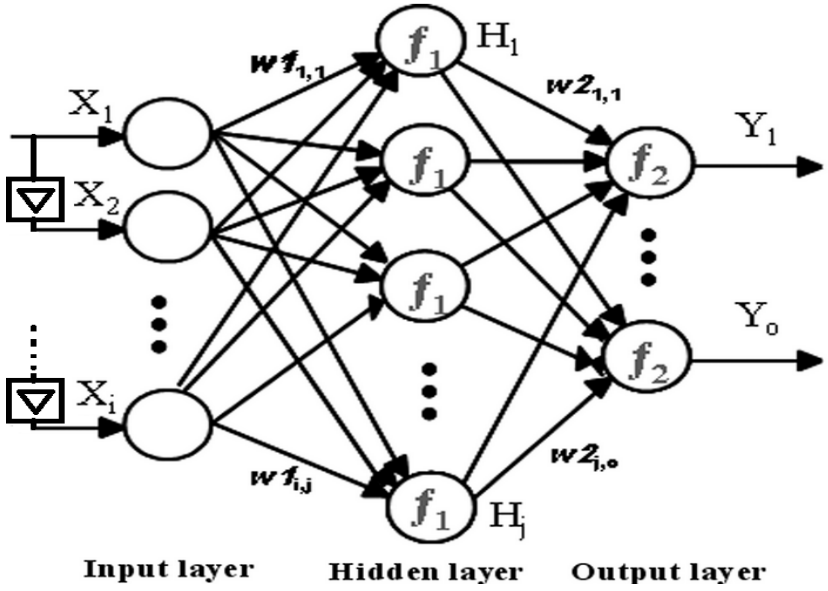

Figure 4.6 Dynamic FFNN architecture (Multi steps-ahead model).

During the training process, the histories of the last ten thousands of effective sampled link traffic are used for training purpose. One training pattern contains seventy six effective sampled traffic values from the history in row as input values and *WS* expected output values. The expected output values are the effective sampled link values within the *WS* time period after the input values. By shifting, ten thousands (minus *WS* values) training patterns are generated (see Figure 4.7).

According to the prediction process within the multi steps-ahead prediction models, the last seventy six traffic values are used as inputs for the FFNN. Then, the FFNN predicts all values for the link load during *WS* period. Also, in this model, the prediction process is triggered every *WS* period.

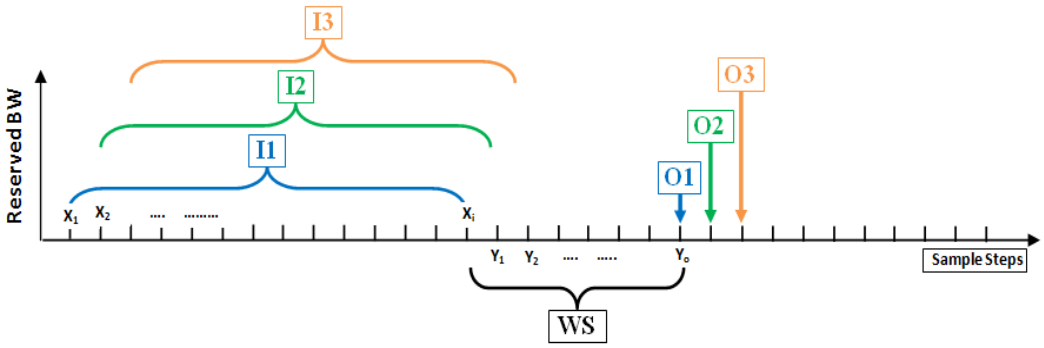

Figure 4.7 The training process (PFLR algorithm - Multi steps-ahead model).

4.1.2.2 PFLRv.1 pseudo code

The main core of PFLRv.1 algorithm is divided into four stages. The first stage is the prediction of the future traffic load on every link after (or during) *WS* period. The predicted available BW is calculated in the second stage using the next equation:

The predicted available BW = link capacity – Predicted Load Equation (4.1)

In the third stage, the predicted available BW and current residual BW of each link are combined to represent the reciprocal of available BW (*RBW*) using the following formula:

$$RBW = \frac{(1-\alpha)}{Available\ BW} + \frac{\alpha}{Predicted\ available\ BW} \qquad \text{Equation (4.2)}$$

The *RBW* formula is controlled by a parameter, named α, which represents a weight for the predicted value. A low α reduces the influence of the predicted value on the BW. A high value of α increases the influence and suppresses the current value of available BW.

Finally, the normal (unchanged) routing algorithm runs to compute a weight-optimized path. In other words, PFLR algorithm does not change the routing algorithm that is already running in the network; it just modifies the links weights by taking the future link load into account. Thus the proposed algorithm has the

capability to run with any dynamic routing algorithm which depends on the residual BW during its computation.

Algorithm 4.1: PFLRv.1 algorithm.

Input:	The network topology and residual available BW. The route requests between the ingress-egress pairs.
Output:	Routed paths through the network.

Algorithm:

1) Repeat the following steps for a specific fixed period (e.g. a hundred traffic samples).
   a) Predict the future traffic load on all network links after a *WS* period.
   b) Obtain the predicted available BW in each link using Equation (4.1).
   c) Repeat the following steps until the time of the *WS* elapses.
      i. Compute the reciprocal of available BW using equation (4.2).
      ii. Compute the best path using the normal routing algorithm without changing anything.
2) Train the predictor according to the link load histories.
3) Go to step 1.

In the PFLRv.1 algorithm, the training process is triggered every a specific fixed period (e.g. a hundred traffic samples).

4.1.3 The PFLRv.2 approach

The new proposed feature of the PFLRv.2 algorithm is the parameters adaptation process (see Figure 4.8). Two parameters are adapted and optimized depending on the prediction accuracy: The *WS* parameter and the Prediction Validity Period (*PVP*) that represents the duration of period for which the prediction is valid with a high degree of confidence. In the PFLRv.2 algorithm, the aim is to give the proposed algorithm the ability to adapt the prediction parameters in order to efficiently predict the traffic load and optimize the routing performance.

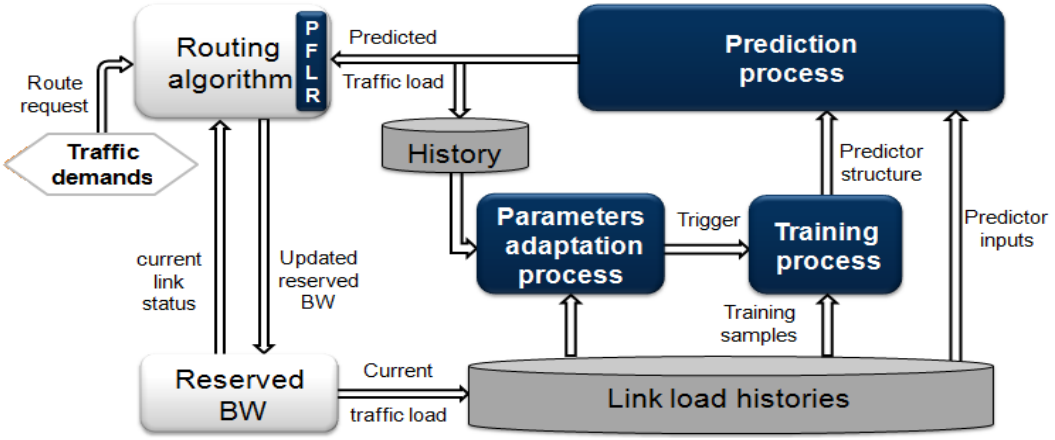

Figure 4.8 Predicting of Future Load-based Routing (PFLRv.2) model.

4.1.3.1 The new features of PFLRv.2 algorithm

In the PFLRv.2 algorithm, a new adaptive feature is proposed, called parameters adaptation process. The main objective for this process is to give the proposed predictor the ability to optimize the prediction parameters such as *WS* and *PVP* parameters. As described before, the prediction takes place every *WS* period and the predictor structure is not changed until *PVP* period has elapsed. A *PVP* period contains multiple *WS* periods. In other words, *PVP* represents how many times the prediction is happened. If the *PN* parameter is the Prediction Numbers, then:

$$PVP = WS * PN \qquad \text{Equation (4.3)}$$

The parameters adaptation process depends on the predictions accuracy that is calculated by comparing the actual and predicted traffic loads. Therefore, two archiving processes are required to archive the actual and predicted traffic loads during the run of algorithm. The prediction accuracy can be represented by the prediction error. There are many error representation methods [106]. In this work, the Root Mean Square Error (*RMSE*) is used to represent the prediction accuracy. If *AL* is the actual traffic load and *PL* is the predicted traffic load, then the *RMSE* value is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(AL_i - PL_i)^2}{N}} \qquad \text{Equation (4.4)}$$

The relationship between the *RMSE* for the prediction and the *WS* parameter is studied in order to correctly adjust the *WS* parameter depending on the *RMSE* value.

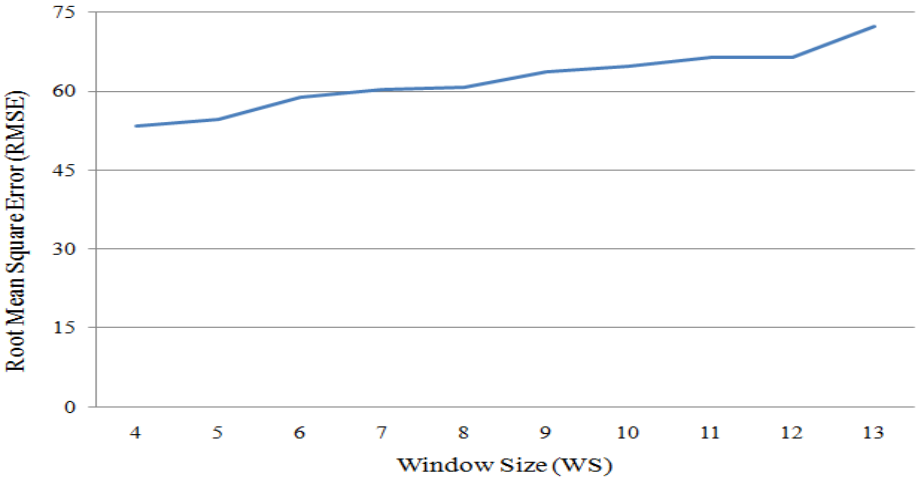

Figure 4.9 The relationship between the *RMSE* and *WS* parameters.

Figure 4.9 shows the relationship between the *RMSE* and *WS* parameter with respect to the WSP_PFLR (WSP algorithm with PFLR modification) algorithm. In this relationship study, the WSP_PFLR algorithm runs within the MIRA topology [36] and with respect to generated traffic. During this experiment, different *WS* values are tested and the *RMSE* for the prediction is measured. The results show that, the *RMSE* increases when the *WS* parameter increases. Therefore, the algorithm aims to decrease the *WS* parameter when the *RMSE* value increases.

The parameters adaptation procedure consists of four steps. The first step is the computation of *RMSE* using Equation (4.4). In the second step, the *PN* parameter is adjusted based on the comparison between the *RMSE* and the Error Threshold (*ETh*) parameters (see Figure 4.10). For example, if *RMSE* value equals to or less than *Eth* value, this means that the prediction accuracy is very good and the number of predictions should be increased by two.

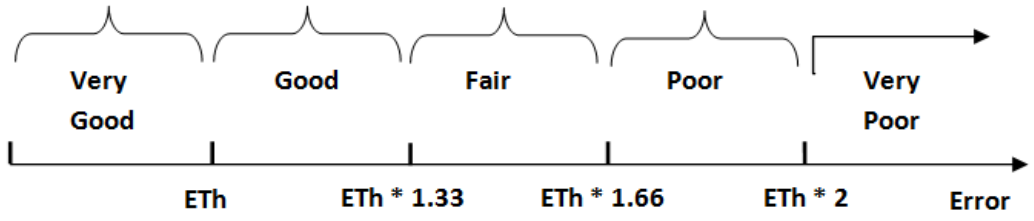

Figure 4.10 The *PN* parameter adaptation.

In the third step, the procedure compares PN_{n+1} and PN_{Base} parameters where PN_{Base} contains the last value of *PN* when *WS* is changed. If PN_{Base} is increased by

PNTh, then *WS* should be incremented by one. If PN_{Base} is decreased by *PNTh,* then WS should be decremented by one.

Algorithm 4.2: Parameters adaptation procedure.

1) Compute the *RMSE* of prediction using Equation (4.4).
2) Update the *PN* parameter with respect to the following comparisons:
   a) If $RMSE \leq ETh$, $PN_{n+1}=PN_n+2$.
   b) If $RMSE>Eth$ & $RMSE \leq ETh*1.33$, $PN_{n+1}=PN_n+1$.
   c) If $RMSE>ETh*1.66$ & $RMSE \leq ETh*2$, $PN_{n+1}=PN_n-1$.
   d) If $RMSE>ETh*2$, $PN_{n+1}=PN_n-2$.
3) Update the *WS* parameter with respect to the following comparisons:
   a) If $PN_{n+1} \geq PN_{Base}+PNTh$, $WS=WS+1$, $PN_{Base}=PN_{n+1}$.
   b) If $PN_{n+1} \geq PN_{Base}-PNTh$, $WS=WS-1$, $PN_{Base}=PN_{n+1}$.
4) Compute the new *PVP* value, $PVP=WS * PN_{n+1}$.

The last procedure step is calculating the new value of *PVP* parameter using Equation (4.3). Figure 4.11 shows an example that demonstrates how to adapt *WS* parameter depending on the comparison between the PN_{n+1}andPN_{Base} parameters.

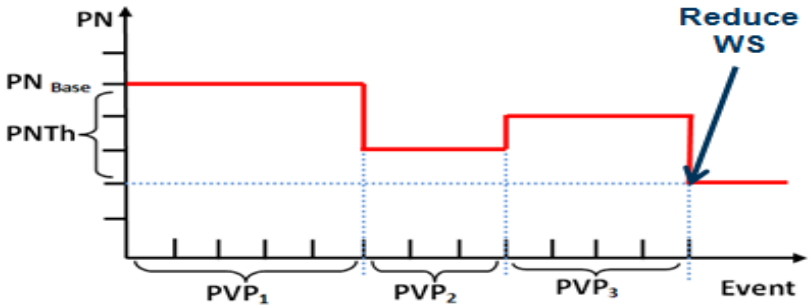

Figure 4.11 The *WS* parameter adaptation.

4.1.3.2 PFLRv.2 pseudo code

In the PFLRv.1 algorithm, the training process is triggered every a specific fixed period (e.g. a hundred traffic samples). In the PFLRv.2 algorithm, the training process is triggered every *PVP* period which is adapted depending on the

prediction accuracy. When the time of *PVP* has elapsed, the parameters adaptation procedure will be called to adapt the *WS* and *PVP* parameters. Then the training process will be triggered.

Algorithm 4.3: PFLRv.2 algorithm.

Input:	The network topology and residual available BW. The route requests between the ingress-egress pairs.
Output:	Routed paths through the network.

Algorithm:

1) Repeat the following steps until the time of the *PVP* period has elapsed.
   a) Predict the future traffic load on all network links after a *WS* period.
   b) Obtain the predicted available BW in each link using Equation (4.1).
   c) Repeat the following steps until the time of the *WS* has elapsed.
      i. Compute the reciprocal of available BW using equation (4.2).
      ii. Compute the best path using the normal routing algorithm.
2) Call parameters adaptation procedure (Algorithm 4.2) to adapt *WS* and *PVP*.
2) Train the predictor according to the link load histories.
3) Go to step 1.

4.1.4 Complexity analysis of PFLR algorithm

The PFLR algorithm requires additional computational time to achieve an enhanced routing performance. This time consists of two parts, the training time and the prediction time of the predictors. As mentioned before, the predictors are distributed on the nodes. Therefore, every node is responsible for an average of *(|E|/|V|)* operation. Where *|E|* is the number of links and *|V|* is the number of nodes. The training operation happens only one time every *PVP* period. The prediction also happens every *WS* period. Thus, the training requires *O ((|E| Tt) / (|V| PVP))* time steps, where *Tt* is the training time of one predictor and the prediction operation requires *O ((|E| Pt) / (|V| WS))* time steps, where *Pt* is the prediction time of one predictor.

4.2 Prediction-based Decentralized Routing (PDR)

This section provides a detailed description for the Prediction-based Decentralized Routing algorithm (PDR). The first part of this section discusses the main characteristics of the new innovative idea of PDR algorithm. The second part describes the model structure of PDR algorithm. The third part of this section describes the PDR algorithm methodology. The forth part of this section describes the proposed prediction model and pseudo code for the first version of PDR algorithm (PDRv.1). The fifth part of this section describes the proposed prediction model, the new features and pseudo code for the second version of PDR algorithm (PDRv.2). The sixth part discusses the parameter adaptation process which exists in both PDR algorithm versions. The last part discusses the complexity analysis of PDR algorithm.

4.2.1 The characteristics of innovative idea

Before going ahead to describe the proposed algorithm in more details, this section summarizes the main characteristics of the new innovative idea and outlines the new features for this proposed routing algorithm:

- The proposed approach is fully decentralized and self-organized approach and is based on the Ant Colony Optimization (ACO) technique.
- In this approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters.
- The use of link state information helps the routing algorithm to efficiently achieve the BW guarantee of the provided QoS. Additionally, considering the future value of the network link loads leads to reduce the interference between the reserved requests in the future and so reduce the occurrence of network congestions and increases the network utilization.
- PDR algorithm uses an efficient ant's selection methods (for the intermediate nodes) which considers the predicted link load to better estimate for the congestion within network links. This feature gives the

ability to efficiently distribute the ants on the network topology and accurately discover the best paths.

- The PDR algorithm uses similar prediction mechanism to the PFLR algorithm but with local-based implementation.
- Additionally, the PDR algorithm has the ability to locally adapt to the internal algorithm parameters, such as the prediction validity period, in order to efficiently predict the link traffics and so effectively enhance the routing performance.

4.2.2 The PDR model

Figure 4.12 outlines the operations of PDR. In the algorithm, ants are distributed through the network to discover the best paths. The ants use a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit. This is simpler and requires less control parameters. After selecting the best path, the routing algorithm forwards the packets through the network and updates the reserved BW of each link that belongs to the best path between the source and the destination.

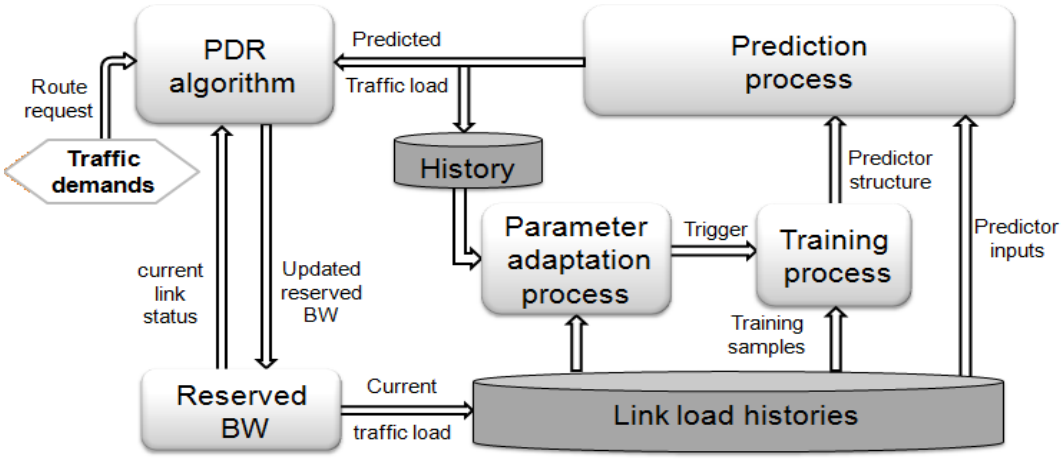

Figure 4.12 Prediction-based Decentralized Routing (PDR) algorithm.

The idea behind design of PDR algorithm is similar to the idea of PFLR algorithm which depends on the consideration of the future link load to enhance the performance of Ant-based routing algorithms. Therefore, a traffic predictor is proposed to accurately predict the traffic behavior. The ANN is used for building an adaptive traffic predictor in order to predict future link loads.

The proposed predictor has two different processes: the training, and the prediction process. In the training process, the internal structure of FFNN is constructed by a training based on traffic samples of link histories. During the prediction processes, the future link load on every link is estimated after *WS* period. In the detailed implementation, each link has a predictor which is placed on one of the directly connected nodes. Each predictor works on its link history and has its own parameter values. In other words, the predictions are made decentralized to achieve a fast prediction and to conquer the complexity of prediction.

In the parameter adaptation process, The Prediction Validity Period (*PVP*) parameter is adapted and self-optimized depending on the prediction accuracy. The *PVP* parameter represents the duration of a period for which the prediction is valid with a high degree of confidence. With the help of this feature, the training of each predictor is triggered independently of each other.

4.2.3 The PDR methodology

The PDR algorithm is built on the principles of the TB routing framework. In the TB design (see Figure 2.2), each router has two tables: a link probability table *Pt* and an average transmission delay table *avg*. *Pt* contains *m* rows, one for each destination node. Each row has *K* entries, one for each outgoing link of the router. The entry *pt [d,i]* is the probability of sending a packet to destination d on the outgoing link *i*. The table *avg* has *m* entries, one for each destination node. The entry *avg (d)* is the average transmission delay from the current node to the destination *d*, which is computed from the last *M* scout packets that arrived from node *d*. A scout packet is sent from the source to the destination to explore the network. At every intermediate node, the scout packet selects the outgoing link randomly or according to various probabilities that will be described later. When scout packets find their destination, they return to their source on the same path they have arrived on and update their accumulated latency *td* in every intermediate node by $td = td + t(i)$, where $t(i)$ is the current latency of the outgoing link *i*. Then, the scout packets use the accumulated latency *td* to update the *pt* as follows:

$$f(td) = max\left(min\left(\frac{avg(d)}{td}, 10\right), 0.1\right) \quad \text{Equation (4.5)}$$

$$\Delta p = \delta \times f(td) \quad \text{Equation (4.6)}$$

$$pt[d,i] = \frac{(pt[d,i] + \Delta p)}{(1 + \Delta p)} \quad \text{Equation (4.7)}$$

$$pt[d,j]_{j \neq i} = \frac{(pt[d,j])}{(1 + \Delta p)} \quad \text{Equation (4.8)}$$

The average latency *avg (td)* is used to scale the positive reinforcement value of the scout packet. A larger value of *f (td)* indicates a better (shorter) path. *f(td)* is limited to the range [0.1,10] to prevent wide fluctuations in Δp, which is the reinforcement value of *pt[d, j]*.The δ parameter defines the learning rate of the algorithm. All entries in *Pt* of the same destination *d* are scaled by $1+\Delta p$ to ensure that their sum remains equal to one.

In this approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters. The current latency *t (i)* of an outgoing link *i* in the TB algorithm is replaced by the Link Weight formula *LW (i). LW (i)* represents a combination of PFLR and LIOA to reduce the interference among competing flows by balancing the number of flows and the required BW reserved by a link to achieve efficient routing.

The LIOA algorithm represents a cost metric which balances the number and the intensity of the flows offered to the routes. In the LIOA algorithm, the link weight $LW(i) = I^{lc} / (Available\ BW)^{(1-lc)}$, whereas *I* is the number of flows carried on the link and *lc* is the least interference control parameter which represents a trade-off between the number and the magnitude of the flows traversing a link. On the other hand, the PFLR algorithm proposes to incorporate the Predicted Available BW (*PABW*) in the link weight formula to optimize the performance of routing. Based on the previous considerations, the formula for calculating *LW (i)* is:

$$LW(i) = I^{lc}\left(\frac{(1-\alpha)}{(Available\ BW)^{(1-lc)}} + \frac{\alpha}{(PABW)^{(1-lc)}}\right) \quad \text{Equation (4.9)}$$

The *LW (i)* formula is controlled by a parameter α, which represents the prediction weight. A low α reduces the influence of the predicted value on the available BW. A high value of α increases the influence and suppresses the current value of the available BW.

On the other hand, when a node receives a data packet, which needs to be forwarded, data packets will be routed according to the probabilities in the *pt* table that is maintained by the PDR algorithm.

4.2.4 The PDRv.1 approach

In this section, the proposed prediction model and pseudo code for the PDRv.1 approach are presented and discussed in more details.

4.2.4.1 Proposed prediction model

In the PDRv1 algorithm, the static FFNN is used. The structure of the used static FFNN is shown in Figure 4.13. It consists of three layers: The input layer which contains three neurons; the hidden layer which contains fifteen neurons and only one neuron in the output. The Levenberg-Marquardt [105] training algorithm is used because it is the fastest and most accurate one in this case. Different FFNN design and different values of training period size are tested to achieve an efficient predictor. In contrast to the training process in the previous version of PDR algorithm that is event-based, the training process in PDR algorithm is time-based.

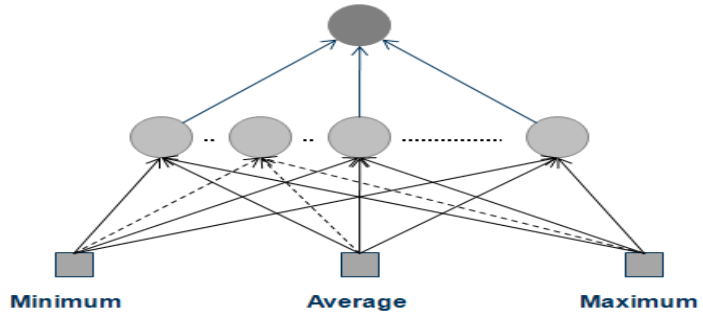

Figure 4.13 Static feed forward neural network architecture.

In the event-based approach, if a new path is requested in the network, a new event is generated. During the training process of PFLR algorithm, a history of the last thousand events (plus *WS*) of the link traffic values is used for training

purpose. However in the training process of PDRv.1 algorithm, a history of the last hundred time units of the link traffic values is used for training purpose. One training pattern contains the minimum, maximum and average of traffic during a time unit. This pattern is formed in a row as input values and one expected output value. The expected output value is a history value *WS* time after the input values. By shifting, one hundred of training patterns are generated. In PDRv.1 algorithm, the training process is triggered every *PVP* period which is adapted depending on the prediction accuracy.

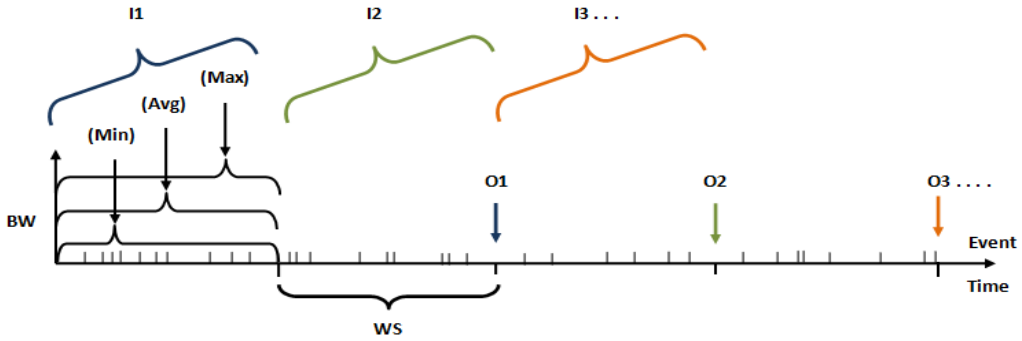

Figure 4.14 The training process (PDRv.1 algorithm).

In the prediction process of PDRv.1 algorithm, the minimum, maximum and average of the traffic during the last time unit are used as inputs for the static FFNN. Then, the static FFNN predicts a value for the link load after a *WS* period.

The prediction process is triggered every *WS* period. In other words, the prediction happens every *WS* period and the predictor structure is not changed until *PVP* period has elapsed. An analysis study is done to select the best value of *WS* parameter.

4.2.4.2 PDRv.1 pseudo code

The following table lists and describes the internal steps of PDRv.1 algorithm.

Algorithm 4.4: PDRv.1 algorithm.	
Input:	The network topology and residual available BW. The route requests between the ingress-egress pairs.
Output:	Routed paths through the network.

Algorithm:

1) Repeat the following steps until the time of the *PVP* elapses.
   a) At regular intervals of *WS*, predict the available BW on all links in the network after a specified *WS*.
   b) At regular intervals of *N*, each node generates and sends an ant to a destination.
   c) When a node receives an ant:
      i. It will forward the ant and selects the next link for the ant´s route randomly.
      ii. The ant never selects an outgoing link that leads to a node that has been visited earlier in its path (a loop). If there is no such outgoing link, the ant will die.
   d) When the current node is the destination, then, the ant will return to the source on the same path on which it has arrived.
   e) At each intermediate node :
      i. Compute *LW (i)* of the outgoing link *i* on every link in the backward path using Equation (4.9).
      ii. Compute *td*, *td=td+ LW (i).*
      iii. Update the *pt* and *avg* tables using Equations (4.5, 4.6, 4.7 and 4.8).
3) Call the parameter adaptation procedure to adapt the *PVP* parameter (See Algorithm 4.6).
2) Train the predictor on the link load histories.
3) Go to step 1.

4.2.5 The PDRv.2 approach

In this section, the proposed prediction model, the new features and pseudo code for the PDRv.2 approach are presented and discussed in more details.

4.2.5.1 Proposed prediction model

In the PDRv2 algorithm, the dynamic FFNN is used. The used dynamic FFNN consists of three layers: The input layer contains eight neurons; the hidden layer contains 15 neurons and only one output neuron (See Figure 4.3). The training process within PDRv.2 algorithm has different procedure than the PDRv.1 algorithm.

During the training process within PDRv.2 algorithm, a history of the last hundred time units of link traffic values is used for training purpose. The traffic value is saved in the history at each quarter of time unit. One training pattern contains eight traffic values (during two time units) from the history in row as input values and one expected output value. The expected output value is the traffic value after *WS* time from the last input. By shifting, four hundreds of training patterns are generated. This process is triggered every *PVP* period which is adapted depending on the prediction accuracy.

In the prediction process of PDRv.2 algorithm, the last eight traffic values are used as inputs for the dynamic FFNN. Then, the dynamic FFNN predicts a value for the link load after *WS* period.

4.2.5.2 The new features of PDRv.2 algorithm

In the PDRv.1 algorithm, all network links are considered in the selection of best paths between the source and destination pairs. Additionally, the scout packet selects the outgoing link randomly at every intermediate node. While in the PDRv.2 algorithm, there are two proposed features of Ant-based mechanism in order to enhance the performance of PDR algorithm.

There are two proposed new features of Ant-based mechanism are incorporated in PDRv.2 algorithm. The first new feature is that, PDRv.2 algorithm burns the links that do not have enough available BW in order to serve the next traffic demand. The second new feature is the ant's selection method for the intermediate nodes in the discovered paths. In the PDRv.2 algorithm, there are three selection methods:

1) The random method: the ant selects randomly the next hop.
2) The best method: the ant check *Pt* table and selects the outgoing link that has the highest *Pt* entity. The idea is to measure the quality of best paths.
3) The exploratory method: the ant uses a combination of the best path information and the congestion of network link in order to select the next hop. This ant selects the outgoing link that has the maximum value of the following formula:

$$Prob\_exp[d,i] = pt[d,i] + (\theta \times cn[d,i]) \quad \text{Equation (4.10)}$$

The congestion of network link *cn[d,i]* is computed according to the following formula:

$$cn[d,i] = \frac{LW(i)}{\sum_{j=1}^{S} LW(j)} \quad \text{Equation (4.11)}$$

The *S* parameter is the number of outgoing links and θ is the weight of the congestion term.

The probability for choosing one from the three selection methods in PDRv.2 algorithm is uniformly distributed.

4.2.5.3 PDRv.2 pseudo code

The following table lists and describes the internal steps of PDRv.2 algorithm.

Algorithm 4.5: PDRv.2 algorithm.

Input:	The network topology and residual available BW. The route requests between the ingress-egress pairs.
Output:	Routed paths through the network.

Algorithm:

1) Repeat the following steps until the time of the *PVP* elapses.
   a) At regular intervals of *WS*, predict the available BW on all links in the network after a specified *WS*.
   b) At regular intervals of *N*, each node generates and sends an ant to a destination.

c) Burn all network links that have not enough available BW to serve the next request demand.

d) When a node receives an ant:

   i. It will forward the ant and selects the next link for the ant´s route according to one from the three selection methods that are discussed in details in section 4.5.2.1.

   ii. The ant never selects an outgoing link that leads to a node that has been visited earlier in its path (a loop). If there is no such outgoing link, the ant will die.

e) When the current node is the destination, then, the ant will return to the source on the same path on which it has arrived.

f) At each intermediate node :

   i. Compute *LW (i)* of the outgoing link i on every link in the backward path using Equation (4.9).

   ii. Compute *td*, *td=td+ LW (i).*

   iii. Update the *pt* and *avg* tables using Equations (4.5, 4.6, 4.7 and 4.8).

4) Call the parameter adaptation procedure to adapt the *PVP* parameter (See Algorithm 4.6).

5) Train the predictor on the link load histories.

6) Go to step 1.

4.2.6 Parameter adaptation process

In the both PDR algorithm versions, there is common process called parameter adaptation process. The main objective for adaptation process is to give the predictor the ability to optimize the *PVP* parameter. A *PVP* parameter contains multiple WS periods to represent how many times the prediction is done.

The parameter adaptation process depends on the predictions accuracy that is calculated by comparing the actual and predicted traffic loads. Therefore, two archiving processes are required to archive the actual and predicted traffic loads

during the run of algorithm. The prediction accuracy can be represented by the prediction error. There are different error representation methods. In this work, the *RMSE* is used to represent the prediction accuracy. During this process, the *PN* parameter is adjusted depending on the comparison between the *RMSE* and *ETh* parameters. Then, the *PVP* parameter is updated.

Algorithm 4.6: Parameters adaptation procedure.

1) Compute the *RMSE* of prediction using Equation (4.4).
2) Update the *PN* parameter with respect to the following comparisons:
   a) If $RMSE \leq ETh$, $PN_{n+1}=PN_n+2$.
   b) If $RMSE>Eth$ & $RMSE \leq ETh*1.33$, $PN_{n+1}=PN_n+1$.
   c) If $RMSE>ETh*1.66$ & $RMSE \leq ETh*2$, $PN_{n+1}=PN_n-1$.
   d) If $RMSE>ETh*2$, $PN_{n+1}=PN_n-2$.
3) Compute the new *PVP* value, $PVP=WS* PNn+1$.

The parameters adaptation procedure consists of three steps. The first step is the computation of *RMSE* using Equation (4.4). In the second step, *PN* parameter is adjusted based on the comparison between the *RMSE* and the Error Threshold (*ETh*) parameters. For example, if *RMSE* value is equal to or less than *Eth* value, this means that the prediction accuracy is very good and the number of predictions should be increased by two. The last procedure step is calculating the new value of *PVP* parameter using Equation (4.3).

4.2.7 Complexity analysis of PDR algorithm

In general, the communication overhead of PDR algorithm is low because the scout packets are sent every too many data packets. The PDR algorithm requires additional computational time to enhance the performance of ant-based routing algorithm. As mentioned before, this time consists of two parts, the training time and the prediction time of the predictors. As described in section 4.1.4, the training operation requires $O((|E| Tt) / (|V| PVP))$ time steps and the prediction operation requires $O((|E| Pt) / (|V| WS))$ time steps.

CHAPTER 5 - Simulation results

During this chapter, the performance of the proposed algorithms is presented and the experimental results are discussed. In the first section, the "performance study" term is defined. Additionally, the details of the following simulation environment are discussed with high concern.

In the second section, the performance of PFLRv.1 algorithm (using the single step-ahead prediction model) is tested and the enhancement of the dynamic routing performance is demonstrated. Additionally, the experimental results are presented with respect to different performance criteria and under different network load scenarios. In the third section, the performance of PFLRv.1 algorithm (using the multi steps-ahead prediction model) is tested and the enhancement of the dynamic routing performance is demonstrated.

In the forth section, the performance of PFLRv.1 and PFLRv.2 is compared with each other and the effect of the new proposed features in the PFLRv.2 is presented. In the fifth section, a comparative study between PFLRv.2 algorithm and various estimation-based routing algorithms is presented.

The sixth section focuses on the decentralized routing algorithms and the performance of various versions of PDR and two ACR algorithms is compared. The experimental result, within this section, shows that the PDR algorithm outperforms the traditional ACR algorithms with respect to different network load scenarios. In the last section, the performance of various versions of PDR and two centralized routing algorithms is compared. The experimental result, within this section, shows that the PDRv.2 algorithm outperforms the comparative centralized routing algorithms.

5.1 Performance studies

In this chapter, the "performance study" term means the experiments that target to test and evaluate the performance of routing algorithms. Furthermore, the "analysis study" term means the experiments that target to select the parameters values of proposed algorithms which achieve the best routing performance. All the experiments are implemented using Microsoft Visual Studio [107] and the ANN toolbox in MATLAB [108].

Table 5.1 summarizes the details of the following performance studies and shows the properties of every performance study. In section 5.2, the performance of PFLRv.1 algorithm (using the single step-ahead prediction model) is tested.
In section 5.3, the performance of PFLRv.1 algorithm (using the multi steps-ahead prediction model) is tested. Section 5.4 demonstrates the performance enhancement of PFLRv.2 algorithm compared to the PFLRv.1 algorithm. Section 5.5 presents a comparative study between the PFLRv.2 algorithm and various estimation-based routing algorithms. Section 5.6 focuses on the comparison between PDR algorithm and ACR algorithms. Section 5.7 focuses on the comparison between PDR algorithm and centralized routing algorithms.
In section 5.1.1, the network topologies, which are considered during these performance studies, are described in details. The details of the used generated and real traffics are discussed in section 5.1.2. Section 5.1.3 gives a short overview about the compared routing algorithms for each performance study. In section 5.1.4, the definitions of the measured parameters are defined.

Table 5.1 The details of performance studies.

Details	The performance studies					
	PFLRv.1 algorithm		PFLRv.2 algorithm		PDR algorithm	
	(Single step-ahead model)	(Multi steps-ahead model)	Vs. PFLRv.1 algorithms	Vs. estimation-based algorithms	Vs. ACR algorithms	Vs. Centralized algorithms
Sec. No.	5.2	5.3	5.4	5.5	5.6	5.7
Network topology	MIRA COST266bt	MIRA GÉANT	MIRA Internet2	MIRA Internet2	MIRA Internet2	MIRA Internet2
Traffic types	Generated	Generated Real	Generated Real	Generated Real	Generated Real	Generated Real
Load scenarios	Moderate Heavy	Moderate Heavy	Moderate Heavy	Moderate Heavy	Moderate Heavy	Moderate Heavy
Compared algorithms	WSP CSPF	WSP LIOA	WSP LIOA	DF-PI PSA	AntNet TB	CSPF LIOA
Approach	Centralized	Centralized	Centralized	Centralized	Decentralized	Centralized Decentralized

5.1.1 Network topologies

Four network topologies are considered during the following performance studies. The first is the MIRA network topology, which was used to evaluate the MIRA algorithm [36]. The second is the COST266bt network topology [109]. The third is a real network topology named Internet2 [110]. The forth is a real network topology named GÉANT [111]. Table 5.2 presents the properties of network topologies. Additionally, all network topologies are described in more details in the next subsections.

Table 5.2 The properties of network topologies.

Network topology	No. of nodes	No. of links	Average of node degree
MIRA	15	28	3.73
COST266bt	28	41	3.54
Internet2	9	13	2.89
GÉANT	20	32	3.2

According to the experimental results of the performance study in section 5.2, the PFLRv.1 algorithm enhances the routing performance within both of MIRA and COST266bt network topologies based on generated traffic demands. Therefore, only one network topology, MIRA, is selected to be considered with respect to the generated traffic demands in all performance studies.

In general, all proposed algorithms, PFLRv.1, PFLRv.2 and PDR, are tested within MIRA network topology based on generated traffic demands and within Internet2 network topology based on real traffic demands.

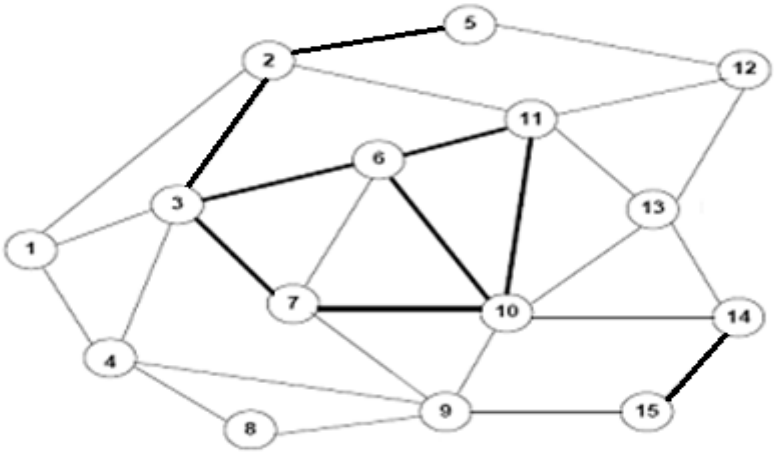

Figure 5.1 MIRA network topology [36].

5.1.1.1 MIRA network

The first network topology is the MIRA network [36] that is often used in the validation of many advanced routing algorithms, such as MIRA [36], DORA [43], PBR [44] and LIOA [45] algorithms.

As described in Chapter 2, the objective of MIRA, DORA, PBA and LIOA algorithms is to enhance the routing performance within the MPLS-based routing. All the previous algorithms aim to consider the future of route requests in order to reduce the interference that happen between the requests in the future.

The MIRA topology has 15 nodes and 28 links as shown in Figure 5.1. The thicker links have a capacity of 4800 capacity units while the thinner links have a capacity of 1200 capacity units.

Figure 5.2 COST266bt network topology [109].

5.1.1.2 COST266bt network

The second network topology is a real network topology that is shown in Figure 5.2. It is a reference topology suited for a pan-European fiber-optic network and is named COST266bt [109]. The COST266bt topology has 28 nodes and 41 links, which is bigger than the MIRA topology. COST266bt topology was used in the

validation of many routing algorithms, such as Interference Minimizing Routing Algorithm (IMRA) [112], [113] and Fast Minimum Interference Routing Algorithm (FMIRA) [114]. Also, both of IMRA and FMIRA algorithms are proposed in order to enhance the routing performance within the MPLS-based routing.

In the COST266bt topology, the thicker links have a capacity of 4800 capacity units while the thinner links have a capacity of 1200 capacity units. Both of MIRA and COST266bt topologies were used with respect to a generated traffic.

5.1.1.3 Internet2 network

The third network topology is a real network topology that is shown in Figure 5.3. It is a reference topology suited for an advanced hybrid optical and packet network in U.S. named Internet2 [110]. The Internet2 topology has 9 nodes and 13 links. All the links of Internet2 network topology have the same capacity which is equal to 149,000 Bps.

Figure 5.3 Internet2 network topology [110].

The Internet2 network topology was used in the evolution of many traffic engineering algorithms, such as [115], [116], and [117]. In [115], The Internet2 topology was used to evaluate the performance of a new link-state routing protocol, called Penalizing Exponential Flow-splitting (PEFT), which target to achieve optimal traffic engineering. In [116], The Internet2 topology was used to evaluate the performance of a new selective protection scheme for handling failures in link state routing protocol. In [117], the Internet2 topology was used to compare the performance of different TE techniques that exploit path diversity.

The Internet2 network topology is tested with respect to a real traffic demands. The real traffic demands are collected from the trace files of the NetFlow tool for the first day of the year 2009 [118].

5.1.1.4 GÉANT network

The forth network topology is a real network topology named GÉANT [111] (See Figure 5.4). The GÉANT network is a pan-European research network that interconnects the European National. The GÉANT topology has 20 nodes and 32 links. All the links of GÉANT network topology have the same capacity which equals to 1200 Mbps.

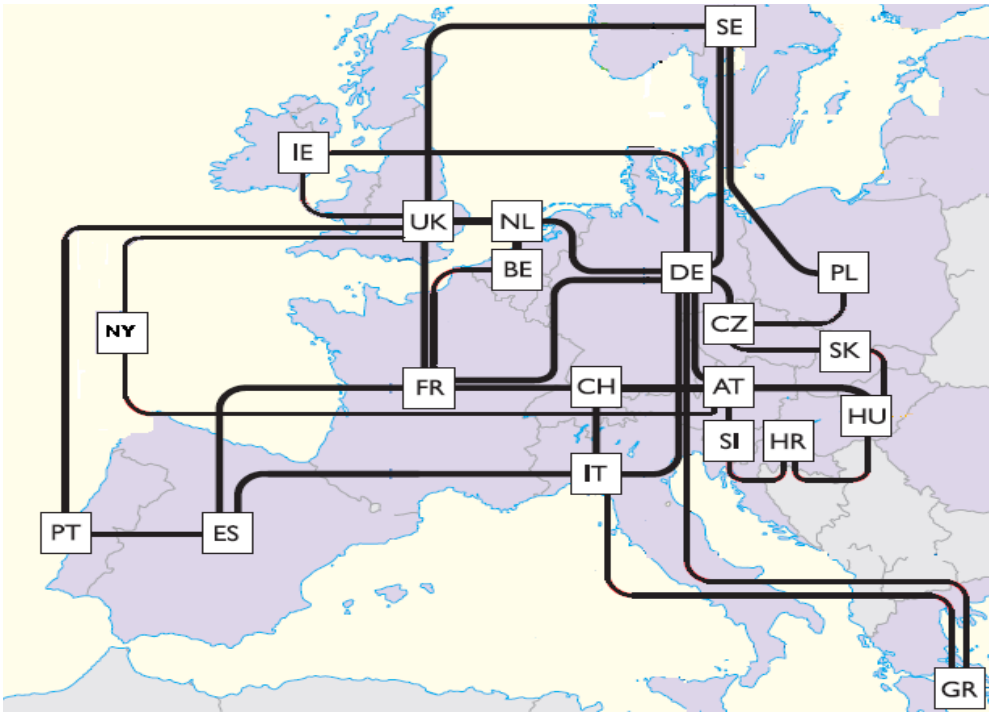

Figure 5.4 GÉANT network topology [111].

The GÉANT network topology is tested with respect to a real traffic demands. The real traffic demands are collected from the trace files of the NetFlow tool for the first two days of the year 2005 [119].

5.1.2 Traffic demands

The performance studies of the proposed algorithms are done based on generated and real traffic demands. In the first subsection, the details of generated traffic are presented. After that, the description of real traffic is presented.

5.1.2.1 Generated traffic

In the generated traffic, all possible combinations of source and destination pairs are assumed in order to consider the most general case. This means that, every node can be a source or a destination. This leads to 210 source and destination pairs for the MIRA network topology and 765 sources and destination pairs for the COST266bt network topology. From the reservation point of view, the total number of source and destination pairs may be decreased. The reason is that, the best path from node *A* to node *B* is not necessary different than the best path from node *B* to node *A*, especially in the centralized routing approach. The traffic is evenly distributed among all source and destination pairs. Each source and destination pair has the same probability to request a path. During the simulation, each request is served one at a time.

During the performance study, there are two different generated traffic models are considered. The first generated traffic model is the Poisson traffic model and the second is the ON-OFF traffic model. In the next sub sections, the description of both of them will be presented in more details.

5.1.2.1.1 Poisson traffic model

The Poisson traffic model is the traditional traffic model used to generate the traffic demands in order to evaluate the routing algorithms [36], [43], [46], [113], [114] and [121]. In this model, the number of requests per time unit follows a Poisson distribution with mean (λ) and the holding time of the requests is exponentially distributed with mean ($1/\mu$).

During the performance studies, the performance of proposed algorithms is tested under different network loads. For statistical purposes, each load scenario is executed five times with the same λ and μ values. In the first load scenario, called Moderate Load (ML), λ is equal to 15 requests per time unit and ($1/\mu$) is equal to 35 time unit. In the second load scenario, called Heavy Load (HL), λ is equal to 20 requests per time unit and ($1/\mu$) is equal to 29 time unit.

Within the same network topology, the "traffic intensity" term represents the mean number of simultaneous served requests in progress, which is defined in [121], as (λ/μ). In the ML scenario, the traffic intensity, (i.e. (λ/μ) or ($\lambda \times (1/\mu)$), is equal to ((15×35) = 525. In the HL scenario, the traffic intensity is equal to

(20×29) = 580. The objective here is to obtain different rejection ratios under the previous network load scenarios. For example, within the MIRA topology and the WSP algorithm, the rejection ratio of requests for the HL scenario is more than the double of rejection ratio for ML scenario.

The request capacities are randomly distributed among 5, 10, 15, …, and 50 capacity units. The difference between the link capacities and request capacities is large in order to allow testing thousands of requests. However, if the difference is too large, a huge number of requests are needed in order to reach the state of rejected requests. Furthermore, the used range of request capacities was used to evaluate advanced routing algorithms within the MPLS-based network, such as [43] and [121].

Additionally, both of link capacities, request capacities, arrival rate of requests and holding time of requests is assumed with these values to obtain the state of rejection for requests. Of course, if there is no rejection for requests, it is not possible to demonstrate the improvement of routing performance.

5.1.2.1.2 ON-OFF traffic model

The second used generated traffic model is the ON-OFF model. The main reason for using another generated traffic model is to efficiently generate a self-similar traffic that can exhibit the main characteristics of real network traffic. According to our discussion in section 3.6, the main characteristics of real network traffic are the burstiness, self-similarity and long rage dependency. There is various traffic models have been proposed to generate self-similar traffic. The common features between all of them are the representing of heavy-tailed phenomena [122].

The ON-OFF traffic model [123] and [124] is one of the most popular self-similar traffic models. It is used in many research works, such as [125], [126], [127] and [128]. The generated traffic using the ON-OFF model is an output of superposition operation of many independent ON-OFF sources. During the ON period, the source transmits the data. However, there is no any data is sent during the OFF Model (See Figure 5.5). The duration of ON and OFF periods are subject to heavy-tailed probability distributions. The common used heavy-tailed distribution in the ON-OFF traffic model is the Pareto distribution.

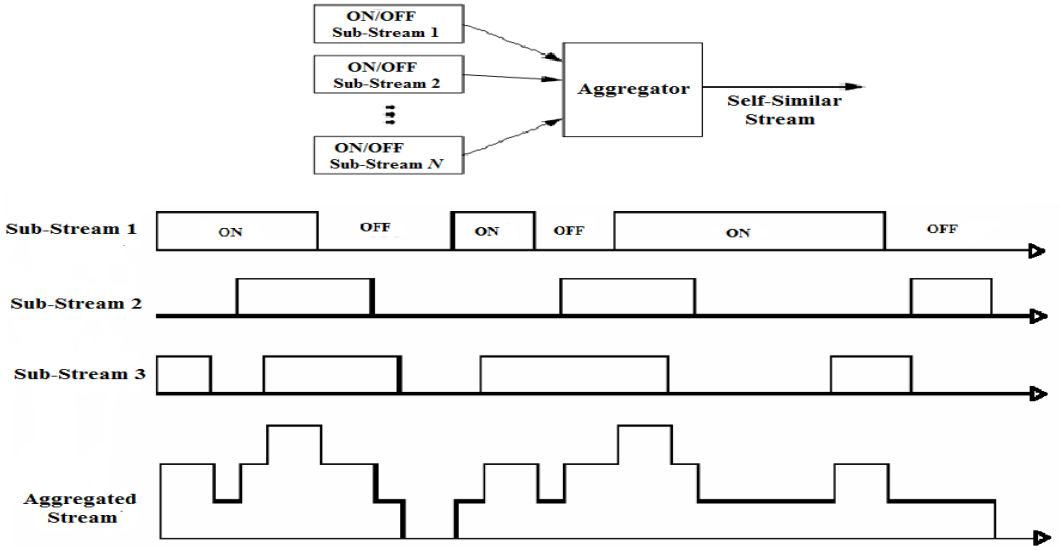

Figure 5.5 ON-OFF Traffic Model [124].

According to the generated Self-similar traffic in the next performance studies, the number of individual sources is thirty two sources. The duration of ON and OFF periods are represented using the Pareto distribution. The Mean of ON period equals five and Pareto shape parameter equals 1.3. However, The Mean of OFF period equals five and Pareto shape parameter equals 1.7. The holding time for requests follows another member of heavy-tailed probability distributions class, named Weibull distributions.

During the performance studies, the performance of proposed algorithms is tested under different network loads. In the moderate network load scenario, the used value for the Weibull scale parameter is 3 and Weibull shape parameter equals 0.7. However, In the heavy network load scenario, the used value for the Weibull scale parameter is 3.2 and Weibull shape parameter equals 0.7. The objective here is to obtain different rejection ratios under the previous network load scenarios. For statistical purposes, each load scenario is executed fifteen times with the same parameter values of used heavy tailed distributions.

In general, According to the referenced scientific researches that use of ON-OFF traffic model, the previously mentioned values of model parameters are selected in order to achieve a generated traffic that exhibits the main characteristics of real network traffic. Additionally, in the section 5.3, the experiment results that measure the burstiness and self- similarly for the generated traffic will be demonstrated.

In the reaming part in the current section, the use of ON-OFF approach to model the World Wide Web (WWW) traffic will be discussed in more details.

- *Web Traffic Model (ON-OFF Approach)*

The ON-OFF traffic model is traditionally used to model the web traffic [129], [130] and [131]. In order to understand how the ON-OFF approach is used to model the web traffic, the detailed structure of web traffic must be explained first. The main building block of the web traffic molding is the web page. The web page is constructed using Hypertext Markup Language (HTML) code. The structure of web page (named main object) is defined using the HTML code. In the main object, the HTML code is usually refers to other files (named inline objects), such as the images or other Web objects that are created with scripting languages.

When the user requests a web page, the Hypertext Transfer Protocol (HTTP) sends a request message to the desired server. After that, the Transmission Control Protocol (TCP) is used to manage the transmission of the web page from the server to the user. When the server receives the HTTP request, it sends the main object to the user. Furthermore, if the web page contains other inline objects, it will be subsequently sent.

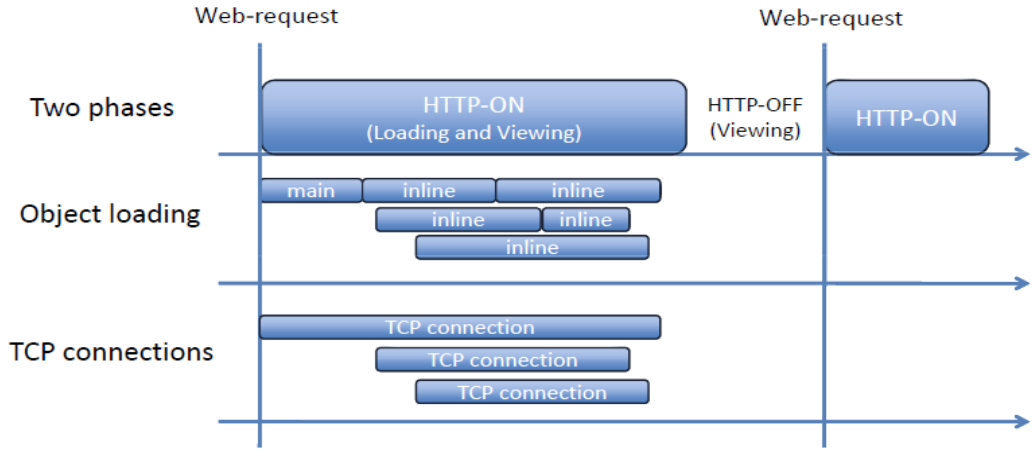

Figure 5.6 The behavioral web traffic model [129].

The proposed model in [129], known as the behavioral model, is the most famous model in the web traffic modeling spot. The behavioral model consists of three levels (See Figure 5.6). The first level is the HHTP connection level, the second is the web object level and the third is the TCP level. In this level, the HTTP level has two phases, HTTP-ON and HTTP-OFF phases. When the user requests a web

page, the HTTP-ON phase will start. This period represents the downloading and the viewing in parallel. During the download for parts of a web-request, the user can view the finished downloaded part of web page. However, the HTTP-OFF phase represents the inactive period between two successive web-requests.

In addition to the mentioned scientific researches for the ON-OFF approach (as a web traffic model), the use of ON-OFF traffic approach to model the web traffic is confirmed and recommended during the following evaluation methodology studies, IEEE wireless broadband standard (IEEE 802.16m) [132], Worldwide Interoperability for Microwave Access (WiMAX) system [133] and Code Division Multiple Access (CDMA) 2000 system [134].

Furthermore, according to the analytical studies of real web traffic, there are two studies aim to validate and verify the use of ON-OFF approach to model the web traffic. The first study examines the web traces collected at the department of computer science within Boston University [135]. The authors in this research work explain the observed phenomena of long range dependence in the web traffic. Also, in this study, the proposed methodology to demonstrate the long range dependence behavior follows the main idea of ON-OFF traffic model in [123] and [124] (which proposes the superposing of heavy-tailed ON/OFF sources in order to exhibit the long-range dependence).

The second analytical study introduces a framework for web traffic model that aims to reproduce the presence of heavy-tailed distributions in traffic characteristics by implementing the superposition of multiple ON-OFF traffic sources [136]. The authors have validated their model when they have compared the characteristics of generated traffic with the attributes of captured HTTP traffic from the Sprint PCS CDMA-1xRTT access network.

5.1.2.2 Real traffic

During the performance study, two different datasets of extracted network traffic are considered. The first dataset is a real traffic matrix that is collected from the trace files of the Internet2 network for the first day of the year 2009 [118]. The second dataset is a real traffic matrix that is collected from the trace files of the GÉANT network for the first two days of the year 2005 [119]. In the next sections, the description for both of them will be presented in more details.

5.1.2.2.1 Internet2 dataset

This dataset is extracted from the trace files of the Internet2 network. These trace files are collected on all the edge links using the NetFlow tool. It contains the TCP/UDP traffic for the first day of the year 2009 [118]. The range of request capacities is between 50 to 5000 Bps. The request capacities are consistent with heavy-tailed distribution. In the rest of this part, the extraction process of traffic demands is explained in details.

In the real scenario, when a specific traffic demand is requested, the routing algorithm selects the best path and forwards the packets in this path. After that, the NetFlow tool archives all flows that go through every network node. There are nine trace files that are collected from the Internet2 network nodes. Each trace file contains the records of flows that go through related network node. Each flow record contains the source IP address, the destination IP address, the time, the total number of bytes in this flow and other related information to this flow.

There is a searching process is preformed in the all nine trace files in order to extract the real traffic demand. The objective here is to take every flow and search about it in all nine trace files. After that, all occurrences of this flow, in all trace files, are ordered by the time in a list. This means, the first occurrence of this flow in the list is the first node which is visited by this flow. In other words, this is the source node. Also, the last occurrence of this flow in the list is the last node which is visited by this flow. In other words, this is the destination node.

In addition to the source and destination information, the request time and the holding time of this flow can be extracted from this list. The request time is the time of occurrence at the source node and the holding time is the difference between the time of occurrence at the destination node and the time of occurrence at the source node.

According to the simulation, all required information of the real traffic demands are extracted now. This means that, the extracted traffic demands can be handled like the generated traffic. During the simulation, the actual used values of link capacities of Internet2 topology are decreased. The objective here is to obtain the state of rejection for requests and test the performance of proposed algorithms under this simulated rejection condition.

5.1.2.2.2 GÉANT dataset

This dataset is extracted from the trace files of the GÉANT network. These trace files are collected on all the edge links using the NetFlow tool. It contains the traffic for the first two days of the year 2005 [119]. The range of request capacities is between 1 to 640 Mbps. In the rest of this part, the extraction process of traffic demands is explained in more details.

The traffic matrices are constructed with the help of the IGP routing information, collected Netflow data and BGP routing information of the GÉANT network. The extraction process of traffic matrices are done by the TOTEM toolbox [137]. TOTEM toolbox is an open-source framework for integrating various TE algorithms.

TOTEM toolbox follows three steps in order to create the traffic matrixes. The first is to build the desired network structure with the help of the one-day IS-IS protocol trace. The next step is computing the routes known by each router to reach each destination prefix. The routes computations are done using an internal module which follows the desired methodology of the BGP routing algorithm. With the help of these computed routes, the received traffic at each ingress route will be routed within the network until it reaches the egress router for this destination. The matrix is constructed by the summation of traffic going from any ingress router to any egress router.

Furthermore, in the section 5.3, the experiment results that measure the burstiness and self- similarly for this real traffic will be demonstrated.

5.1.3 Routing algorithms

The performance of PFLRv.1 algorithm is evaluated with respect to two traditional routing algorithms, WSP and CSPF algorithms, which had been described in section 2.1.2 and 2.1.3. Furthermore, the performance of PFLRv.1 algorithm is evaluated with respect to an advanced routing algorithm, LIOA algorithm, which had been described in section 2.1.6.

The performance of PFLRv.2 algorithm is evaluated with respect to WSP, which is a traditional routing algorithm, and LIOA, which is an advanced routing algorithm. Furthermore, the PFLRv.2 algorithm is compared with two estimation-

based routing algorithms, the DF-PI and PSA algorithms, which had been described in section 2.2.

The two versions of PDR algorithm (PDRv.1 and PDRv.2) are compared with two different ACR routing algorithms, the AntNet and TB algorithms, which had been described in section 2.3. Additionally, the two versions of PDR algorithm are compared with two different centralized routing algorithms, the CSPF and LIOA algorithms, which had been described in sections 2.1.3 and 2.1.6

5.1.4 Measured parameters and statistical analysis

Three measured parameters are presented during the following studies:

- The rejection ratio of requests:

$$= \left(\frac{No.of\ rejected\ requests}{total\ number\ of\ requests}\right) * 100$$

- The bandwidth blocking rate:

$$= \left(\frac{the\ sum\ of\ rejected\ BW}{the\ total\ sum\ of\ requested\ BW}\right) * 100$$

- The rejection ratio of re-routed requests upon link failure:

$$= \left(\frac{No.of\ rejected\ requests\ that\ need\ rerouting}{total\ number\ of\ requests\ that\ need\ rerouting}\right) * 100$$

5.1.5 The comments on expected results

The objective of proposed algorithms is to enhance the dynamic routing performance by reducing the rejection ratio of path requests, minimizing the bandwidth blocking rate and rerouting the requests upon the link failure. In the rest of this section, the comments and reasons of these enhancements are discussed in details.

The current dynamic routing algorithms update the link states with the current available BW. But, due to the varying nature of the available BW, updating the link state with the current measured BW is not an efficient approach to represent the link utilization. Also, the decisions of routing algorithm that depend on a single sample of measured available BW, which has not much significance due to

the variable traffic nature, are not completely accurate. Therefore, it is very important to consider the changes of link loads according to the future path request in order to reduce the interference between the requests in the future and so reduce the occurrence of network congestions and increase the network utilization.

There is small effort in this research direction, such as the DF-PI algorithm that uses the statistical model to estimate the link loads and the PSA algorithm that uses the linear algebra equations to estimate the link loads. In contrast to the DF-PI and PSA algorithms, in the proposed algorithms, the ANN is used to build the adaptive predictor in order to predict future link loads because the ANN offers accurate prediction capabilities with different types of network traffic and has the ability to be adaptive.

One of the important features of the proposed algorithm is the link state representation. The proposed algorithms combine the predicted link load with the current link load with an effective method in order to optimize the link weights. The idea is to reduce the number of wrong and critical decisions in case of uncertain prediction accuracy.

Finally, In addition to the adaptability of the used ANN, the proposed algorithms have the ability to adapt the internal algorithm parameters, such as the length of prediction step, depending on the prediction accuracy in order to efficiently estimate the link traffics and so enhance the routing performance.

5.2 The evaluation of PFLRv.1 algorithm: Single step-ahead model

In this section, the performance of PFLRv.1 algorithm (Single step-ahead model) is evaluated based on several test scenarios and the results are discussed [22], [23]. The PFLRv.1 algorithm is bundled with the WSP and CSPF algorithms. During this performance study, the bundled versions are compared with the unbundled versions. In section 5.2.1, the simulation details are presented. In section 5.2.2, the MIRA topology is considered and the performance of the compared algorithms is tested. In section 5.2.3, the COST266bt topology is considered and the performance of the compared algorithms is tested. In section 5.2.4, the link failure scenario is considered and the performance of the compared algorithms is tested.

5.2.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Three performances parameters are measured:
    - The rejection ratio of requests.
    - The bandwidth blocking rate.
    - The rejection ratio of re-routed requests upon link failure.
  - 10,000 of requests are generated using the Poisson model in each single simulation run.
  - The routing performance will be tested under two different network load scenarios. In each load scenario, an analysis study is performed on the 2,000 requests, which are requested after the first 4,000, in order to select the best values o*f WS* an*d* α parameters. The reason is to be sure that, the traffic behavior already reached the steady state within all network links.
  - In the previous analysis study, the experiment is repeated 100 times (*WS* parameter ranges from 6 to 10 and α parameter ranges from 0 to 1 with 0.05 steps). In each repetition, PFLRv.1 algorithm is tested with different combination of the *WS* and α parameters. At the end of every repetition,

the rejection ratio of requests is measured. The best values of *WS* and *α* parameters that achieve the lowest rejection ratio of requests.

- After selecting the best *WS* and *α* parameters, the PFLRv.1 algorithm starts running after the first 4,000 requests, again, but until the last request of the 10,000 requests. This means that, all the comparisons between the bundled and unbundled routing algorithms are based on 6,000 requests.
- The box plot graph is used in all experiments in order to represent the comparison between the different routing algorithms. Each column in the box plot graph shows six statistical values for many simulation runs: the minimum value, lower quartile (Q1) value, median (Q2), upper quartile (Q3) value, average value and maximum value [138]. The Q1 value, which is the bottom of the box, cuts off lowest 25% of data. The Q2 value, which is the inner line inside the box, separates the higher half of data set from the lower half of data set. The Q3 value, which is the top of the box, cuts off lowest 75% of data set.

- The parameters of PFLRv.1 algorithm:-

Table 5.3 summarizes the result of the analysis studies that are performed in order to select the best values of *WS* and *α* parameters for the PFLRv.1 algorithm in every scenario (see Appendix A.1).

Table 5.3 The best values of *WS* and *α* parameters (PFLRv.1).

Network load scenario	Algorithm	MIRA topology		COST266bt topology	
		WS	*α*	*WS*	*α*
ML	WSP	7	0.15	7	0.2
	CSPF	7	0.15	6	0.15
HL	WSP	7	0.1	10	0.1
	CSPF	8	0.05	8	0.25

5.2.2 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.2.2.1 The ML scenario

Figure 5.7 shows the rejection ratio of requests for the ML scenario in the MIRA topology. The average of results shows that, both of the WSP_PFLRv.1 and

CSPF_PFLRv.1 algorithms outperform the WSP and CSPF algorithms. The WSP_PFLRv.1 algorithm rejects 12.09% less requests than the normal WSP algorithm. Also, the CSPF_PFLRv.1 algorithm rejects 7.14% less requests than the normal CSPF algorithm.

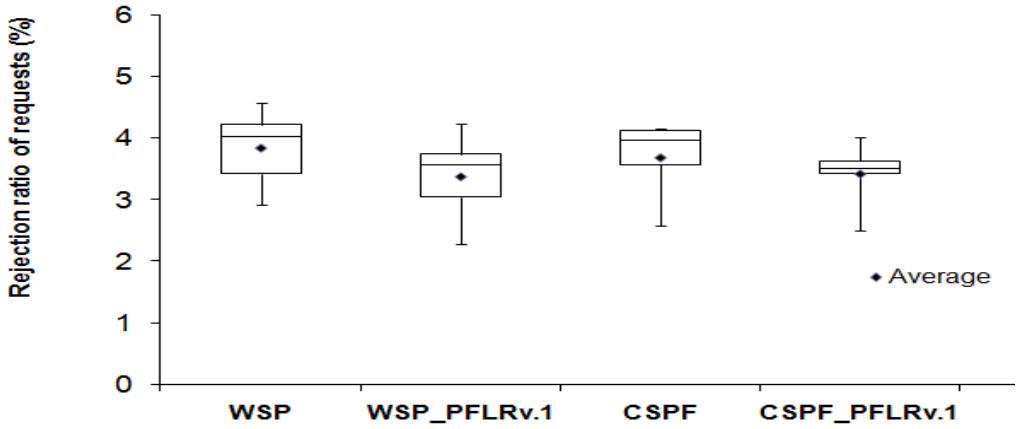

Figure 5.7 The rejection ratio of requests for the ML scenario.

The experimental results show that, combining the predicted available BW with the current available BW is an effective method in order to optimize the link weights. Because of that, the routing performance is improved. The PFLRv.1 algorithm has the same objective of advanced routing algorithms. It aims to consider the future of route requests in order to reduce the interference that happen between the requests in the future and so reduces the occurrence of network congestions and also increase the network utilization.

According to the compared routing algorithms, both of WSP and CSPF algorithms use the current available BW information in order to select the best path between the source and destination nodes. The WSP algorithm prefers the widest path from the equal shortest paths, which contain the minimum number of hops. The widest path is the path in which the bottleneck link (i.e. link with smallest bandwidth) has the largest bandwidth among other bottlenecks in other shortest paths. Furthermore, The CSPF algorithm runs the Dijkstra's algorithm depending on link weights that are inversely proportional to the residual link capacities. The shortest path, which is found by the Dijkstra's algorithm, is the path that has the least total of link weights.

Another remarkable result is that, the CSPF algorithm does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. Both of

WSP_PFLRv.1 and CSPF_PFLRv.1 algorithms combine the predicted available BW with the current available BW in order to represent the available BW information. The WSP_PFLRv.1 algorithm compares all the equal shortest paths and selects the widest path. However, the CSPF_PFLRv.1 algorithm selects the shortest path that is firstly found by the Dijkstra's algorithm without comparing the equal shortest paths. Therefore, the PFLRv.1 algorithm has a better chance to enhance the performance of WSP algorithm more than the CSPF algorithm. Furthermore, the performance of CSPF algorithm is better than the WSP algorithm and whenever the routing selection is closer to the optimal selection, the enhancement becomes harder.

Figure 5.8 shows the bandwidth blocking rate for the ML scenario in the MIRA topology. The results show that, the WSP_PFLRv.1 algorithm rejects 14.74% less bandwidth than the normal WSP algorithm. Also, the CSPF_PFLR algorithm rejects 5.79% less bandwidth than the normal CSPF algorithm.

The PFLR algorithm does not only target to enhance the rejection ratio of requests, but also it targets to enhance the bandwidth blocking rate at the same time. It is not a significant improvement to reduce the rejection ratio of requests and increase the bandwidth blocking rate at the same time.

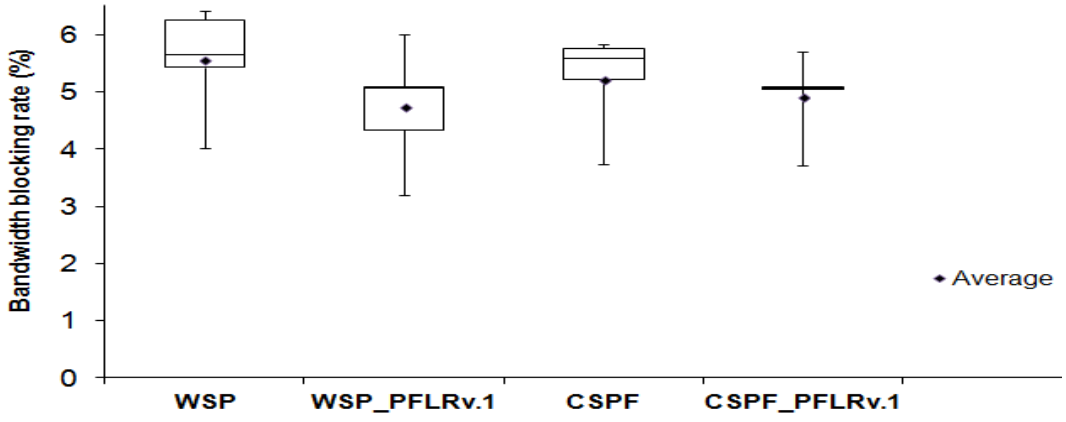

Figure 5.8 The bandwidth blocking rate for the ML scenario.

As described before in the result comments of rejection ratio comparative study, The CSPF algorithm does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. Therefore, the enhancement of BW blocking rate with WSP_PFLRv.1 algorithm is better than the enhancement of BW blocking rate with CSPF_PFLRv.1 algorithm.

5.2.2.2 The HL scenario

Figure 5.9 shows the rejection ratio of requests for the HL in the MIRA topology. The average of results shows that, the WSP_PFLRv.1algorithm rejects 6.30% less requests than the normal WSP algorithm. Also, the CSPF_PFLRv.1 algorithm rejects 4.79% less requests than the normal CSPF algorithm.

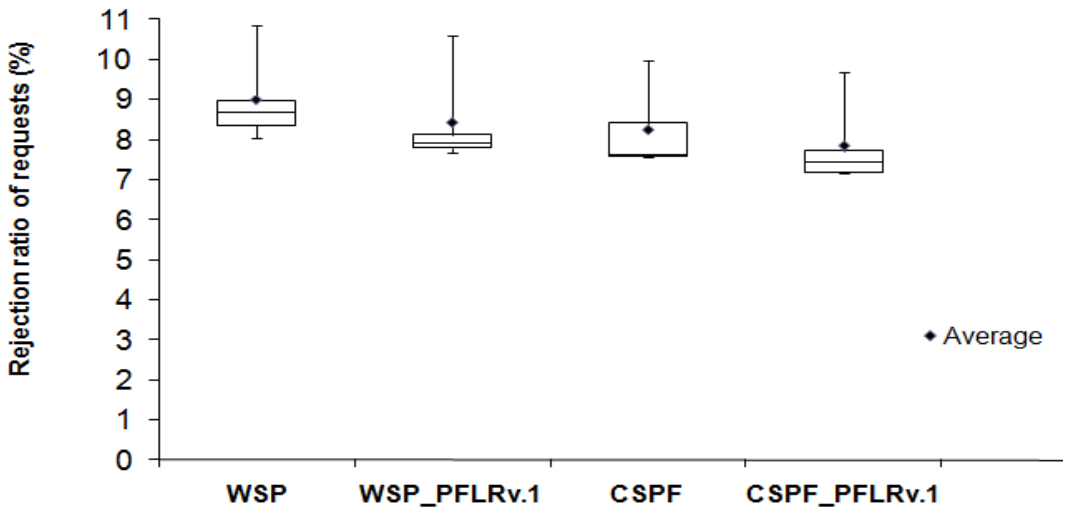

Figure 5.9 The rejection ratio of requests for the HL scenario.

In the HL scenario, the rejection ratio of requests is increased because the requests reserve the BW of the network links for a larger number of events than the ML scenario. Additionally, the total available BW of network links is decreased and this has direct effects on the optimization of the routing algorithms. This means that, the PFLRv.1 algorithm does not have the same large numbers of competitive decisions in order to optimize the routing performance. Thus, the performance enhancement of the routing algorithms using the PFLRv.1 algorithm is affected by network load scenario.

Figure 5.10 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 5.24% less bandwidth than the normal WSP algorithm. Also, the CSPF_PFLR algorithm rejects 4.70% less bandwidth than the normal CSPF algorithm.

As mentioned before, the PFLR algorithm does not only target to enhance the rejection ratio of requests, but also it targets to enhance the bandwidth blocking rate at the same time. It is not a significant improvement to reduce the rejection ratio of requests and increase the bandwidth blocking rate at the same time.

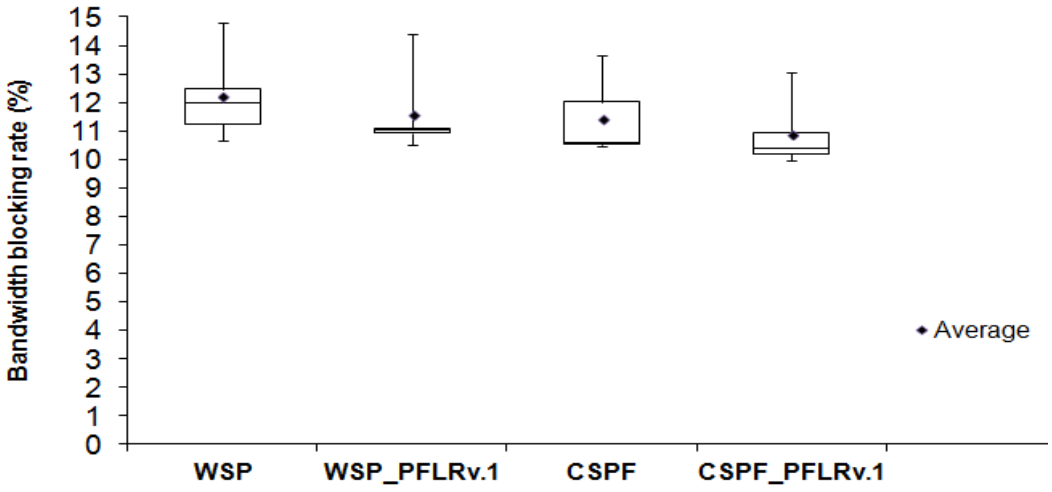

Figure 5.10 The bandwidth blocking rate for the HL scenario.

Furthermore, as described before within the result comments of rejection ratio comparative study (in ML scenario), The CSPF algorithm does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. Therefore, the enhancement of BW blocking rate with WSP_PFLRv.1 algorithm is better than the enhancement of BW blocking rate with CSPF_PFLRv.1 algorithm.

5.2.3 The COST266bt topology

In the following scenarios, the COST266bt topology is considered and the performance of the routing algorithms is tested in both ML and HL scenarios.

5.2.3.1 The ML scenario

Figure 5.11 shows the rejection ratio of requests for the ML in the COST266bt topology. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 6.55% less requests than the normal WSP algorithm. Also, the CSPF_PFLRv.1 algorithm rejects 4.76% less requests than the normal CSPF algorithm.

The rejection ratio of requests increased in the COST266bt topology compared to the MIRA topology. The reason is the difference between their features. The node network degree for the MIRA topology is equal to 3.73. While the node network degree for the COST266bt topology, which is equal to 3.54, is smaller. Furthermore, the percentage of links with the high capacity within the MIRA topology is equal to 32.14%. While the percentage of links with the big capacity within the COST266bt topology, which is equal to 24.39%, is smaller.

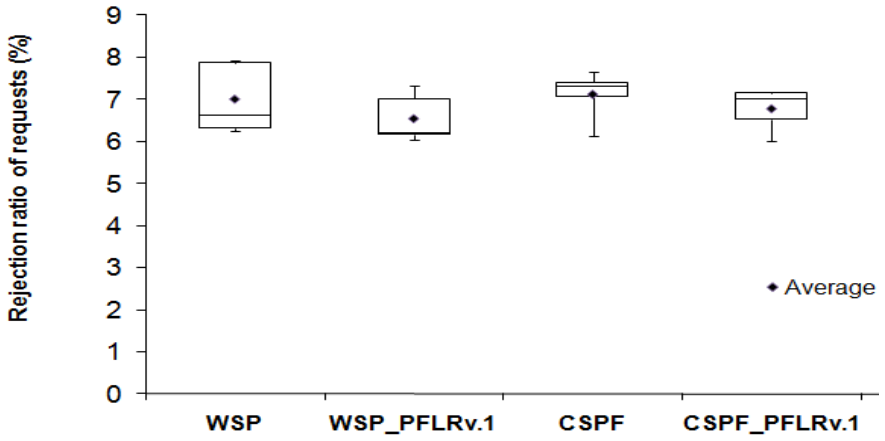

Figure 5.11 The rejection ratio of requests for the ML scenario.

Figure 5.12 shows the bandwidth blocking rate for the moderate ML in the COST266bt topology. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 5.47% less bandwidth than the normal WSP algorithm. Also, the CSPF_PFLR algorithm rejects 5.72% less bandwidth than the normal CSPF algorithm.

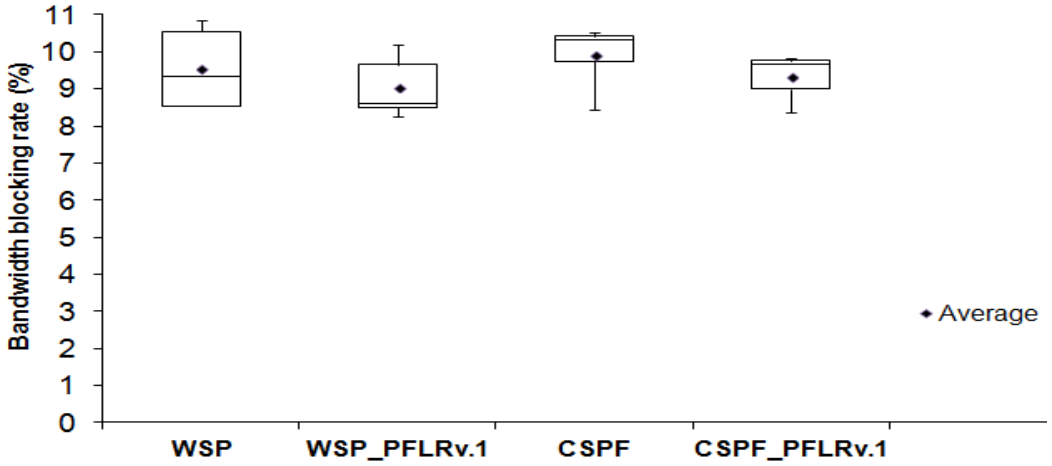

Figure 5.12 The bandwidth blocking rate for the ML scenario.

Also, the bandwidth blocking rate increases within the COST266bt topology compared to the bandwidth blocking rate within the MIRA topology. This is for the same reason which causes the increment for the rejection ratio of requests (See the result comments of rejection ratio comparative study in the current scenario).

5.2.3.2 The HL scenario

Figure 5.13 shows the rejection ratio of requests for the HL in the COST266bt topology. The average of results shows that, the WSP_PFLRv.1 algorithm rejects

4.25% less requests than the normal WSP algorithm. Also, the CSPF_PFLRv.1 algorithm rejects 2.96% less requests than the normal CSPF algorithm. As described before in section 5.2.2.2, the rejection ratio of requests increases for the HL scenario compared to the rejection ratio of requests increased for the ML scenario.

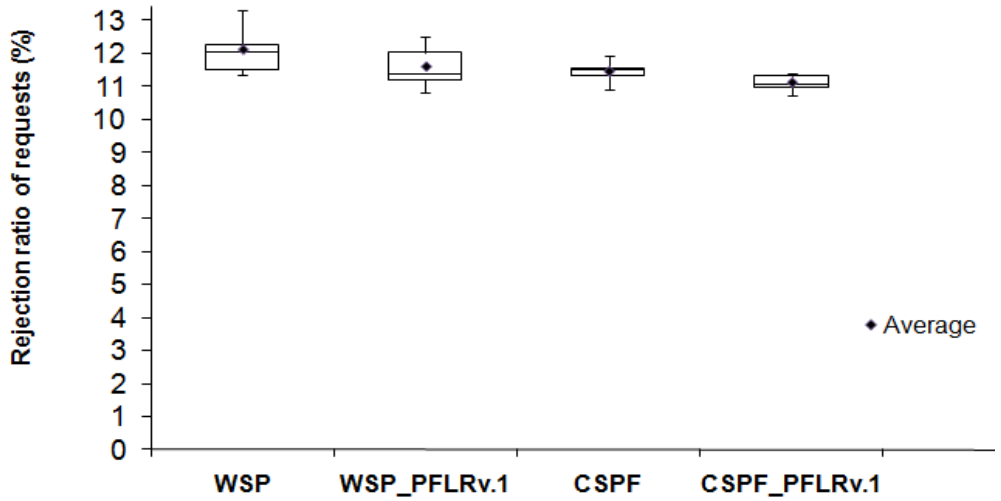

Figure 5.13 The rejection ratio of requests for the HL scenario.

Figure 5.14 shows the bandwidth blocking rate for the HL scenario in the COST266bt topology. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 3.47% less bandwidth than the normal WSP algorithm. Also, the CSPF_PFLR algorithm rejects 2.98% less bandwidth than the normal CSPF algorithm.

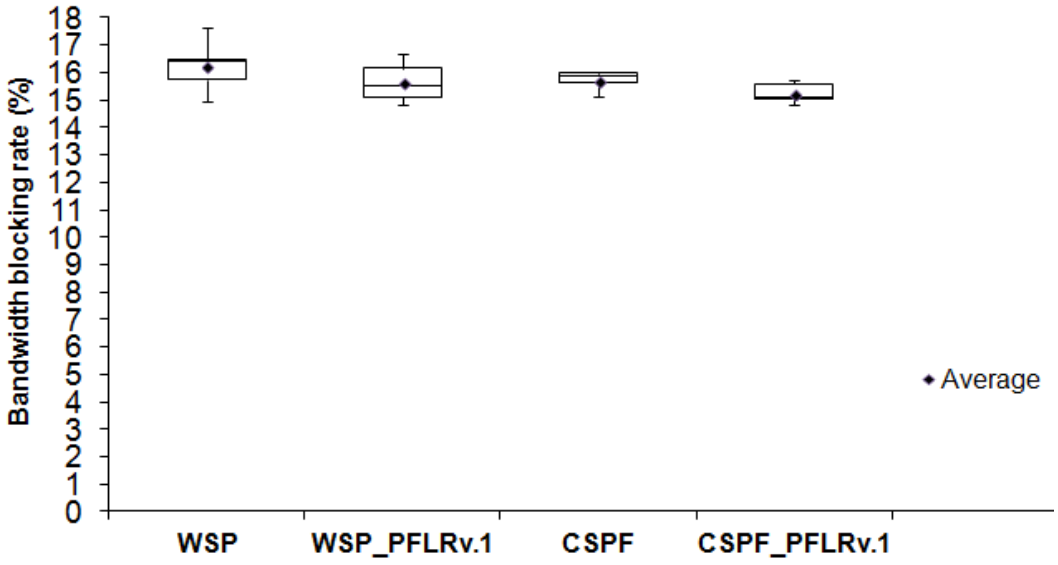

Figure 5.14 The bandwidth blocking rate for the HL scenario.

5.2.4 Rejection ratio of re-routed requests upon link failure

In each trial of the next experiment, one link of the network links is taken down, one at a time. Then the routing algorithms recover the requests which require the rerouting. The previous steps are repeated for each network link. Table 5.4 shows the average rejection ratio for rerouted requests.

Table 5.4 The rejection ratio of rerouted requests upon link failure scenario.

Network load scenario	Algorithm	Rejection ratio for rerouted LSPs (%)	
		MIRA network	COST266bt network
ML scenario	WSP	31.18	47.52
	WSP_PFLR	30.98	47.39
	CSPF	37.25	39.72
	CSPF_PFLR	37.03	39.56
HL scenario	WSP	57.37	65.89
	WSP_PFLR	57.03	65.84
	CSPF	36.28	70.19
	CSPF_PFLR	35.78	70.11

The results show that, the WSP_PFLR and CSPF_PFLR algorithms have a better chance to reroute the requests than the normal WSP and CSPF algorithms. The main reason of this enhancement is the effective mechanism of PFLR algorithm. With the use of PFLR algorithm, the future statuses of the network link loads are considered. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future. Because of that, the traffic load will be efficiently distributed on the network topology. In other words, the traffic load balancing with the help of PFLR algorithm is much better. Thus, when the traffic load will be efficiently distributed, this gives a good chance for the routing algorithm to reroute the requests upon link failure scenario.

5.3 The evaluation of PFLRv.1 algorithm: Multi steps-ahead model

In this section, the performance of PFLRv.1 algorithm (Multi steps-ahead model) is evaluated based on several test scenarios and the results are discussed. During this performance study, the PFLRv.1 algorithm is bundled with the WSP and LIOA algorithms. After that, the bundled versions are compared with the unbundled versions. In section 5.3.1, the simulation details are presented. In section 5.3.2, the experimental results for measuring the burstiness and self-similarity of used generated and real traffic are presented. In section 5.3.2, the MIRA topology is considered and the performance of the compared algorithms is tested. In section 5.3.3, the GÉANT topology is considered and the performance of the compared algorithms is tested.

5.3.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Two performances parameters are measured:
    - The rejection ratio of requests and
    - The bandwidth blocking rate.
  - 30,000 of requests are generated using the ON-OFF traffic model in each single simulation run.
  - The routing performance will be tested under two different network load scenarios. In each load scenario, an analysis study is performed on the 5,000 requests, which are requested after the first 20,000, in order to select the best values o*f WS* an*d* α parameters. The reason is to be sure that, the traffic behavior already reached the steady state within all network links.
  - In the previous analysis study, the experiment is repeated 120 times (*WS* parameter ranges from 1 to 12 with and α parameter ranges from 0 to 1 with 0.1 steps). In each repetition, the PFLRv.1 algorithm is tested with different combination of the *WS* and α parameters. At the end of every repetition, the rejection ratio of requests is measured. The best values of

WS and *α* parameters are the values that achieve the lowest rejection ratio of requests.

- After selecting the best *WS* and *α* parameters, the PFLRv.1 algorithm starts again running after the first 20,000 requests, but until the last request of the 30,000 requests. This means that, all the comparisons between the bundled and unbundled routing algorithms are based on 10,000 requests.
- Also, the box plot graph is used in all experiments in order to represent the comparison between the different routing algorithms.

- The parameters of PFLRv.1 algorithm:-

Table 5.5 summarizes the result of analysis studies that are performed in order to select the best values of *WS* and *α* parameters for the PFLRv.1 algorithm within MIRA and GÉANT topologies.

Table 5.5 The best values of *WS* and *α* parameters.

Compared algorithms	Algorithm parameter	MIRA		GÉANT
		ML	HL	
WSP	*WS*	12	12	8
	α	0.4	0.3	0.3
LIOA	*WS*	10	10	5
	α	0.3	0.3	0.25

5.3.2 The approve of self-similar traffic

Before going ahead to test and show the routing performance for the comparative algorithms, the self-similar traffic features for the generated and real traffic will be presented first. For each traffic type, the burstiness and self-similarity features will be demonstrated.

Figures 5.15, 5.16 and 5.17 show the values of aggregated traffic with respect to different time scales for the generated traffic using Poisson, the generated traffic using ON-OFF model and the real traffic of GÉANT topolgy respectively.

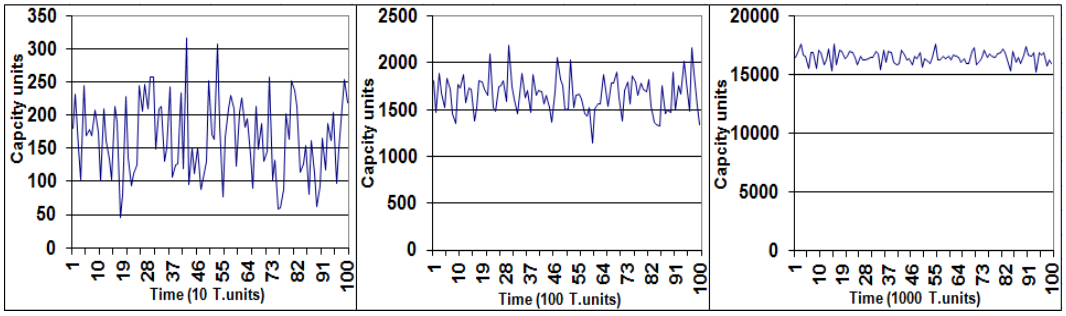

Figure 5.15 The burstiness of generated traffic using Poisson model.

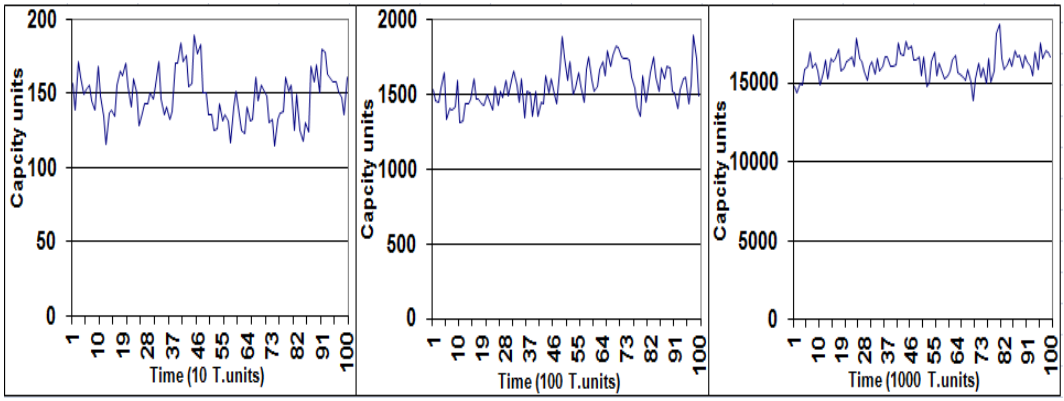

Figure 5.16 The burstiness of generated traffic using ON-OFF model.

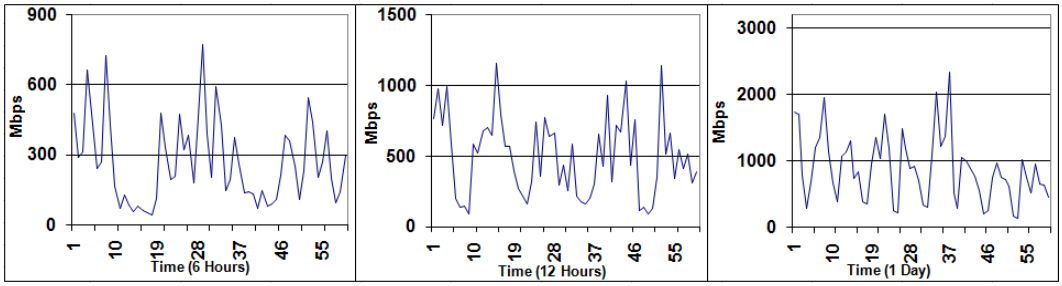

Figure 5.17 The burstiness of real traffic of GÉANT topology.

Figure 5.15 show that, the curve of aggregated generated traffic using the Poisson model is going to be smooth in the large time scale. In the contrast of aggregated generated traffic using the Poisson model, the curves still bursty in the aggregated generated traffic using ON-OFF model and real traffic even in the large time scales (See Figures 5.16 and 5.17).

According to the self-similarity feature, the variance time plot is used in order to measure the self-similar parameter *H* (Hurst parameter [139]). The Hurst parameter is an index for the long rang dependency. If the H value of a process is larger than 0.5 and less than one, this means that, this process exhibits long range dependency. In order to draw the variance time plot, the traffic is aggregated at different scales *m*. Then, the variance of aggregated traffic *X(m)* is calculated. After that, the variance of *X(m)* is plotted versus *m* on log-log plot. Finally, the fitting least square line that goes through the points is drawn. The H parameter is calculated base on the measured line slope, *H = (1-(Line Slope/2))*.

Figures 5.18, 5.19 and 5.20 show the variance time plot for the generated traffic using Poisson, the generated traffic using ON-OFF model and the real traffic of GÉANT topolgy respectively.

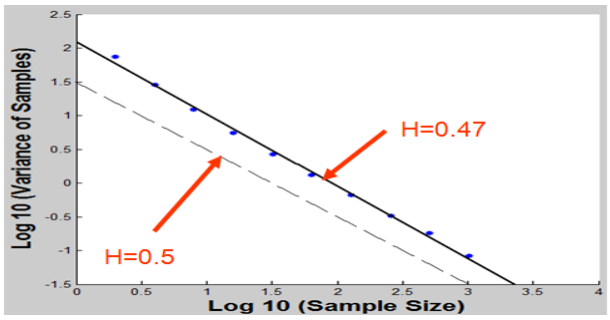

Figure 5.18 The variance time plot of generated traffic using Poisson model.

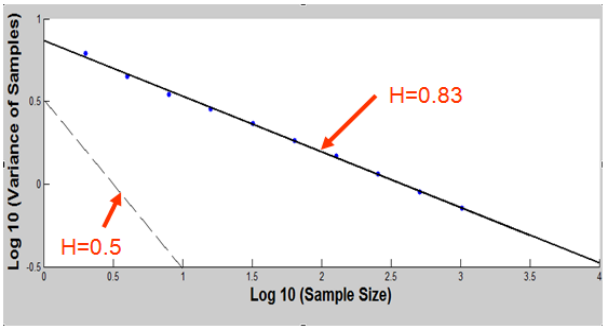

Figure 5.19 The variance time plot of generated traffic using ON-OFF model.

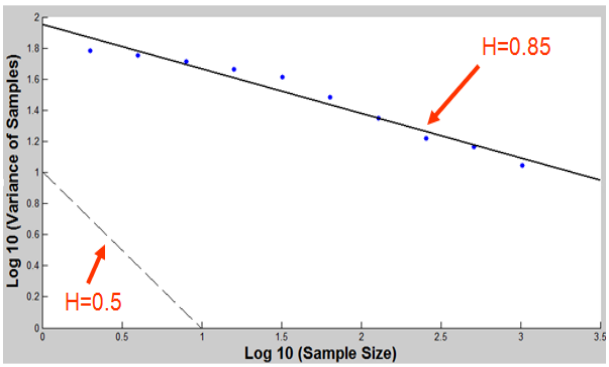

Figure 5.20 The variance time plot of real traffic of GÉANT topology.

Figure 5.18 show that, the H parameter of generated traffic using the Poisson model is less than 0.5. This means that, it does not exhibit the long range dependency feature. In the contrast of generated traffic using the Poisson model, the H parameter is more than 0.5 in the generated traffic using ON-OFF model and in the real traffic (See Figures 5.19 and 5.20). This means that, both of them exhibit the long range dependency feature.

5.3.3 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.3.3.1 The ML scenario

Figure 5.21 shows the rejection ratio of requests for the ML scenario in the MIRA topology. The average of results shows that, the WSP_PFLRv.1 algorithm (Multi steps-ahead model) rejects 17.45% less requests than the normal WSP algorithm. Also, the LIOA_PFLRv.1 algorithm (Multi steps-ahead model) rejects 14.36% less requests than the normal LIOA algorithm.

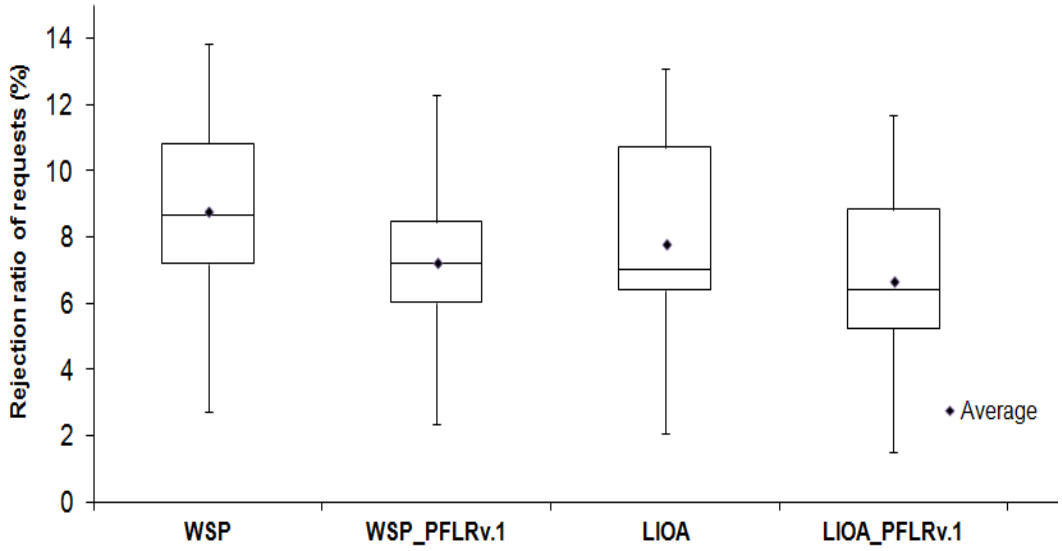

Figure 5.21 The rejection ratio of requests for the ML scenario.

However, with considering the same network load condition, the WSP_PFLRv.1 (Single step-ahead model) algorithm rejects 6.29% less requests than the normal WSP algorithm and rejects 5.24% less bandwidth than the normal WSP algorithm. Also, the LIOA_PFLRv.1 (Single step-ahead model) algorithm rejects 5.73% less requests than the normal LIOA algorithm and rejects 3.57% less bandwidth than the normal LIOA algorithm. This means that, the performance enhancement using the multi steps-ahead model is much better than the performance enhancement using the Single step-ahead model.

The main reason of this enhancement of performance is that, the multi steps-ahead mode provides more accurate traffic prediction. Moreover, there are two reasons for this accurate prediction. The first reason is the use of efficient representation methods for both the traffic samples and future link loads. In the multi steps-ahead model, the traffic sample length is determined by the mean of inter-update interval for the traffic on this network link. However, in the Single step-ahead model, the traffic sample is considered every new route request on the complete network

topology. The sample representation method in the Single step-ahead model cause a lot of data redundancy because not all network links will be updated after every route request. This data redundancy affects the prediction accuracy and increase the prediction overhead.

Additionally, the future link load in the Single step-ahead model is the predicted load value after *WS* period. However, in the multi steps-ahead model, the future link load is the average of multiple predicted load values during *WS* period. The good representation methods within the multi steps-ahead model lead to more accurate traffic predictor.

The second reason for this accurate prediction of multi Steps-ahead prediction mode is that, the measured results in the current scenario are based on the used self-similar traffic model in the current simulation scenario. However, the measured results in the scenario of single step-ahead model are based on the used Poisson traffic model. Of course, the prediction of self-similar traffic model is much easier and accurate. According to the previous mentioned two reasons, the *RMSE* in the multi steps-ahead model reached to 0.1.

According to the compared routing algorithms, both of WSP and LIOA algorithms use the current available BW information in order to select the best path between the source and destination nodes. The WSP algorithm prefers the widest path from the equal shortest paths, which contain the minimum number of hops. The LIOA algorithm is an advanced routing algorithm which aims to reduce the interference among competing flows by giving the link weights balanced values of number and quantity of flows. After that, LIOA algorithm runs the Dijkstra's algorithm depending on link weights in order to select the shortest path. The shortest path, which is found by the Dijkstra's algorithm, is the path that has the least total of link weights.

Another remarkable result is that, the LIOA algorithm, like the CSPF algorithm, does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. The WSP_PFLRv.1 algorithm compares all the equal shortest paths and selects the widest path. The selection of widest path here does not depend on the current available BW only, but it depends on the predicted available BW also. However, the link weights within LIOA_PFLRv.1 algorithm

are represented by combining the predicted available BW with the current available BW and the number of flows. After that, it selects the shortest path that is firstly found by Dijkstra's algorithm without comparing the equal shortest paths. Therefore, PFLRv.1 algorithm has a better chance to enhance the performance of WSP algorithm more than LIOA algorithm. Furthermore, the performance of LIOA algorithm is better than WSP algorithm. Whenever the routing selection is closer to the optimal selection, the enhancement becomes harder.

Figure 5.22 shows the bandwidth blocking rate for the ML scenario in the MIRA topology. The results show that, the WSP_PFLRv.1 algorithm rejects 17.63% less bandwidth than the normal WSP algorithm. Also, the LIOA_PFLRv.1 algorithm rejects 14.19% less bandwidth than the normal LIOA algorithm.

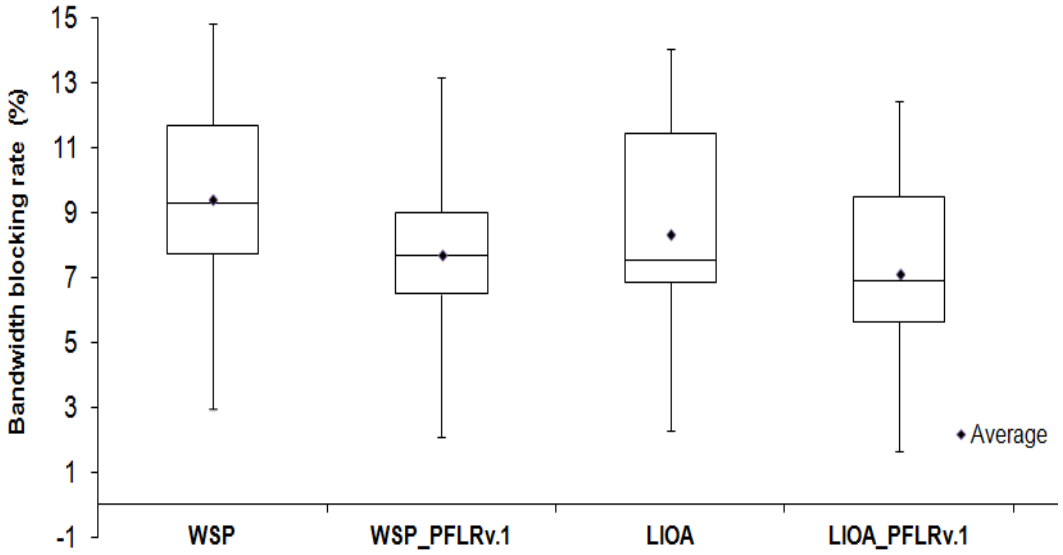

Figure 5.22 The bandwidth blocking rate for the ML scenario.

The PFLR algorithm does not only target to enhance the rejection ratio of requests, but also it targets to enhance the bandwidth blocking rate at the same time. It is not a significant improvement to reduce the rejection ratio of requests and increase the bandwidth blocking rate at the same time.

As described before in the result comments of rejection ratio comparative study, The LIOA algorithm does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. Therefore, the enhancement of BW blocking rate with WSP_PFLRv.1 algorithm is better than the enhancement of BW blocking rate with LIOA_PFLRv.1 algorithm.

5.3.3.2 The HL scenario

Figure 5.23 shows the rejection ratio of requests for the HL in the MIRA topology. The average of results shows that, both of the WSP_PFLRv.1 and LIOA_PFLRv.1 (Multi steps-ahead model) algorithms outperform the WSP and LIOA algorithms. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 12.23% less requests than the normal WSP algorithm. Also, the LIOA _PFLRv.1 algorithm rejects 10.91% less requests than the normal LIOA algorithm.

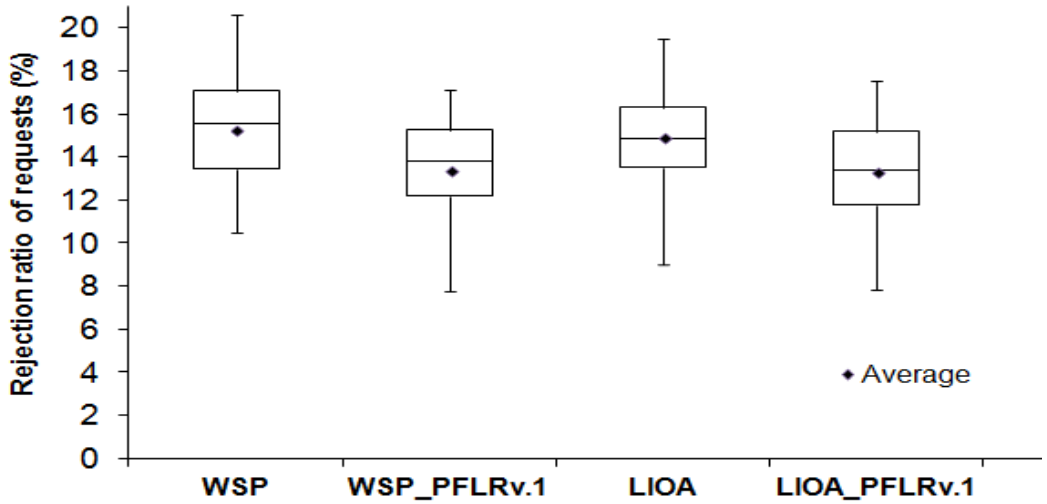

Figure 5.23 The rejection ratio of requests for the HL scenario.

The rejection ratio of requests is increased within In the HL scenario compared to its value within the ML scenario. The main reason is that, the holding time of requests is larger in the HL scenario. Thus, the BW will be reserved within the network links for a larger period and this causes more rejection of further requests in the future.

Additionally, in the HL scenario, the performance enhancement of the routing algorithms using the PFLRv.1 algorithm is decreased compared to ML scenario. The reason is that, the total available BW of network links is decreased and this has direct effects on the optimization of the routing algorithms. This means that, the PFLRv.1 algorithm does not have the same large numbers of competitive decisions in order to optimize the routing performance. Thus, the performance enhancement of the routing algorithms using the PFLRv.1 algorithm is affected by network load scenario.

Figure 5.24 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 12.46% less bandwidth than the normal WSP algorithm. Also, the LIOA_PFLR algorithm rejects 10.96% less bandwidth than the normal LIOA algorithm.

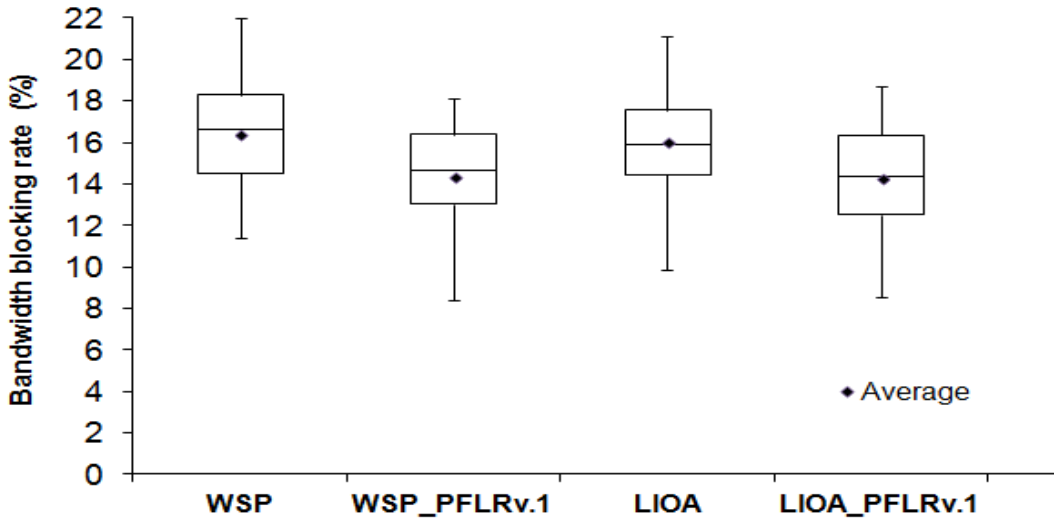

Figure 5.24 The bandwidth blocking rate for the HL scenario.

In general, as mentioned before, The PFLR algorithm (within both network load scenarios) does not only target to enhance the rejection ratio of requests, but also it targets to enhance the bandwidth blocking rate at the same time. It is not a significant improvement to reduce the rejection ratio of requests and increase the bandwidth blocking rate at the same time.

5.3.4 The GÉANT topology

Figure 5.25 shows the rejection ratio of requests for the real traffic scenario. The real dataset is extracted from the trace files of the GÉANT network. It contains the traffic for the first two days of the year 2005 [119]. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 8.97% less requests than the normal WSP algorithm. Also, the LIOA_PFLRv.1 algorithm rejects 7.09% less requests than normal LIOA algorithm.

With the help of PFLRv.1 algorithm, it is remarkable that the performance enhancement within the GÉANT network is less than the performance enhancement using the PFLR algorithm within the MIRA network. There are two reasons for this behavior. The first reason is that, the size range of requested flows

BW within the GÉANT network is larger than their respective value within MIRA topology. This wide range of requested BW size leads to higher prediction error and so cause less performance enhancement.

The second reason is that, the generated traffic within MIRA topology between the various source and destination pairs is uniformly distributed. However, the real traffic within GÉANT topology between the various source and destination pairs is not uniformly distributed. This leads to easier and more balanced prediction process in MIRA network case.

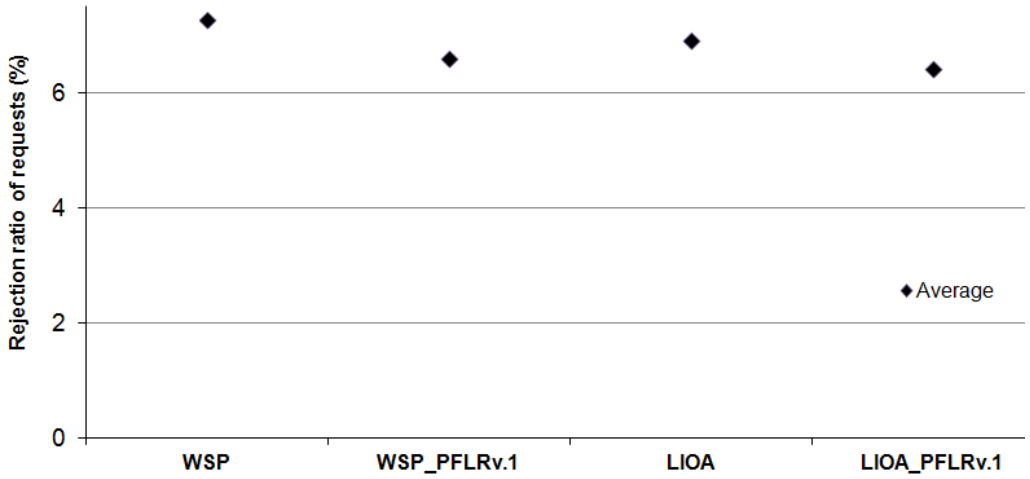

Figure 5.25 The rejection ratio of requests for the real traffic scenario.

Figure 5.26 shows the bandwidth blocking rate for the real traffic scenario. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 3.50% less bandwidth than the normal WSP algorithm. Also, LIOA_PFLRv.1 algorithm rejects 4.07% less bandwidth than the normal LIOA algorithm.

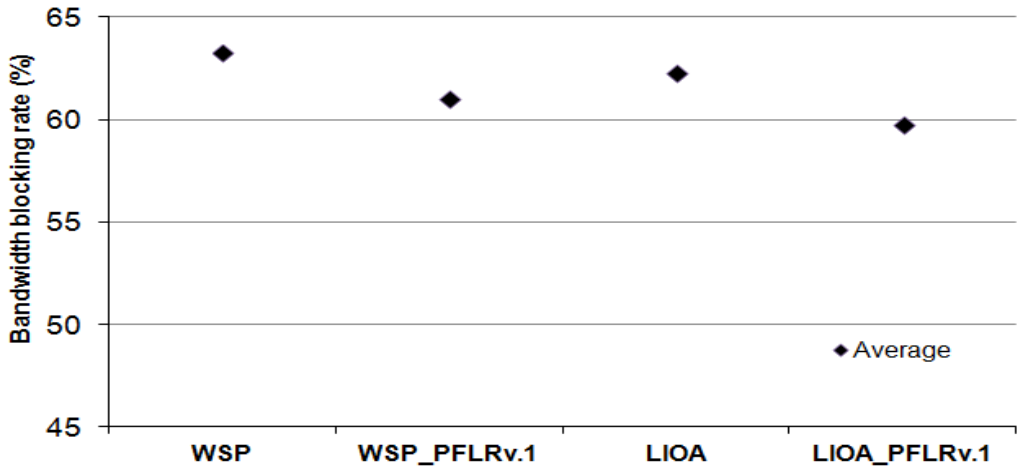

Figure 5.26 The bandwidth blocking rate for the real traffic scenario.

5.4 The evaluation of PFLRv.2 algorithm

In this section, the performance of the PFLRv.2 algorithm is evaluated based on several test scenarios and the experimental results are discussed [24]. The PFLR (v.1 and v.2 – Single step-ahead prediction model) algorithms are bundled with WSP and LIOA algorithms and compared with the unbundled versions. In section 5.4.1, the simulation details are presented. In section 5.4.2, MIRA topology is considered and the performance of compared algorithms is tested. In section 5.4.3, Internet2 topology is considered and the performance of the compared algorithms is tested. In section 5.4.4, the computation time of PFLR versions is measured.

5.4.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Three performances parameters are measured:
    - The rejection ratio of requests,
    - The bandwidth blocking rate and
    - The computation time.
  - This experiment uses the same procedure of the analysis study in section 5.2 in order to focus on the steady state of network traffic and select the best values for the parameters of PFLRv.1 algorithm, (i.e. *WS* and α), in all tested scenarios.
  - According to the parameters of PFLRv.2 algorithm, the same procedure of the analysis study is done in order select the best values of *ETh* and α parameters that achieve the lowest rejection ratio of requests.
- The parameters of PFLRv.1 and PFLRv.2 algorithms for MIRA topology:-
  Table 5.6 summarizes the result of analysis studies that are performed in order to select the best values of *WS* and α parameters for the PFLRv.1 algorithm and select the best values of *Eth* and α parameters for the PFLRv.2 algorithm in the MIRA topology (see Appendix A.1 and A.2).
- The parameters of PFLRv.1 and PFLRv.2 algorithms for Internet2 topology:
  Table 5.7 summarizes the result of analysis studies that are performed in order to select the best values of *WS* and α parameters for the PFLRv.2 algorithm

and select the best values of *Eth* and *α* parameters for the PFLRv.2 algorithm in the Internet2 scenario (see Appendix A.1 and A.2).

Table 5.6 The best values of *WS* and *ETh* and *α* and parameters (MIRA).

Network load scenario	Algorithm	PFLRv.1		PFLRv.2	
		WS	*α*	*ETh*	*α*
ML	WSP	7	0.15	51	0.25
	LIOA	8	0.2	60	0.2
HL	WSP	7	0.1	51	0.05
	LIOA	10	0.15	54	0.1

Table 5.7 The best values of *WS* and *ETh* and *α* and parameters (Internet2).

Algorithm	PFLRv.1		PFLRv.2	
	WS	*α*	*ETh*	*α*
WSP	6	0.85	1800	0.8
LIOA	9	0.8	2500	0.65

5.4.2 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.4.2.1 The ML scenario

Figure 5.27 shows the rejection ratio of requests for the ML scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 3.42% less requests than the WSP_PFLRv.1 algorithm and 15.1% less requests than the normal WSP algorithm. Also, the LIOA_PFLRv2 algorithm rejects 6.74% less requests than the LIOA_PFLRv.1 algorithm and 12.79% less requests than the normal LIOA algorithm.

The main reason for the more enhancements using the PFLRv.2 algorithm is that, PFLRv.2 algorithm has an additional adaptive parameter process. The PFLRv.1 algorithm uses a fixed value of *WS* parameter and the training process is triggered every a hundred event. However, The PFLRv.2 algorithm has the ability to adapt the *WS* and *PVP* parameters and the training process is triggered depending on the prediction accuracy. Therefore, the PFLRv.2 algorithm has a better chance in order to outperform the PFLRv.1 algorithm with respect to many performance criteria.

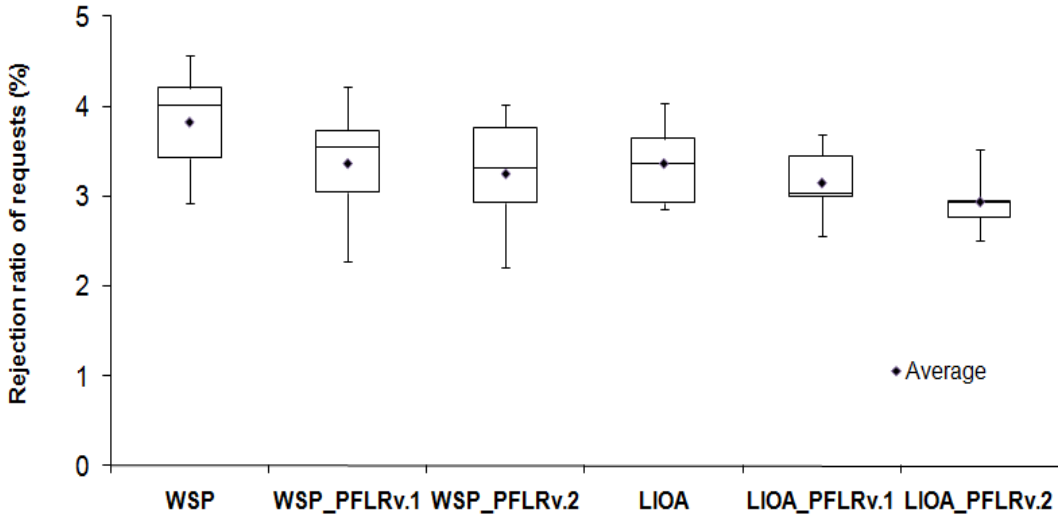

Figure 5.27 The rejection ratio of requests for the ML scenario.

According to the compared algorithms, the LIOA algorithm is an advanced routing algorithm which aims to reduce the interference among competing flows by balancing the number and quantity of flows. The MIRA algorithm outperforms the WSP and CSPF algorithms.

The LIOA algorithm, like the CSPF algorithm, does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. The WSP_PFLRv.1 algorithm compares all the equal shortest paths and selects the widest path. The selection of widest path here does not depend on the current available BW only, but it depends on the predicted available BW also. However, the link weights within the LIOA_PFLRv.1 algorithm are represented by combining the predicted available BW with the current available BW and the number of flows. After that, it selects the shortest path that is firstly found by the Dijkstra's algorithm without comparing the equal shortest paths. Therefore, the PFLRv.1 algorithm has a better chance to enhance the performance of WSP algorithm more than the LIOA algorithm. Furthermore, the performance of LIOA algorithm is better than the WSP algorithm. Whenever the routing selection is closer to the optimal selection, the enhancement becomes harder.

Figure 5.28 shows the bandwidth blocking rate for the ML scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 2.57% less bandwidth than the WSP_PFLRv.1algorithm and 16.93% less bandwidth than the normal WSP algorithm.

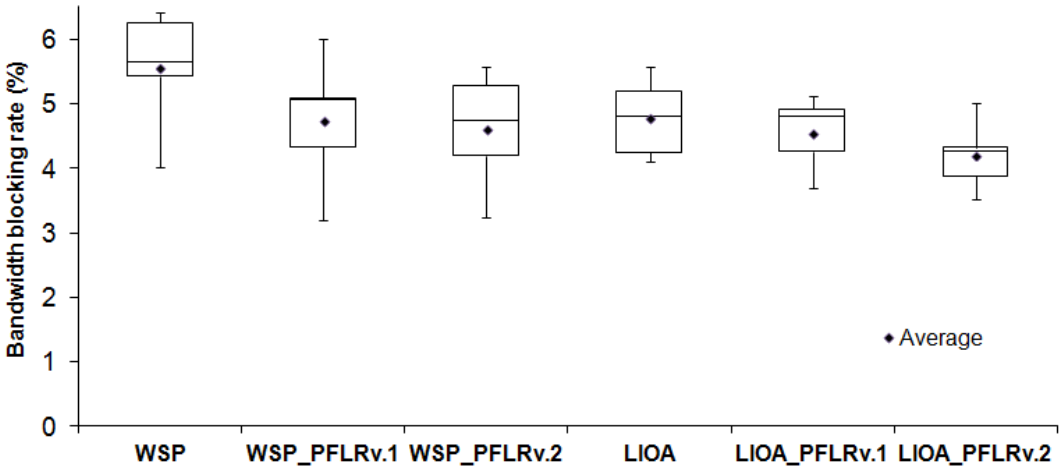

Figure 5.28 The bandwidth blocking rate for the ML scenario.

Also, the LIOA_PFLRv.2 algorithm rejects 7.73% less bandwidth than the LIOA_PFLRv.1algorithm and 12.13% less bandwidth than the normal LIOA algorithm.

In general, the PFLR algorithm does not only target to enhance the rejection ratio of requests, but also it targets to enhance the bandwidth blocking rate at the same time. It is not a significant improvement to reduce the rejection ratio of requests and increase the bandwidth blocking rate at the same time. Furthermore, the PFLRv.2 algorithm enhances the bandwidth blocking rate more than the PFLRv.1 algorithm. This is for the same reason that causes the more enhancements for the rejection ratio of requests.

As described before in the result comments of rejection ratio comparative study, The LIOA algorithm does not profit from the PFLRv.1 algorithm in the same way as the WSP algorithm does. Therefore, the enhancement of BW blocking rate with WSP_PFLRv.2 algorithm is better than the enhancement of BW blocking rate with CSPF_PFLRv.2 algorithm.

5.4.2.2 The HL scenario

Figure 5.29 shows the rejection ratio of requests for the HL scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 4.31% less requests than the WSP_PFLRv.1algorithm and 10.34% less requests than normal the WSP algorithm. Also, The LIOA_PFLRv.2 algorithm rejects 4.08% fewer requests than the LIOA_PFLRv.1 algorithm and 9.58% less requests than the normal LIOA algorithm. As described before in section 5.2.1.2, the performance enhancement

of routing algorithms using the PFLRv.1 algorithm is affected by load scenario. Also the network load scenario has the same effect on the PFLRv.2 algorithm.

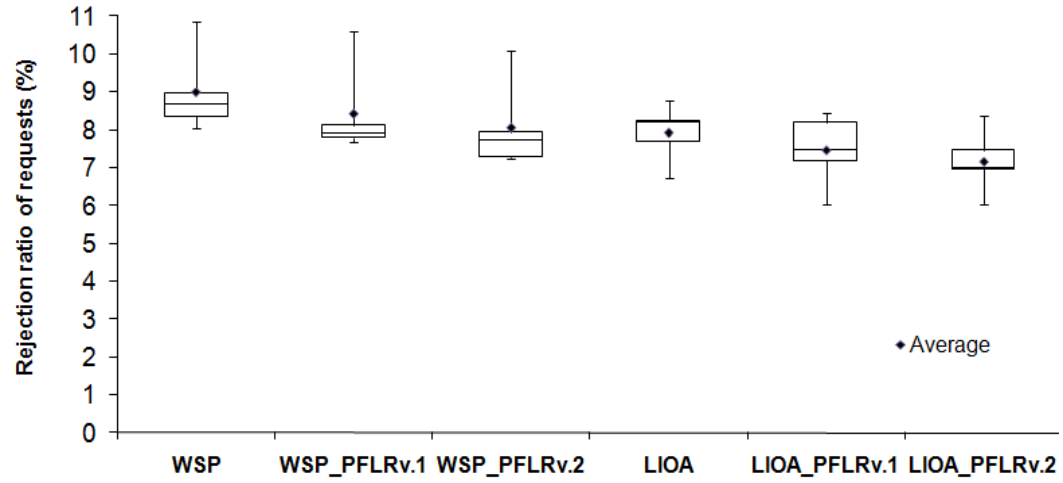

Figure 5.29 The rejection ratio of requests for the HL scenario.

Figure 5.30 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 4.42% less bandwidth than the WSP_PFLRv.1algorithm and 9.43% less bandwidth than the normal WSP algorithm. Also, the LIOA_PFLRv.2 algorithm rejects 6.02% less bandwidth than the LIOA_PFLRv.1algorithm and 9.38% less bandwidth than the normal LIOA algorithm.

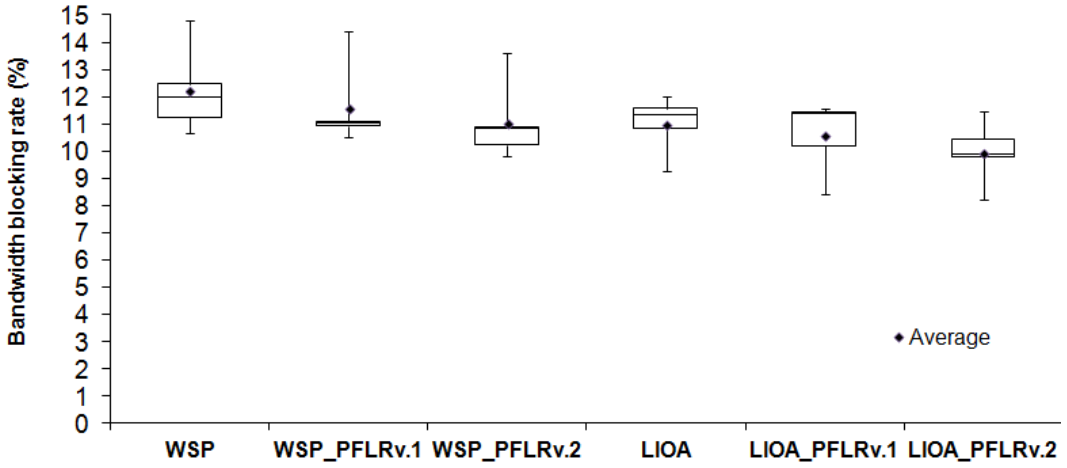

Figure 5.30 The bandwidth blocking rate for the HL scenario.

5.4.3 Real traffic scenario

Figure 5.31 shows the rejection ratio of requests for the real traffic scenario. The following result is a real trace file that contains the TCP/UDP traffic for the first day of 2009 year [118] with in Internet2 topology. The average of results shows

that, WSP_PFLRv.2 algorithm rejects 2.77% less requests than WSP_PFLRv.1 algorithm and 9.22% less requests than the normal WSP algorithm. Also, the LIOA_PFLRv.2 algorithm rejects 3.91% less requests than the LIOA_PFLRv.1 algorithm and 11.24% less requests than the normal LIOA algorithm.

With the help of PFLR algorithm, it is remarkable that the performance enhancement within the Internet2 network is less than the performance enhancement using the PFLR algorithm within the MIRA network. There are two reasons for this behavior. The first reason is that, the size range of requested flows BW within the Internet network is larger than their respective value within MIRA topology. This wide range of requested BW size leads to higher prediction error and so cause less performance enhancement.

The second reason is that, the generated traffic within MIRA topology between the various source and destination pairs is uniformly distributed. However, the real traffic within Internet2 topology between the various source and destination pairs is not uniformly distributed. This leads to easier and more balanced prediction process in MIRA network case.

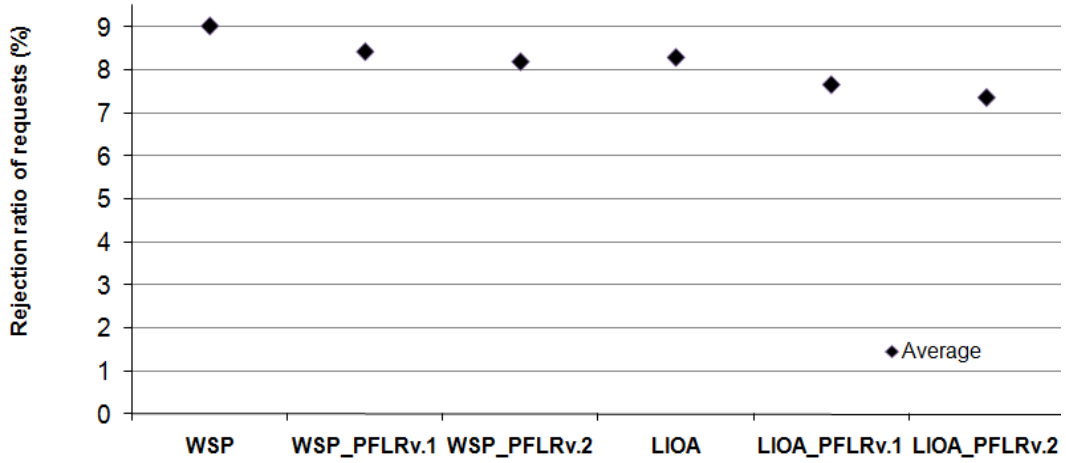

Figure 5.31 The rejection ratio of requests for the real traffic scenario.

Figure 5.32 shows the bandwidth blocking rate for real traffic scenario. The average of results shows that, WSP_PFLRv.2 algorithm rejects 0.08% less bandwidth than WSP_PFLRv.1 algorithm and 0.54% less bandwidth than normal WSP algorithm. Also, LIOA_PFLRv.2 algorithm rejects 0.27% less bandwidth than LIOA_PFLRv.2 algorithm and 1.32% less bandwidth than normal LIOA algorithm.

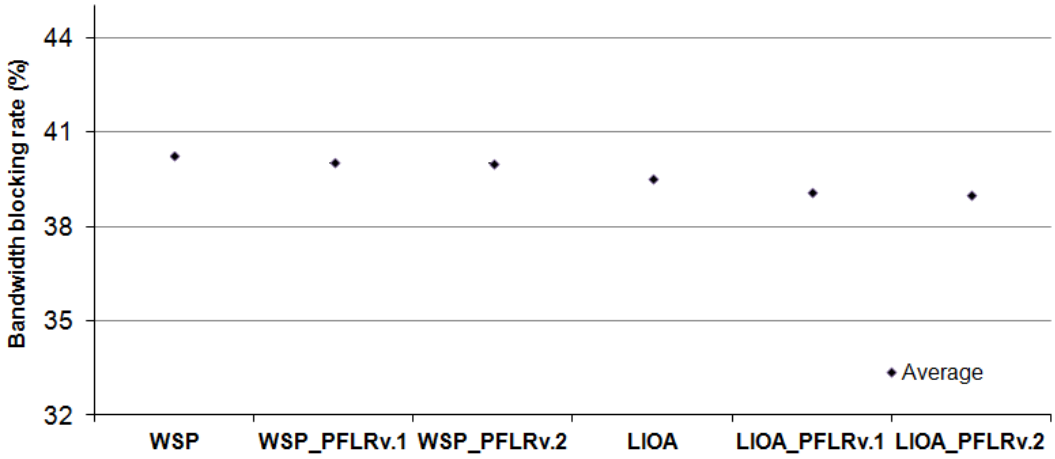

Figure 5.32 The bandwidth blocking rate for the real traffic scenario.

5.4.4 Computation time

Table 5.8 shows the computation time of PFLR versions for different load scenarios. In the following Experiments, the used processor speed is 1.8 GHZ and 1 GB RAM. In order to calculate the computation time of PFRL algorithm, firstly, the simulation time of normal routing algorithm,(T (Alg)), is measured, Then, the simulation time of bundled routing algorithm with PFLR algorithm, (T (Alg_PFLR)), is measured. As described before in chapter four, the prediction and training operations are made in parallel on all network links. Therefore, the computation time of PFRL (v1 or v2) algorithm is equal to:

(T (Alg_PFLR) - T (Alg)) / Numbers of network links.

In general, the PFLRv.2 algorithm is faster than the PFLRv.1 algorithm in all scenarios. The main reason is the new adaptive feature within the PFLRv.2 algorithm. In the PFLRv.1 algorithm, the training process is triggered every hundred event and the WS is always fixed. However in the PFLRv.2 algorithm, the *PVP* and *WS* parameters are dynamic. They are adjusted depending on the prediction accuracy. Therefore, the training process is not triggered as long as the prediction accuracy is good enough.

Table 5.8 The computation time of PFLR versions (Sec.).

Algorithm	**Network Load scenario**	
	ML	**HL**
PFLRv.1	1.36 Sec.	1.33 Sec.
PFLRv.2	0.66 Sec.	0.62 Sec.

5.5 The PFLRv.2algorithm vs. estimation-based routing algorithms

In this section, a new comparative study between the PFLRv.2 algorithm and different estimation-based routing algorithms is introduced [24]. The objective is to prove the efficiency of PFRLv.2 algorithm, based on some test scenarios and discuss the results. The DF-PI, PSA and PFLRv.2 algorithms are bundled with the WSP algorithm and compared with each other. In section 5.5.1, the simulation details are presented. In section 5.5.2, the MIRA topology is considered and the performance of the compared algorithms is tested. In section 5.5.3, the Internet2 topology is considered and the performance of the compared algorithms is tested.

5.5.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Two performances parameters are measured:
    - The rejection ratio of requests and
    - The bandwidth blocking rate.
  - This experiment uses the same procedure in section 5.2 in order to focus on the steady state of network traffic.
  - This experiment uses the same procedure of the analysis study in section 5.4 in order to select the best values for the parameters of PFLRv.2 algorithm, (i.e. *ETh* and α),in all tested scenarios.
- The parameters of PFLRv.2 algorithm:-

Table 5.9 summarizes the result of analysis studies that are performed in order to select the best values of *Eth* and α parameters for the PFLRv.2 algorithm (see Appendix A.2 for more details).

Table 5.9 The best values of *ETh* and α and parameters (WSP_PFLRv.2).

Algorithm parameter	MIRA		Internet2
	ML	HL	
ETh	54	0.1	1800
α	48	0.3	0.8

5.5.2 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.5.2.1 The ML scenario

Figure 5.33 shows the rejection ratio of requests for the ML scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 6.32% less requests than the WSP_PSA algorithm and 8.37% less than the DF-PI_WSP algorithm and 15.1% less than the WSP algorithm.

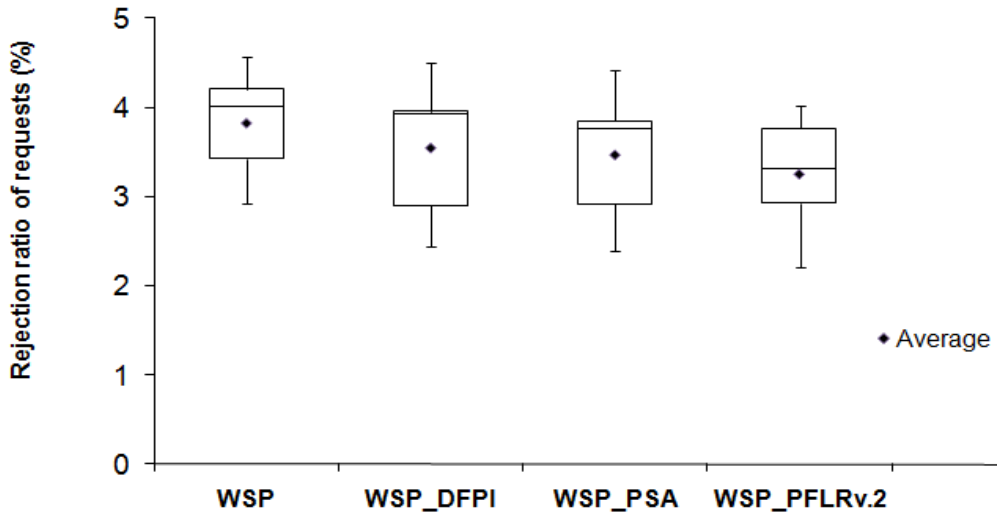

Figure 5.33 The rejection ratio of requests for the ML scenario.

The DF-PI algorithm is a statistical which uses the maximum, the minimum and the average of last recent samples of available BW during a past period to estimate the future of available BW. Then, it uses the estimated available BW in the link weights to enhance the performance of the WSP algorithm. While, the PSA is a linear prediction approach which solves the linear prediction equations to estimate the available BW and also tells the duration for which the estimate is valid with a high degree of confidence.

The experiment results show that, the proposed mechanism of PFLRv.2 algorithm enhances the performance of routing algorithms much more than statistical and linear prediction equations approaches. The PFLRv.2 mechanism combines the predicted available BW with the current available BW and incorporates both of them in the link weight formula to optimize the performance of routing algorithm.

Additionally, both of the length of prediction step and prediction validity period is adapted depending on the prediction accuracy.

Figure 5.34 shows the bandwidth blocking rate for the ML scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 12.12% less bandwidth than the WSP_PSA algorithm and 12.47% less than the WSP_DF-PI algorithm and 16.93% less than the WSP algorithm.

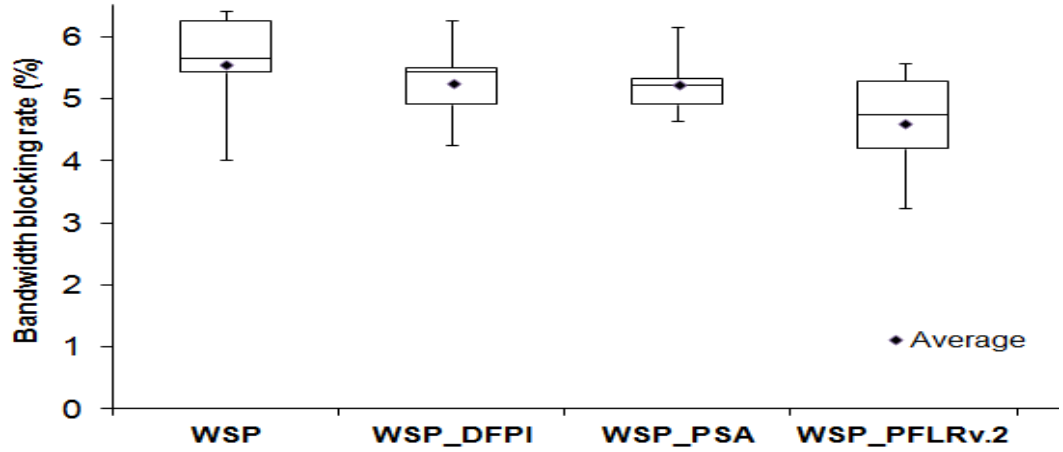

Figure 5.34 The bandwidth blocking rate for the ML scenario.

5.5.2.2 The HL scenario

Figure 5.35 shows the rejection ratio of requests for the HL scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 4.37% less requests than the WSP_PSA algorithm and 7.08% less than the DF-PI_WSP algorithm and 10.34% less than the WSP algorithm.

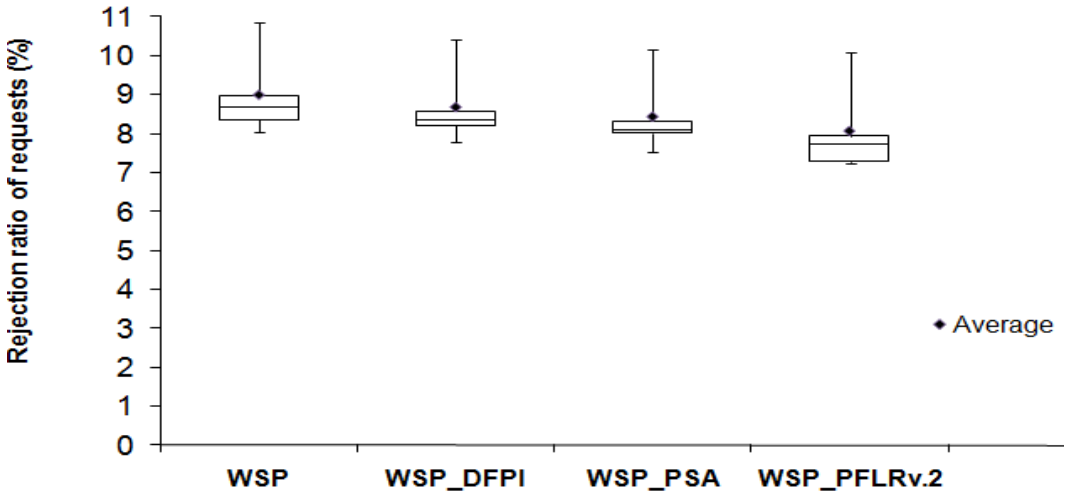

Figure 5.35 The rejection ratio of requests for the HL scenario.

As described before in section 5.2.2.2, the performance enhancement of routing algorithms using the PFLRv.2 algorithm is affected by network load scenario.

Figure 5.36 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 3.47% less bandwidth than the WSP_PSA algorithm and 5.92% less than WSP_DF-PI algorithm and 9.43% less than the WSP algorithm.

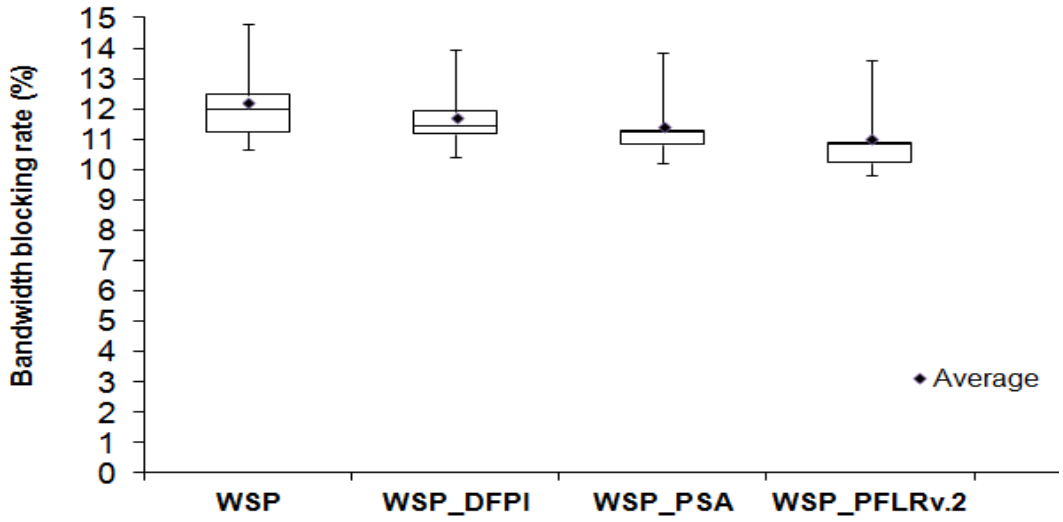

Figure 5.36 The bandwidth blocking rate for the HL scenario.

5.5.3 Real traffic scenario

Figure 5.37 shows the average of rejection rate for the real traffic scenario related to the simulation time. The average of results shows that, the WSP_PFLRv.2 algorithm rejects the fewest number of requests.

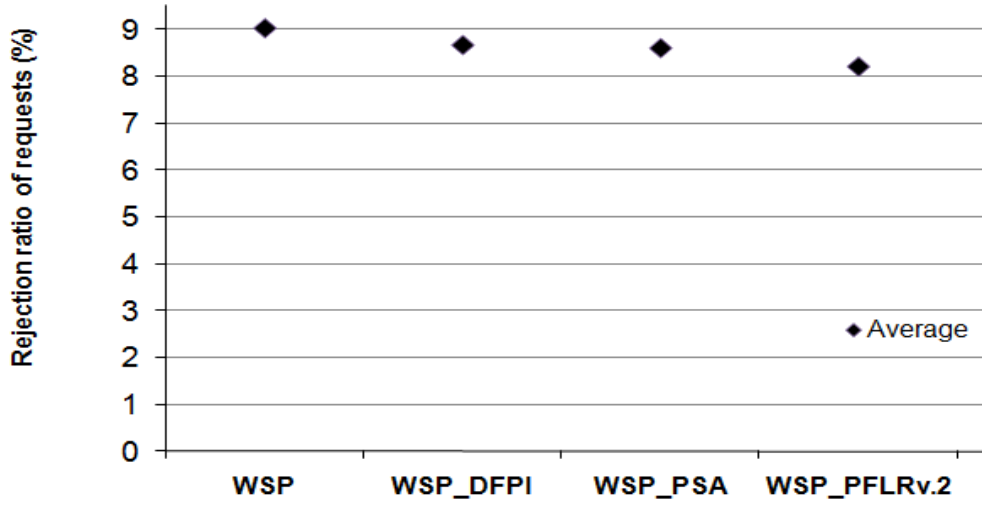

Figure 5.37 The rejection ratio of requests for the real traffic scenario.

Based on the results, the WSP_PFLRv.2 algorithm rejects 4.65% less requests than WSP_PSA algorithm and 5.38% less than DF-PI_WSP algorithm and 9.22% less than WSP algorithm.

Figure 5.38 shows the bandwidth blocking rate for the real traffic scenario related to the simulation time. The average of results shows that, the WSP_PFLRv.2 algorithm rejects 0.35% less bandwidth than the WSP_PSA algorithm and 0.38% less than the WSP_DF-PI algorithm and 0.54% less than the WSP algorithm.

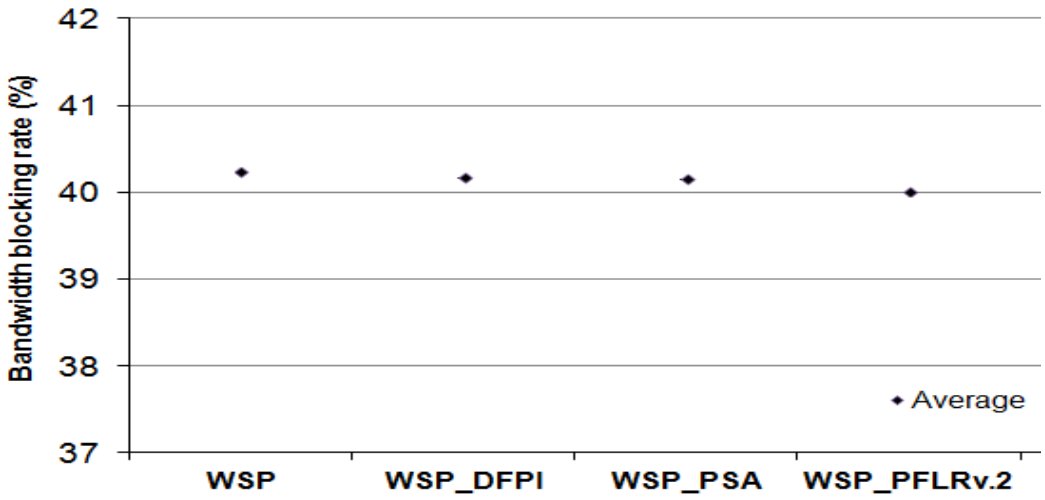

Figure 5.38 The bandwidth blocking rate for the real traffic scenario.

5.6 The PDR algorithm vs. ACR algorithms

In this section, the performance of PDR v.1 and PDRv.2 are evaluated based on some test scenarios and discuss the results [25], [26] and [27]. The AntNet and TB algorithms are modified, by replacing the transmission delay with the available BW information to be able to compare the PDR algorithm with them. In section 5.6.1, the simulation details are presented. In section 5.6.2, the MIRA topology is considered and the performance of the compared algorithms is tested. In section 5.6.3, the Internet2 topology is considered and the performance of the compared algorithms is tested.

5.6.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Three performances parameters are measured:
    - The rejection ratio of requests and
    - The bandwidth blocking rate.
    - The effect of prediction use.
  - This experiment uses the same procedure in section 5.2 in order to focus on the steady state of network traffic.
  - This experiment uses the same procedure of the analysis study in section 5.4 in order to select the best values for the parameters of PDR algorithm, (i.e. *ETh* and α), in all tested scenarios.
- The parameters of PDR algorithm:-

Table 5.10 describes the parameters of PDR algorithm and shows the used value in this simulation. Table 5.11 summarizes the result of analysis studies in Appendix A.3 that are performed in order to select the best values of *Eth* and α parameters for the PDR algorithm.

Table 5.10 The parameters of PDR algorithm.

Variable	Value
lc (least interference control parameter)	0.1
M (keep the average of the last M of *td*)	15
δ (learning rate)	0.01

θ (congestion weight)	0.25
WS (window size)	1

Table 5.11 The best values of *ETh* and α and parameters (PDR).

Algorithm	MIRA		Internet2
	ML	HL	
ETh	60	54	1800
α	0.6	0.5	0.25

5.6.2 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.6.2.1 The ML scenario

Figure 5.39 shows the rejection ratio of requests for the ML scenario. The average of results shows that, the PDRv.2 algorithm rejects 38.76% less requests than PDRv.1 algorithm, 46.17% less requests than TB algorithm and 63.40% less requests than AntNet algorithm. However, the PDRv.1 algorithm rejects 12.10% less requests than the TB algorithm and 40.24% less requests than the AntNet algorithm.

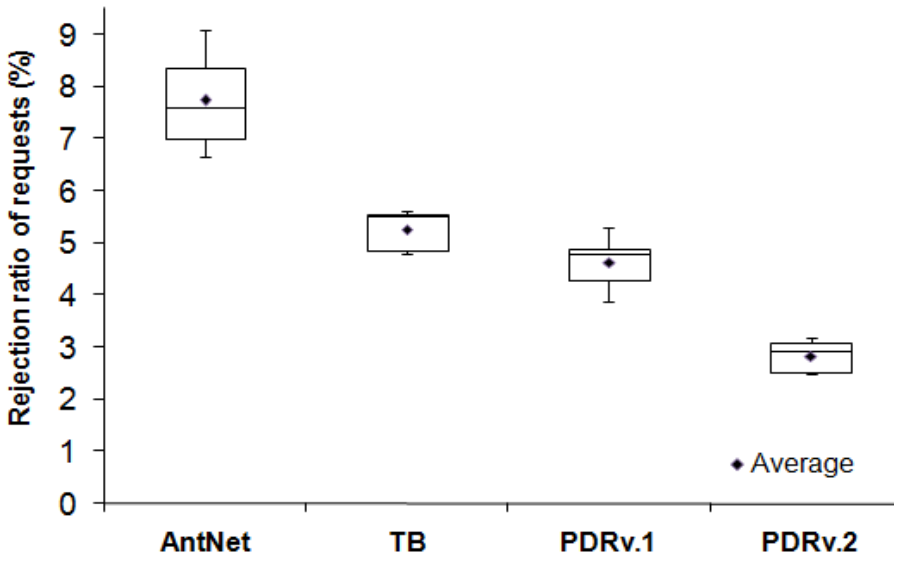

Figure 5.39 The rejection ratio of requests for the ML scenario.

The AntNet algorithm is considered the first algorithm that is inspired by ant colony behavior to solve the routing problem. However, The TB algorithm, which

is an advanced ant-based routing algorithm, is meant to be an extension of existing link-state protocols such as OSPF, which provides shortest-path information to initialize the probability table. Therefore, TB does not require a learning period to discover the network topology.

The experiment result shows that, the PDR mechanism is an effective approach which combines the current available BW and the predicted available BW in order to determine the amount of pheromone to deposit. Additionally, the proposed predictor uses an adaptive mechanism to be able to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics.

The PDRv.2 algorithm outperforms the PDRv.1 algorithm because the PDRv.2 algorithm uses a new adaptive Ant-based mechanism to be able to efficiently distribute the ants on the network topology and accurately discover the best paths. Additionally, the used Ant-based mechanism is incorporated with a new efficient prediction approach, which uses the dynamic FFNN instead of the static FFNN that is used in the previous version.

Figure 5.40 shows the bandwidth blocking rate for the moderate load scenario. The average of results shows that, the PDRv2 algorithm rejects 38.34% less BW than PDRv.1 algorithm, 44.97% less BW than the TB algorithm and 62.37% less BW than the AntNet algorithm. However, the PDRv.1 algorithm rejects 10.75% less bandwidth than the TB algorithm and 38.96% less bandwidth than the AntNet algorithm.

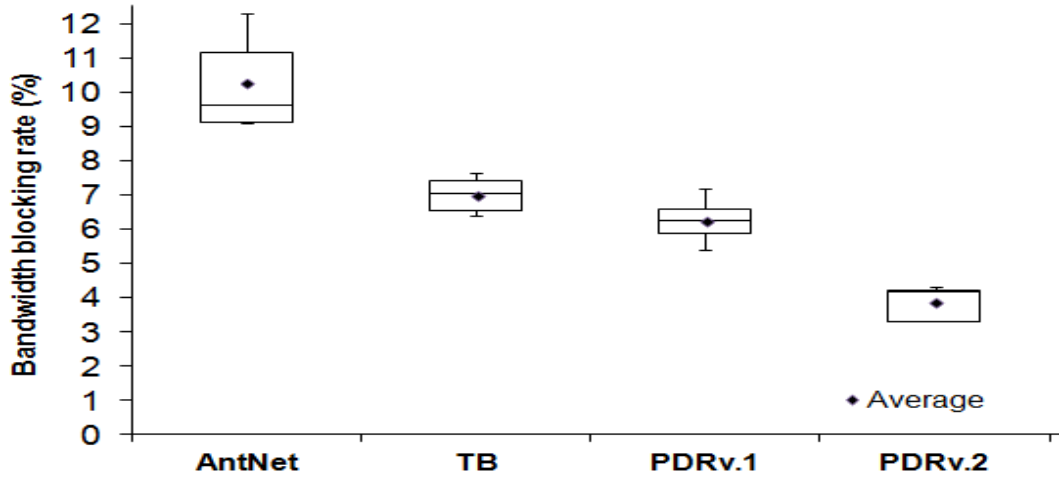

Figure 5.40 The bandwidth blocking rate for the ML scenario.

5.6.2.2 The HL scenario

Figure 5.41 shows the rejection ratio of requests for the HL scenario. The average of results shows that, PDRv.2 algorithm rejects 29.04% less requests than PDRv.1 algorithm, 33.77% less requests than TB algorithm and 45.82% less requests than AntNet algorithm. However, PDRv.1 algorithm rejects 6.66% less requests than TB algorithm and 23.65% less requests than AntNet algorithm.

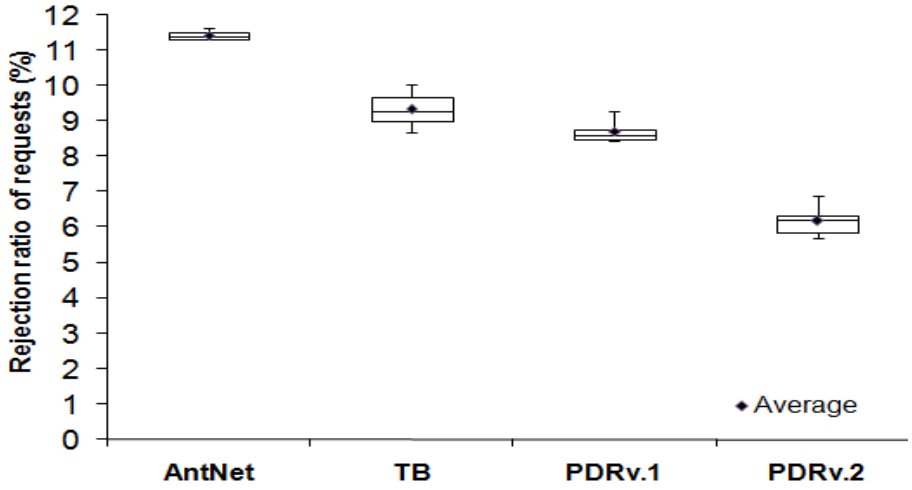

Figure 5.41 The rejection ratio of requests for the HL scenario.

Figure 5.42 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, PDRv.2 algorithm rejects 27.78% less BW than PDRv.1 algorithm, 32.06% less BW than TB algorithm and 44.55% less BW than AntNet algorithm. However, PDRv.1 algorithm rejects 5.93% less bandwidth than TB algorithm and 23.23% less bandwidth than AntNet algorithm.

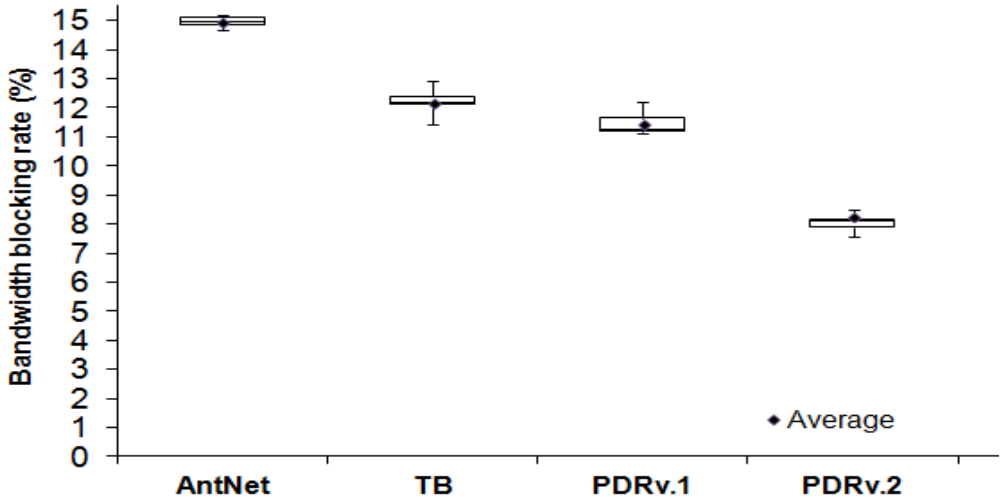

Figure 5.42 The bandwidth blocking rate for the HL scenario.

5.6.3 Real traffic scenario

Figure 5.43 shows the rejection ratio of requests for the real traffic scenario. Based on the results, PDRv.2 algorithm rejects 7.30% less requests than PDRv.1 algorithm, 18.80% less requests than TB algorithm and 29.64% less requests than AntNet algorithm. However, the PDRv.1 algorithm rejects 12.41% less requests than the TB algorithm and 24.10% less requests than the AntNet algorithm.

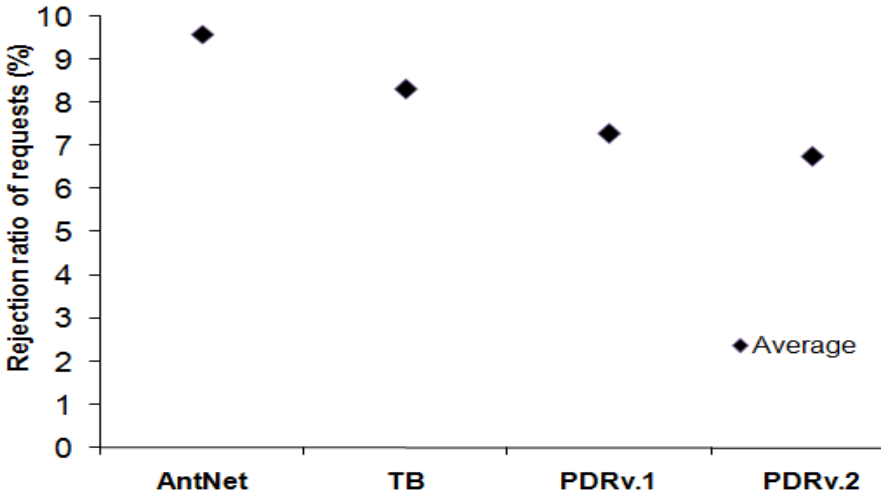

Figure 5.43 The rejection ratio of requests for the real traffic scenario.

Figure 5.44 shows the bandwidth blocking rate for the real traffic scenario. Based on the results, PDRv.2 algorithm rejects 6.13% less BW than PDRv.1 algorithm, 4.21% less BW than the TB algorithm and 11.47% less BW than the AntNet algorithm. However, the PDRv.1 algorithm rejects 4.17% less bandwidth than the TB and 5.68% less bandwidth than the AntNet algorithm.

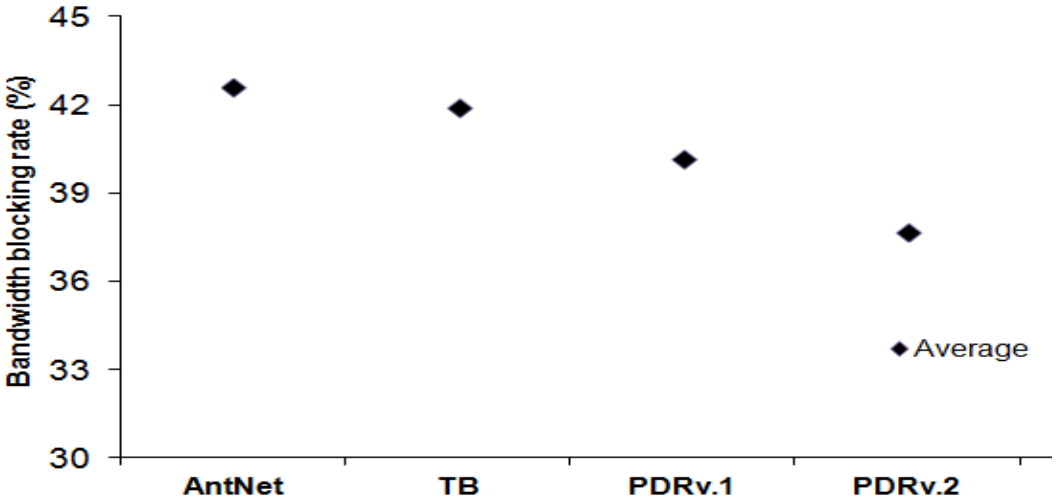

Figure 5.44 The bandwidth blocking rate for the real traffic scenario.

5.6.4 The effect of prediction use

Table 5.12 shows the enhanced performance for the rejection ratio of requests (%) depending on the prediction use. In this section, we aim to study the effect of prediction use. Therefore, we run the PDRv.1 and PDRv.2 algorithms one time without the prediction use (α=0) and another time with the prediction use.

In general, the prediction use within the PDRv.2 algorithm has a positive impact on the routing performance more than the prediction use within PDRv.1 algorithm. The main reason for this enhancement is the new structure of used dynamic FFNN.

Table 5.12 The enhanced performance depending on the prediction use.

Network Load scenario	PDRv.1 algorithm	PDRv.2 algorithm
ML scenario	6.27 (%)	8.37 (%)
HL scenario	3.32 (%)	4.20 (%)

5.7 The PDR algorithm vs. centralized routing algorithms

In this section, the performance of PDR v.1 and PDRv.2 are evaluated based on some test scenarios and discuss the results [27]. During this section, the various versions of PDR algorithm are compared with two different centralized routing algorithms, CSPF and LIOA algorithm. In section 5.7.1, the simulation details are presented. In section 5.7.2, the MIRA topology is considered and the performance of the compared algorithms is tested. In section 5.7.3, the Internet2 topology is considered and the performance of the compared algorithms is tested.

5.7.1 The simulation details

The simulation details are presented in the following points:-

- Simulation workflow:-
  - Two performances parameters are measured:
    - The rejection ratio of requests and
    - The bandwidth blocking rate.
  - This experiment uses the same procedure in section 5.2 in order to focus on the steady state of network traffic.
  - This experiment uses the same procedure of the analysis study in section 5.4 in order to select the best values for the parameters of PDR algorithm, (i.e. *ETh* and α), in all tested scenarios.
  - This experiment uses the same parameter values of the PDRv.1 and PDRv.2 algorithms in section 5.6.1.

5.7.2 The MIRA topology

In the following scenarios, the MIRA topology is considered and the performance of routing algorithms is tested in both ML and HL scenarios.

5.7.2.1 The ML scenario

Figure 5.45 shows the rejection ratio of requests for the ML scenario. The average of results shows that, the PDRv.2 algorithm rejects 32.89% less requests than the CSPF algorithm and 28.77% less than the LIOA algorithm. However, the PDRv.1 algorithm rejects more requests than the CSPF and LIOA algorithms. The PDRv.2 algorithm outperforms the PDRv.1 algorithm because the PDRv.2 algorithm uses

a new adaptive Ant-based mechanism to be able to efficiently distribute the ants on the network topology and accurately discover the best paths. Additionally, the used Ant-based mechanism is incorporated with a new efficient prediction approach, which uses the dynamic FFNN instead of the static FFNN that is used in the previous version.

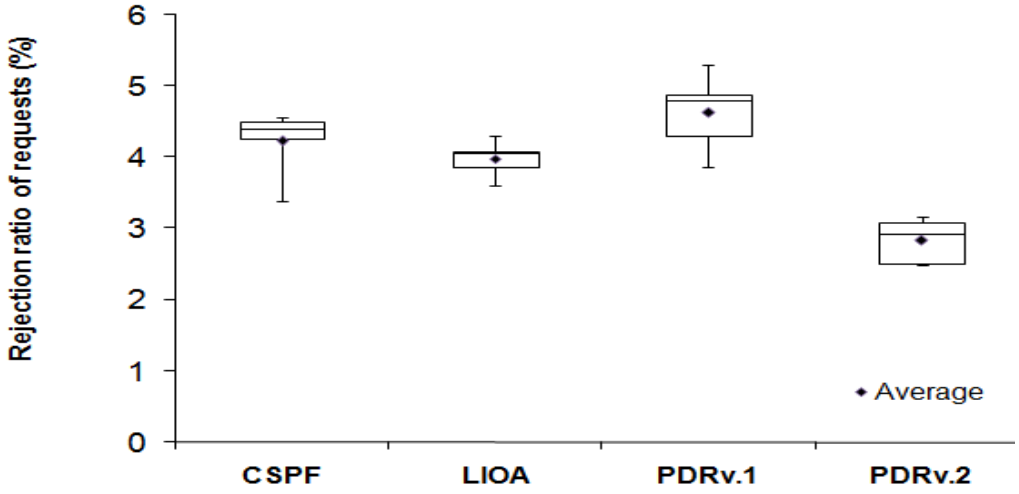

Figure 5.45 The rejection ratio of requests for the ML scenario.

Figure 5.46 shows the bandwidth blocking rate for the ML scenario. The average of results shows that, the PDRv2 algorithm rejects 35.86% less BW than the CSPF algorithm and 31.55% less BW than the LIOA algorithm. However, the PDRv.1 algorithm rejects more bandwidth than the CSPF and LIOA algorithms.

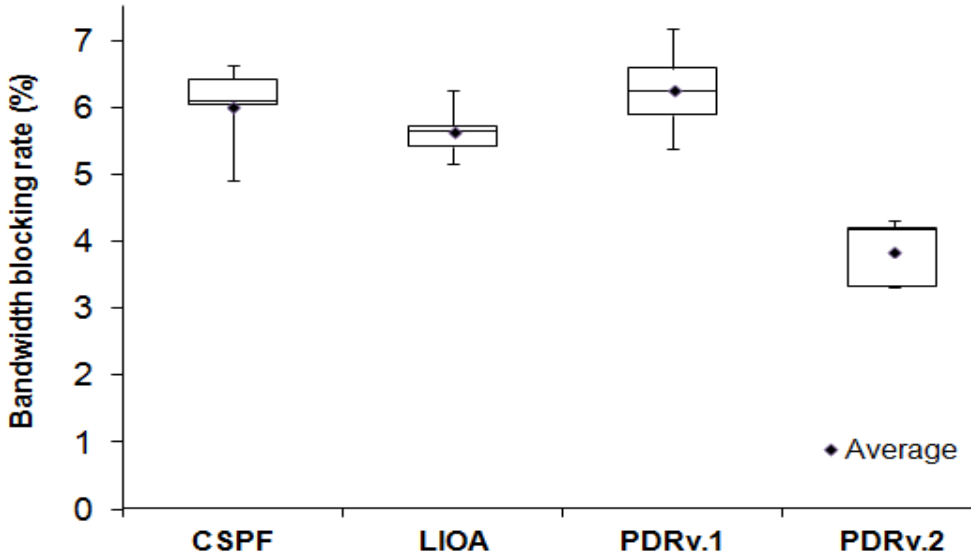

Figure 5.46 The bandwidth blocking rate for the ML scenario.

5.7.2.2 The HL scenario

Figure 5.47 shows the rejection ratio of requests for the HL scenario. The average of results shows that, the PDRv.2 algorithm rejects 19.60% less requests than CSPF algorithm and 16.17% less than the LIOA algorithm. However, the PDRv.1 algorithm rejects more requests than the CSPF and LIOA algorithms.

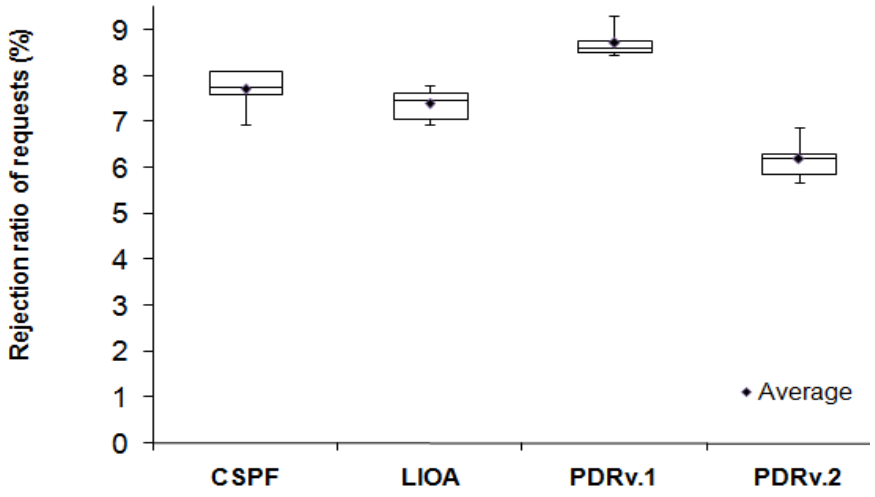

Figure 5.47 The rejection ratio of requests for the HL scenario.

Figure 5.48 shows the bandwidth blocking rate for the HL scenario. The average of results shows that, the PDRv2 algorithm rejects 22.77% less BW than the CSPF algorithm and 19.27% less BW than the LIOA algorithm. However, the PDRv.1 algorithm rejects more bandwidth than the than the CSPF and LIOA algorithms.

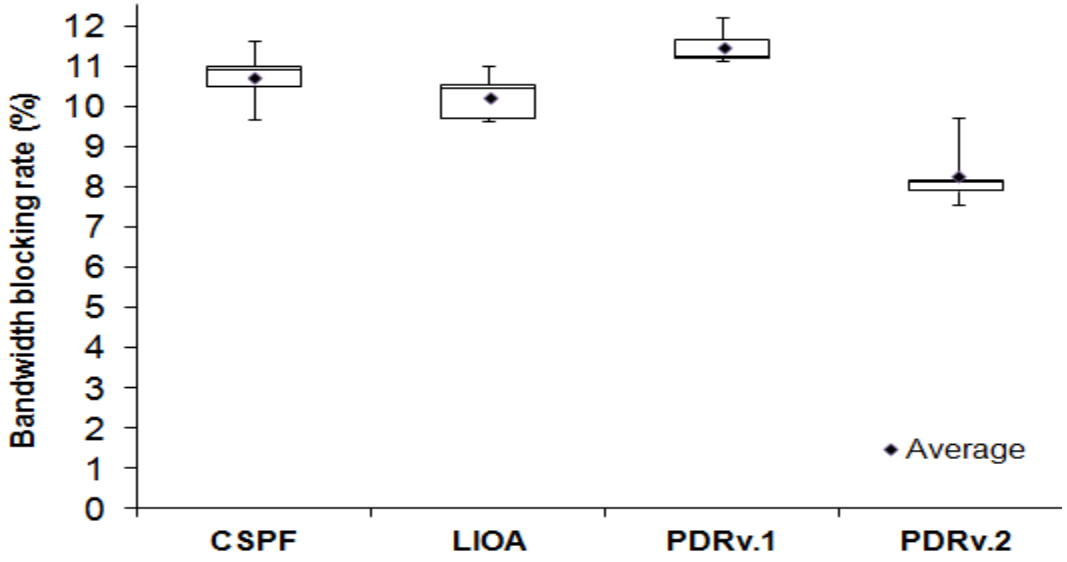

Figure 5.48 The bandwidth blocking rate for the HL scenario.

5.7.3 Real traffic scenario

Figure 5.49 shows the rejection ratio of requests for the real traffic scenario. The average of results shows that, the PDRv.2 algorithm rejects 6.49% less requests than CSPF algorithm and 10% less than the LIOA algorithm. However, the PDRv.1 algorithm rejects more requests than the CSPF and LIOA algorithms.

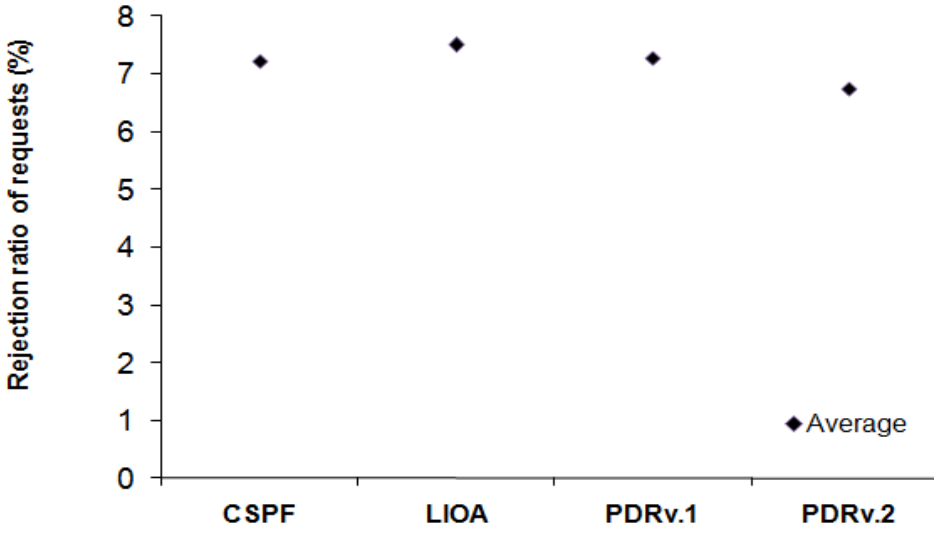

Figure 5.49 The rejection ratio of requests for the real traffic scenario.

Figure 5.50 shows the bandwidth blocking rate for the real traffic scenario. The average of results shows that, the PDRv2 algorithm rejects 6.88% less BW than the CSPF algorithm and 6.47% less BW than the LIOA algorithm. However, the PDRv.1 algorithm rejects more bandwidth than the than the CSPF and LIOA algorithms.

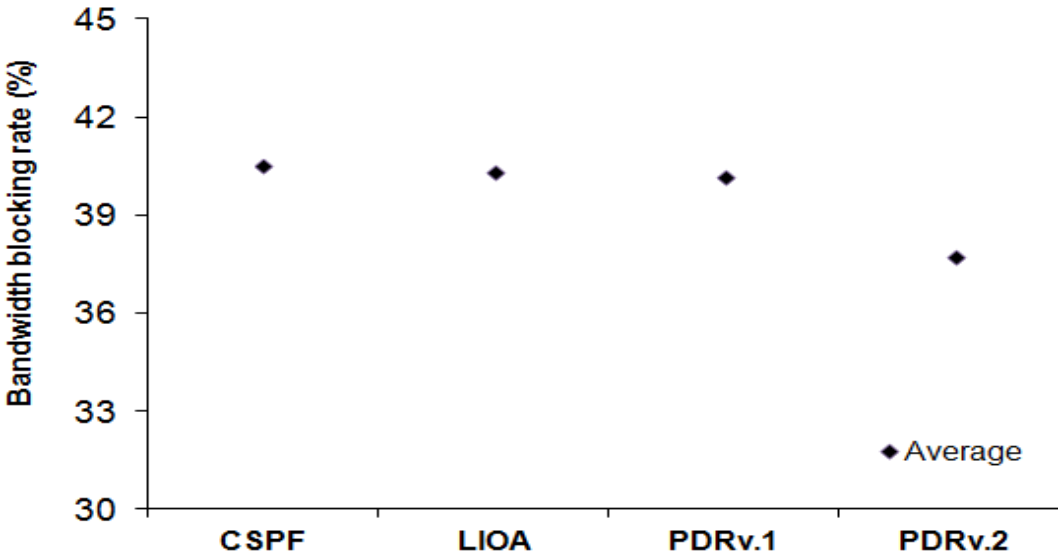

Figure 5.50 The bandwidth blocking rate for the real traffic scenario.

CHAPTER 6 - Conclusions and future work

6.1 Conclusions

In this dissertation, two different algorithms are introduced within the research area of dynamic routing. The first algorithm is routing maintenance algorithm that runs beside the centralized routing algorithm. The second algorithm is a new fully decentralized and self-organized routing algorithm.

6.1.1 The PFLR algorithm

In this dissertation, a new TE routing maintenance algorithm, named Predicting of Future Load-based Routing (PFLR) algorithm, is introduced [22], [23], [24] that can efficiently enhance the performance of dynamic routing algorithms. The PFLR algorithm runs beside any routing algorithm that depends on the available BW information of network links in order to select the best paths between the source and destination pairs.

With the use of PFLR algorithm, the future status of the network link loads will be considered. The considering of future network link loads has a big impact in reducing the interference between the path requests in the future and so reduces the occurrence of network congestions and at the same time leads to increase the network utilization.

The most important feature of PFLR algorithm is the link state (weight) representation. The proposed algorithm combines the predicted link load with the current link load with an effective method in order to optimize the link weights. The idea is to reduce the number of wrong and critical decisions in case of uncertain prediction accuracy. This approach uses an ANN to build an adaptive predictor that predicts future link loads. ANN offers accurate prediction capabilities with different types of network traffic (generated and real traffic) and has the ability to be adaptive. Additionally, the advanced version of PFLR algorithm (PFLRv.2) has the ability to adapt the parameters of prediction model, such as the length of prediction step and the prediction validity period, in order to efficiently estimate the link traffics and so enhance the routing performance.

The performance of the PFLR algorithm is compared with respect to the WSP, CSPF and LIOA algorithms, within four different network topologies, with different traffic types and under different network traffic conditions. In general, the PFLR algorithm performs considerably better than the comparative algorithms with respect to various performance comparison criteria, such as the rejection ratio of requests (In the best case, it rejects 17.45% less requests than the normal algorithms), the bandwidth blocking rate (In the best case, it rejects 17.63% less BW than the normal algorithms) and the rejection ratio of re-routed requests upon link failure scenario.

Additionally, a comparative study between the PFLR algorithm and different estimation-based dynamic routing algorithms are presented. The DF-PI, PSA and PFLR algorithms are bundled with WSP algorithm. After that, they are experimentally compared with each other using generated and real traffic scenarios. In general, the PFRL algorithm performs considerably better than the PSA and DF-PI algorithms with respect to different performance comparison criteria.

According to limitations of PFLR algorithm, there are three issues should be considered during the use of PFLR algorithms. The first issue is that, the predicted available BW information should be also distributed to all network nodes. The distribution for additional information increases the routing overhead a little bit.

The second issue is that, while the network load condition becomes heavier (the rejection ratio of requests is high), the performance enhancement of the routing algorithms using the PFLR algorithm is decreased compared to the lighter load condition scenario. Therefore, the recommended advice is that, it must be a significance improvement for the performance in order to balance the overhead cost of information distribution.

The third issue is that, in case of very wide range for the BW of requests, the ANN requires more time in order to efficiently estimate the future values of network traffic. Although the overhead of ANNs processes is reduced by distributing the predictors on the various network nodes, the cost of ANN training still must be carefully considered (specially, in dynamic routing case).

6.1.2 The PDR algorithm

According to the second contribution in this dissertation, a new fully decentralized and self-organized routing algorithm is proposed, named Prediction-based Decentralized Routing (PDR) algorithm [25], [26], [27]. This algorithm is a member of traffic-aware routing algorithms. In the same time, this algorithm is considered as a new member of Ant Colony Routing (ACR) class. ACR algorithms are inspired from real ants' behaviors which have the ability of discovering the shortest path to a food source from their nest without any knowledge of geometry but with a keen sense of smell. In this approach, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit, so that it has a simpler process and less control parameters.

Using the information of link state helps the routing algorithm to efficiently achieve the BW guarantee of the provided QoS. Additionally, considering the future value of the network link loads leads to reduce the interference between the reserved requests in the future and so reduce the occurrence of network congestions and increases the network utilization.

The PDR algorithm uses similar prediction mechanism to the PFLR algorithm but with local-based implementation. Additionally, the PDR algorithm has the ability to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics.

The performance of our proposed PDR algorithm is compared with the TB and AntNet algorithms, within two different network topologies, with different traffic types and under different network traffic conditions. In general, the proposed algorithm performs considerably better than the comparative algorithms with respect to different performance comparison criteria, such as the rejection ratio of requests (In the best case, it rejects 63.40% less requests than the comparative algorithm), the bandwidth blocking rate (It rejects 62.37% less BW than the comparative algorithms).

The advanced version of PDR algorithm (PDRv.2) outperforms the PDRv.1 algorithm because it uses a new adaptive Ant-based mechanism to be able to efficiently distribute the ants on the network topology and accurately discover the

best paths. Additionally, the used Ant-based mechanism is incorporated with a new efficient prediction approach, which uses the dynamic ANN instead of the static ANN that is used in the previous version. Based on a comparative study between various versions of PDR and various centralized routing algorithms, such as CSPF and LIOA algorithms, PDR v.2 algorithm (In contrast to PDRv.1 algorithm) performs considerably better than various centralized algorithms with respect to various performance comparison criteria, such as the rejection ratio of requests (In the best case, it rejects 32.89% less requests than the traditional centralized algorithm) and the bandwidth blocking rate (It rejects 35.86% less BW than the traditional centralized algorithm).

6.2 Future work

According to the performance study, the performance testing of PFLR and PDR algorithms is planed with more complex network topologies. Also, other generated traffic models and various real traffic demands will be tested. In addition, the testing of other performance criteria will be done. Finally, the comparison of the PFLR and PDR algorithms with other dynamic routing algorithms is planned.

According to the methodology of proposed algorithms, the focus will be much more on the minimizing of computation time for proposed algorithms. In addition, the use of other ANN structure is planned in order to increase the prediction accuracy of proposed predictors. Finally, the focus will be much more on the decentralized routing approaches.

In this dissertation, all the works are only focusing on one type of QoS guarantees (BW guarantee). In the future work, the consideration of other QoS requirements, such as the end-to-end delay, is planned too.

References

[1] Awduche, D.; Chiu, A.; Elwalid, A.; Widjaja, I. & Xiao, X. Overview and Principles of Internet Traffic Engineering. IETF, RFC3272, 2002.

[2] Crawley, E.; Nair, R.; Rajagopalan, B. & Sandick, H. A Framework for QoS-based Routing in the Internet. IETF, RFC2386, 1998.

[3] Medhi, D. & Ramasamy, K. Network Routing: Algorithms, Protocols, and Architectures. Morgan Kaufmann, 2007.

[4] Castineyra, I.; Chiappa, N. & Steenstrup, M. The Nimrod Routing Architecture. IETF, RFC1992, 1996.

[5] Xiao, X. Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective. Morgan Kaufmann, 2008.

[6] Blum, C. & Merkle, D. (Eds.) Swarm Intelligence: Introduction and Applications. Springer, 2008.

[7] Malkin, G. RIP Version 2.IETF, RFC2453, 1998.

[8] Inc. Cisco Systems. Internetworking Tech. Handbook. Cisco Press, 2003.

[9] Sivabalan, M. & Mouftah, H. T. Approaches to Link-State Alternate Path Routing in Connection-Oriented Networks. Int. Symposium on Modeling, Analysis and Simulation of Computer and Telecom. Systems, 92-100, 1998.

[10] Hopps, C. Analysis of an Equal-Cost Multi-Path Algorithm. IETF, RFC2992, 2000.

[11] Pei, G.; Gerla, M. & Chen, T.-W. Fisheye State Routing: A Routing Scheme for Ad HocWireless Networks. IEEE International Conference on Communications, 70-74, 2000.

[12] Crawley, E.; Berger, L.; Berson, S. & Krawczyk, J. A Framework for Integrated Services and RSVP over ATM. IETF, RFC2382, 1998.

[13] Kirov, G. & Lakov, D. Soft Computing Agents for MPLS Networks. Cybernetics and Information Technologies, 2(2), 2002.

[14] Corson, S. & Macker, J. Mobile Ad hoc Networking Routing Protocol Performance Issues and Evaluation Considerations. IETF, RFC2501, 1999.

[15] Varadarajan, S.; Ramakrishnan, N. & Thirunavukkarasu, M. Reinforcing reachable routes. Computer Networks, 43(3), 389-416, 2003.

[16] Clausen, T. & Jacquet, P. Optimized Link State Routing Protocol (OLSR).IETF, RFC3626, 2003.

[17] Afzal, S. R.; Biswas, S.; Koh, J.; Raza, T.; Lee, G. & Kim, D. RSRP: A Robust Secure Routing Protocol for Mobile Ad Hoc Networks. IEEE Wireless Communications and Networking Conf. WCNC, 2313-2318, 2008.

[18] Graziani, R. & Johnson, A. Routing Protocols and Concepts, CCNA Exploration Companion. Guide Cisco Press, 2007.

[19] Callon, R. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. IETF, RFC1159, 1990.

[20] Chandra, R. & Scudder, J. Capabilities Advertisement with BGP-4.IETF, RFC2842, 2000.

[21] Rutgers, C. L. H. An Introduction to IGRP. Cisco Press, 1991.

[22] Turky, A. A. & Mitschele-Thiel, A. MPLS Online Routing Optimization Using Prediction. Network Control and Optimization, Lecture Notes in Computer Science5425, Springer, 45-52, 2008.

[23] Turky, A. A. & Mitschele-Thiel, A. Use of Load Prediction Mechanism for Dynamic Routing Optimization. IEEE Symposium on Computers and Communications ISCC, 782-786, 2009.

[24] Turky, A. A. & Cap, C. H. An Efficient Routing Maintenance Mechanism Using Adaptive Traffic Prediction. IEEE, IET Int. Symposium on Communication Systems, Networks and Digital Signal Processing (IEEE CSNDSP), 1-6, 2012.

[25] Turky, A. A. & Mitschele-Thiel, A. Prediction-based Decentralized Routing Algorithm. Electronic Communications of the EASST, 17, 2009.

[26] Turky, A. A.; Liers, F. & Mitschele-Thiel, A. Self-optimizing Mechanism for Prediction-based Decentralized Routing.7th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine) 2010, LNICST 74, 454-468, 2011.

[27] Turky, A. A. & Cap, C. H. An effective QoS Providing Mechanism Using Prediction-based Decentralized Routing. IEEE International Conference on Advanced Information Networking and Applications (IEEE AINA), 1089-1096, 2013.

[28] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. Introduction to Algorithms. MIT Press and McGraw-Hill, 2001.

[29] Dijkstra, E. W. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik, 1, 269-271, 1959.

[30] Black, P. E. Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology, 2008.

[31] Fredman, M. L. & Tarjan, R. E. Fibonacci Heaps and Their Uses in Improved Network Optimization. Journal of the ACM, 34(3), 596-615, 1987.

[32] Bellman, R. On A routing Problem. Quarterly of Applied Mathematics, 16, 87-90, 1958.

[33] Ford, L. R. & Fulkerson, D. R. Flows in Networks. Princeton University Press, 1962.

[34] Guerin, R. A.; Orda, A. & Williams, D. QoS Routing Mechanisms and OSPF Extensions. IEEE Global Telecom. GLOBECOM, 3, 1903-1908, 1997.

[35] Crawley, E.; Nair, R.; Rajagopalan, B. & Sandick, H. A Framework for QoS-based Routing in the Internet. IETF, RFC2386, 1998.

[36] Kar, K.; Kodialam, M. & Lakshman, T. V. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE J. Selected Areas in Comm., 18(12), 2566-257, 2000.

[37] Rosen, E.; Viswanathan, A. & Callon, R. Multiprotocol Label Switching Architecture. IETF, RFC3031, 2001.

[38] Muthukrishnan, K. &Malis, A. A Core MPLS IP VPN Architecture IETF, RFC2917, 2000.

[39] Awduche, D.; Berger, L.; Gan, D.; Li, T.; Srinivasan, V. & Swallow, G. RSVP-TE: Extensions to RSVP for LSP Tunnels. IETF, RFC3209, 2001.

[40] Jamoussi, B.; Andersson, L.; Callon, R.; Dantu, R.; Wu, L.; Doolan, P.; Worster, T.; Feldman, N.; Fredette, A.; Girish, M.; Gray, E.; Heinanen, J.; Kilty, T. &Malis, A. Constraint-Based LSP Setup using LDP. IETF, RFC3212, 2002.

[41] Ahuja, R. K.; Magnanti, T. L. & Orlin, J. B. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, 1993.

[42] Goldberg, A. V. & Tarjan, R. E. A New Approach to the Maximum-Flow Problem. The eighteenth annual ACM symposium on Theory of computing, 136-146, 1986.

[43] Boutaba, R.; Szeto, W. & Iraqi, Y. DORA: Efficient Routing for MPLS Traffic Engineering. Journal of Network and Systems Management, 10(3), 309-325, 2002.

[44] Suri, S.; Waldvogel, M. & Warkhede, P. R. Profile-Based Routing: A New Framework for MPLS Traffic Engineering. Quality of Future Internet Services, Lecture Notes in Computer Science2156, 138-157, 2001.

[45] Bagula, A. B.; Botha, M. & Krzesinski, A. E. Online Traffic Engineering: the Least Interference. Optimization Algorithm. IEEE International Conference on Communications, 2, 1232-1236, 2004.

[46] Yin, C. S.; Ling, T. & Yaacob, M. H. Hop-by-hop QoS Routing using Statistical Distribution-free Approach. Malaysian Journal of Computer Science, 18(2), 28-37, 2005.

[47] Hahn, G. J. & Meeker, W. Q. Statistical Intervals: A Guide for Practitioners. Wiley-Interscience, 1991.

[48] Apostopoloulos, G.; Guerin, R.; Kamat, S. & Tripathi, S. Improving QoS Routing Performance under Inaccurate Link State Information. The 16th International Teletrac Congress, 1351-1362, 1999.

[49] Anjali, T.; Scoglioand, C. & Oliveira, J. C. A New Path Selection Algorithm for MPLS Networks Based on Available Bandwidth Estimation. 3rd int. conf. on quality of future internet services and internet charging, 2002.

[50] Jain, M. &Dovrolis, C. Pathload: A Measurement Tool for End-to-End Available Bandwidth. Passive and Active Measurements Workshop, 14-25, 2002.

[51] Claise, B. Cisco Systems NetFlow Services Export Version 9 IETF, RFC3954, 2004.

[52] Case, J.; Mundy, R.; Partain, D. & Stewart, B. Introduction and Applicability Statements for Internet-Standard Management Framework IETF, RFC3410, 2002.

[53] Tobi Oetiker's MRTG - The Multi Router Traffic Grapher, http://oss.oetiker.ch/mrtg/index.en.html.

[54] Hazewinkel, M. Wiener–Hopf method, Springer, Encyclopaedia of Mathematics, http://eom.springer.de/W/w097910.htm, 2002.

[55] Vaidyanathan, P. P. The Theory of Linear Prediction. Synthesis Lectures on Signal Processing. Morgan and Claypool, 2009.

[56] Olariu, S. & Zomaya, A. Y. Handbook of Bioinspired Algorithms and Applications. Chapman and Hall, 2005.

[57] Dorigo, M.; Caro, G. D. & Gambardella, L. M. Ant Algorithms for Discrete Optimization. Artificial Life, 5(2), 137-172, 1999.

[58] Poli, R. Analysis of the publications on the applications of particle swarm optimization. Journal of Artificial Evolution and Applications, 1-10, 2008.

[59] S.Camazine; Deneubourg, J.-L.; Franks, N. R.; Sneyd, J.; Theraulaz, G. & Bonabeau, E. (Eds.) Self-Organization in Biological Systems Princeton. University Press, 2001.

[60] Bonabeau, E.; Dorigo, M. & Théraulaz, G. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.

[61] Michel, R. & Middendorf, M. An Island Model Based Ant System with Lookahead for the Shortest Super sequence Problem. The 5th International Conference on Parallel Problem Solving from Nature, 692-701, 1998.

[62] Stutzle, T. & Hoos, H. MAX-MIN Ant System and Local Search for the TS Problem. IEEE Int. Evolutionary Computation Conf, 309-314, 1997.

[63] Caro, G. D. & Dorigo, M. AntNet: A Mobile Agents Approach to Adaptive Routing. IRIDIA, Brussels University, 1997.

[64] Donati, A. V.; Darley, V. & Ramachandran, B. An Ant-Bidding Algorithm for Multistage Flowshop Scheduling Problem: Optimization and Phase Transitions. Springer, Advances in Metaheuristics for Hard Optimization, 111-136, 2008.

[65] Donati, A. V.; Montemanni, R.; Casagrande, N. & Gambardella, L. M. Time Dependent Vehicle Routing Problem with a Multi Ant Colony System. European Journal of Operational Research, 185(3), 1174-1191, 2008.

[66] Lourenço, H. & Serra, D. Adaptive Search Heuristics for The Generalized Assignment Problem. Mathware and Soft Computing, 7(1), 1-24, 2002.

[67] Dorigo, M.; Maniezzo, V. & Colorni, A. Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics, 26(1), 29-41, 1996.

[68] Caro, G. D. & Dorigo, M. AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research (JAIR), 9, 317-365, 1998.

[69] Caro, G. D. Ant Colony Optimization and its application to adaptive routing in telecommunication networks. Polytechnic School, University of Brussels, 2004.

[70] Boyan, J. & Littman, M. Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks. Advances in Neural Information Processing Systems 6 (NIPS6), 671-678, 1994.

[71] Choi, S. & Yeung, D.-Y. Predictive Q-routing: A Memory-based Reinforcement Learning Approach to Adaptive Trace Control. Advances in Neural Information Processing Systems 8 (NIPS8), 945-951, 1996.

[72] Yun, H. & Z.Heywood, A. N. Intelligent Ants for Adaptive Network Routing. Conf. on Comm. Networks and Services Research, 255-261, 2004.

[73] Gabber, E. & Smith, M. A. Trail Blazer: A Routing Algorithm Inspired by Ants. The IEEE International Conference on Network Protocols, 36-47, 2004.

[74] Stergiou, C. & Siganos, D. Neural Networks and their Uses. SURPRISE 96 Journal, 4, 1996.

[75] McCulloch, W. & Pitts,W. A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5, 115-133, 1943.

[76] Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, 65(6), 386-408, 1958.

[77] Halici, U. Lecture Notes on Introduction to Neural Networks, Information Institute, Middle East Technical University, 2001.

[78] Gupta, M. M.; Jin, Liang & Homma, N. Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory. Wiley-IEEE Press, 2003.

[79] Eberhart, R. C. & Shi, Y. Computational Intelligence: Concepts to Implementation. Morgan Kaufmann, 2007.

[80] Baldi, P. F. & Hornik, K. Learning in Linear Neural Networks: A Survey. IEEE Trans. On Neural Networks, 6(4), 837-858, 1995.

[81] Breiman, L. & Friedman. J.H. Function approximation using ramps. Snowbird Workshop on Machines that Learn, 1994.

[82] Valova, I.; Gueorguieva, N. & Kempka, M. Feed-Forward Neural Networks With Sigmoidal And Radial Basis Functions Hidden Neurons, Systems, Man and Cybernetics, 2, 1630-1635, 2003.

[83] Shilling, R.J.; Carroll, J.J. & Al-Ajlouni, A.F. Approximation of nonlinear Systems with Radial Basis Function Neural Networks. IEEE Transactions on Neural Networks, 12(1), 1-15, 2001.

[84] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. The National Academy of Sciences of the USA, 79(8), 2554-2558, 1982.

[85] Kang, S. & Isik, C. Partially Connected Feed Forward Neural Networks Structured by Input Types. IEEE TRANSACTIONS ON NEURAL NETWORKS, 16(1), 175-184, 2005.

[86] Werbos, P.J. Back-propagation Through Time: what It Does and How to Do It. Special Issue on Neural Networks, 78(10), 1550-1560, 1990.

[87] Weibel, A.; Hanazawa, T.; Hinton, G. & Lang, K. Phenomena Recognition Using Time-Delay Neural Networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(1), 328-339, 1989.

[88] Jordan, M.I. Serial order: A parallel distributed processing approach. Tech. Rep. No.8604, University of California, Institute for Cognitive Science, 1986.

[89] Elman, J.L. Finding Structure in Time. Cognitive Science, 14(2), 179-211, 1990.

[90] Kohonen, T. Self-Organizing Maps. Information Sciences, Springer Series, 30(1), 2001.

[91] Martinetz, T. & Schulten, K. A Neural Gas Network Learns Topologies. Artificial Neural Networks, Elsevier, 397-402, 1991.

[92] Ackley, D. H.; Hinton, G. E. & Sejnowski, T. J. A Learning Algorithm for Boltzmann Machines. Cognitive Science, 9(1), 147–169, 1985.

[93] Wang, D.; Gao, Y. & Yang, P. Applying Neural Network to Reinforcement Learning in Continuous Spaces. International Symposium on Neural Networks (ISNN), 621-626, 2005.

[94] MAKRIDAKIS, S.; WEELWRIGHT, S. & HYNDMAN, R. Forecasting: Methods and Applications, John Wiley & Sons, 1998.

[95] Barabas, M.; Boanea, G.; Rus, A. B.; Dobrota, V. & Domingo-Pascual, J. Evaluation of Network Traffic Prediction Based on Neural Networks with Multi-task Learning and Multiresolution Decomposition, IEEE ICCP, 95–102, 2011.

[96] Eswaradass, A.; Sun, X.H. & Wu, M. Network Bandwidth Predictor (NBP): A System for Online Network performance Forecasting. IEEE International Symposium on Cluster Computing and the Grid, 265-268, 2006.

[97] Cortez, P.; Rio, M.; Rocha, M. & Sousa, P. Forecasting Internet Traffic by Neural Networks under Univariate and Multivariate Strategies. In. Artificial Neural Networks and Intelligent Information, 61-71, 2008.

[98] Chabaa, S. & Antari, J. Identification and Prediction of Internet Traffic Using Artificial Neural Networks. J. of Intelligent Learning Sys. and Applications, 2(3), 147-155,2010.

[99] Dharmadhikari, V. B. & Gavade, J. D. An NN Approach for MPEG Video Traffic Prediction, 2nd International Conference on Software Technology and Engineering, 57–61, USA, 2010.

[100] Cortez, P.; Ri, M & Sousa, P. Multi-scale Internet traffic forecasting using neural networks and time series methods. Expert Systems Journal, Wiley Press, 29(2), 143–155, 2012.

[101] HANSEGAWA, M.; WU, G. & MIZUNO, M. Applications of nonlinear prediction methods to the internet traffic, IEEE International Symposium on Circuits and Systems. Australia, 169–172, 2001.

[102] PARK, K. & WILLINGER, W. Self-Similar Network Traffic and Performance Evaluation, WILEY, 2000.

[103] Karagiannis, T.; Molle, M. & Faloutsos, M. Long-range dependence - Ten years of Internet traffic modeling. IEEE Internet Computing, (8), 57– 64, 2004.

[104] Lambiotte, R.; Tabourier, L. & Delvenne, J. Burstiness and Spreading on Temporal Networks, The European Physical Journal B, 86(7), 2003.

[105] Hagan, M.T.; Demuth, H.B & Beale, M.H. Neural Network Design. PWS Publishing, 1996.

[106] Cortez, P.; Miguel, R.; Rocha, M. & Sousa, P. Internet Traffic Forecasting using Neural Networks. International Joint Conference on Neural Networks 4942-4949, 2006

[107] Microsoft Visual Studio, http://www.Microsoft.com/visualstudio.

[108] Neural Network Toolbox, MATLAP V.7, http://www.mathworks.com /products/neuralnet.

[109] Maesschalck, S.; Colle, D.; Lievens, D. & et al. Pan-European Optical Transport Networks: an Availability-based Comparison. Photonic Network Communications, 5(3), 203-225, 2003.

[110] The Internet2 Network Topology, http://www.internet2.edu/network.

[111] The Géant Network Topology, http://geant3.archive.geant.net/Network.

[112] Hendling, K.; Losert, T.; Huber, W. & Jandl, M. Interference Minimizing Bandwidth Guaranteed On-line Routing Algorithm for Traffic Engineering. The 12th IEEE International Conf. of Networks, 497-503, 2004.

[113] Hendling,K.; Losert,T. & Jandl, M. An Intelligent Interference-Minimizing Routing Algorithm. Intelligent Systems at the Service of Mankind 2(1), 187-204, 2006.

[114] Fodor, P.; Enyedi, G. & Cinkler, T. FMIRA: A Fast and Efficient Traffic Engineering Method for Transport Networks. MPLS/GMPLS Workshop, 129-140, 2006.

[115] Dahai, X.; Mung, C. & Rexford, J. Link-State Routing with Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering 2008. The 27th IEEE Conference on Computer Communications INFOCOM, 466-474, 2008.

[116] Hou, M.; Wang, D; Xu,M. & Yang, J. Selective Protection: A Cost-Efficient Backup Scheme for Link State Routing. The 29th Int'l Conference on Distributed Computing Systems, 68-75, 2009.

[117] Muscariello, L.; Perino, D. & Nardelli, B. Towards real implementations of dynamic robust routing exploiting path diversity. IEEE Ultra Modern Telecommunications & Workshops, 1-6, 2009.

[118] The Internet2 Observatory Data Collections, http://www.internet2.edu/observatory/archive/.

[119] Uhlig, S.; Quoitin, B.; Balon, S. & Lepropre, J. Providing public intradomain traffic matrices to the research community. ACM SIGCOMM Computer Communication Review, 36(1), January 2006.

[120] Marzo, J. L.; Calle, E. & Vil1a, P. Performance Evaluation of Minimum Interference Routing in Network Scenarios with Protection Requirements. Elsevier, Computer Communications, 30 (1), 3161–3168, 2007

[121] Gnanasivam, P. Telecommunication Switching and Networks. New Age Publications, 2008.

[122] Resnick, S.I. Heavy-tail phenomena, Probabilistic and Statistical Modeling. Springer Series in Operations Research and Financial Engineering, 2007.

[123] Willinger, W.; Taqqu, M.; Sherman, R. & Wilson, D. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. IEEE/ACM Transactions on Networking, 5(1), 71-86, 1997.

[124] Taqqu, M. S.; Willinger, W. & Sherman, R. Proof of a fundamental result in self-similar traffic modeling. ACM/SIGCOMM Computer Communication Review, 27, 5-23, 1997.

[125] Staehle, D.; Leibnitz, K. & Tran-Gia, P. Source Traffic Modeling of Wireless Applications. Technical report (261). University of Würzburg, 2000.

[126] Dombry, C. & Kaj, Ingemar. The on–off network traffic model under intermediate scaling. Springer, Queueing Systems, 69(1), 29-44, 2001.

[127] Chimeh, J.D.; Hakka, M. & Alavian, S.A. Internet Traffic and Capacity Evaluation in UMTS Downlink Evaluation in UMTS Downlink. Future Generation Communication n and Networking, 547-552, 2007.

[128] Shin, J. & Pinkston, TM. The Performance of Routing Algorithms under Bursty Traffic Loads. Int. Conf. on Parallel and Distributed Processing Techniques and Applications, 2003.

[129] Choi, H. & Limb, J. O. A behavioral model of web traffic. Network Protocols, (ICNP ’99), 327–334, 1999.

[130] Mah, Bruce A. An empirical model of http network traffic. In IEEE INFOCOM 97, 2, 592–600, 1997.

[131] Tran-Gia, P.; Leibnitz, K. & Staehle, D. Source traffic modeling of wireless applications. International Journal of Electronics and Communications, 55(1), 27–36, 2000.

[132] Zhuang, J.; Jalloul, L.; Novak, R. & Park, J. IEEE 802.16m Evaluation Methodology Document (EMD). IEEE 802.16 Contribution 802.16m-08/004r5, January 2009.

[133] WiMAX System Evaluation Methodology, December 2007.

[134] 3GPP2 Contribution C.R1002-0, CDMA2000 Evaluation Methodology, December 2004.

[135] Crovella , Mark E. & Bestavros, Azer. Self-similarity in World Wide Web traffic: evidence and possible causes. In Proceedings of ACM SIGMETRICS, 160-169, 1996.

[136] Rolland, C.; Ridoux, J. & Baynat, B. Hierarchical Models for Web Traffic on CDMA-1xRTT Networks. Technical Report, LIP6 - Equipe NPA, Université Pierre & Marie Curie (Paris VI), http://www-rp.lip6.fr/rolland/techreport.pdf. 15 pages, June 2006.

[137] TOTEM Toolbox. A Toolbox for Traffic Engineering Methods. http://totem.info.ucl.ac.be, 2005.

[138] McGill, R.; Tukey, John W. & Larsen, A. Wayne.(February 1978). Variations of Box Plots. The American Statistician, 32 (1), 12-16, 1978.

[139] Hurst, H.E. Long-term storage of reservoirs: an experimental study. Transactions of the American Society of Civil Engineers, 116(1), 770-799, 1951.

Appendix A - Sensitivity analysis of algorithm parameters

This appendix presents the analysis studies of PFLR and PDR algorithms that are performed in order to select the best values of parameters. The first part covers the analysis study of PFLRv.1 algorithm in all scenarios. The second part focuses on PFLRv.2 algorithm. Finally, the analysis study of PDR algorithm is presented.

A.1 The PFLR v.1 algorithm

In the next experiments, analysis studies are performed on two thousands of requests, which are requested after the first four thousands to focus on the steady state, in order to select the best values of *WS* and α parameters. The aim is to select the best combination of *WS* and α parameters that have the least average of rejection ratio.

A.1.1 The MIRA topology

A.1.1.1 The ML scenario

Figure A.1 shows the average of rejection ratio for requests in the ML scenario using the WSP_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the WSP_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 7 and α equals 0.15.

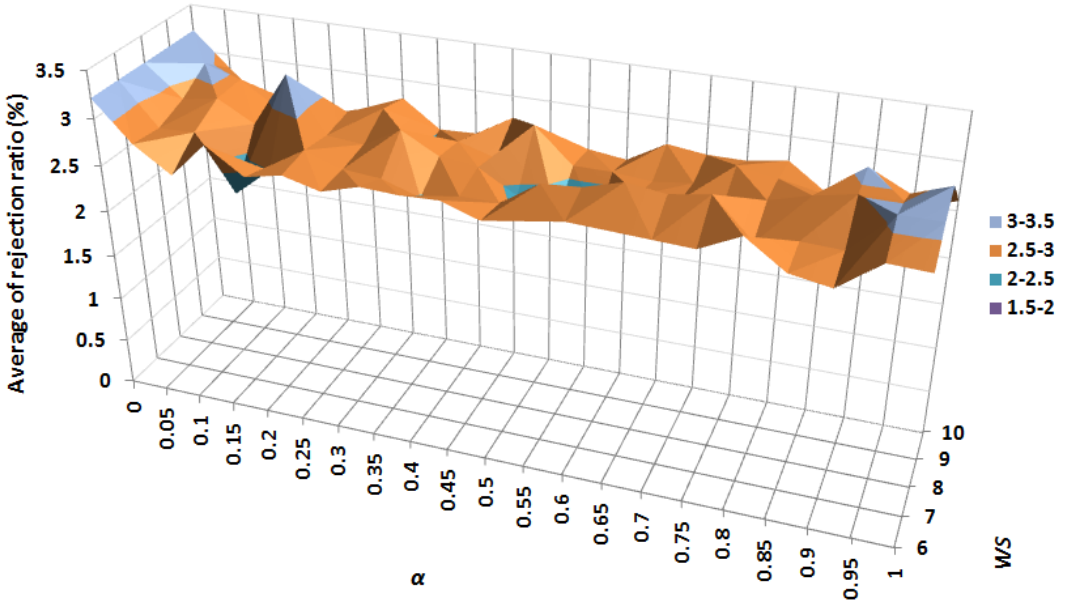

Figure A.1 Average of rejection ratio for the ML scenario (WSP_PFLRv.1).

Figure A.2 shows the average of rejection ratio for requests in the ML scenario using the CSPF_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the CSPF_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 7 and α equals 0.15

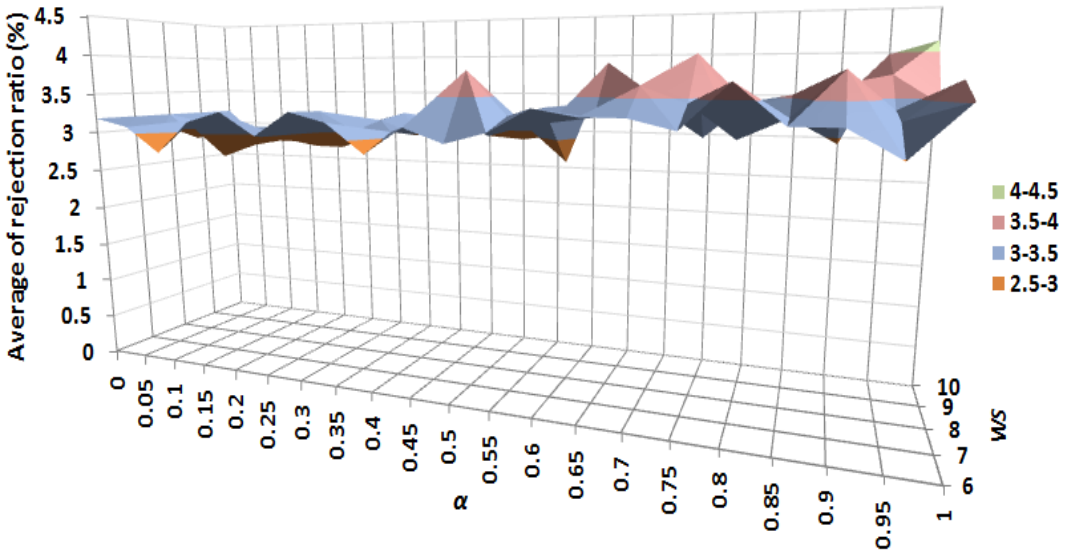

Figure A.2 Average of rejection ratio for the ML scenario (CSPF_PFLRv.1).

Figure A.3 shows the average of rejection ratio for requests in the ML scenario using the LIOA_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the LIOA_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 8 and α equals 0.2.

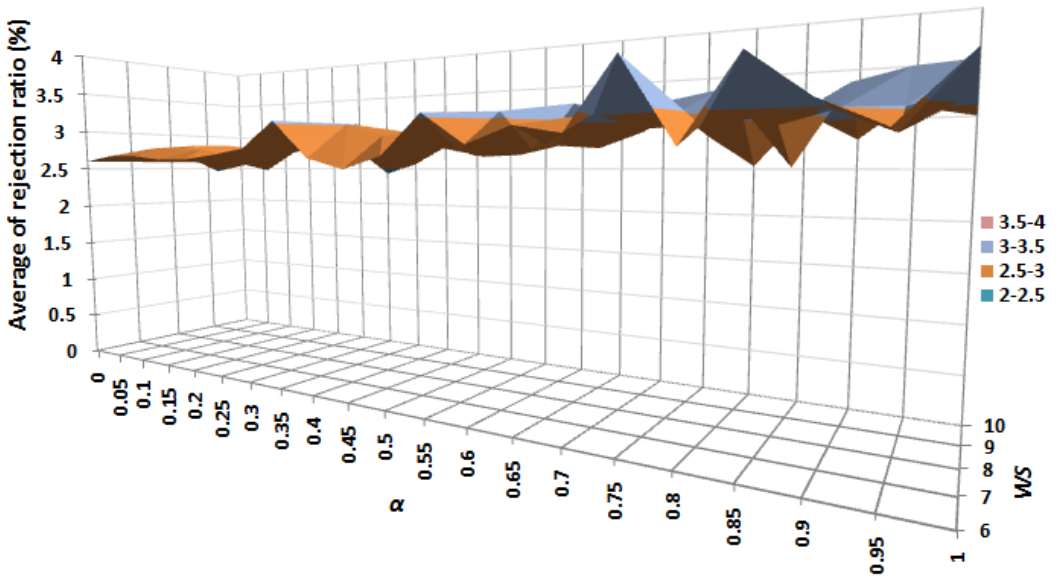

Figure A.3 Average of rejection ratio for the ML scenario (LIOA_PFLRv.1).

A.1.1.2 The HL scenario

Figure A.4 shows the average of rejection ratio for requests in the HL scenario using the WSP_PFLRv.1 algorithm with different combination of *WS* and *α* parameters. The result shows that, the WSP_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 7 and *α* equals 0.1.

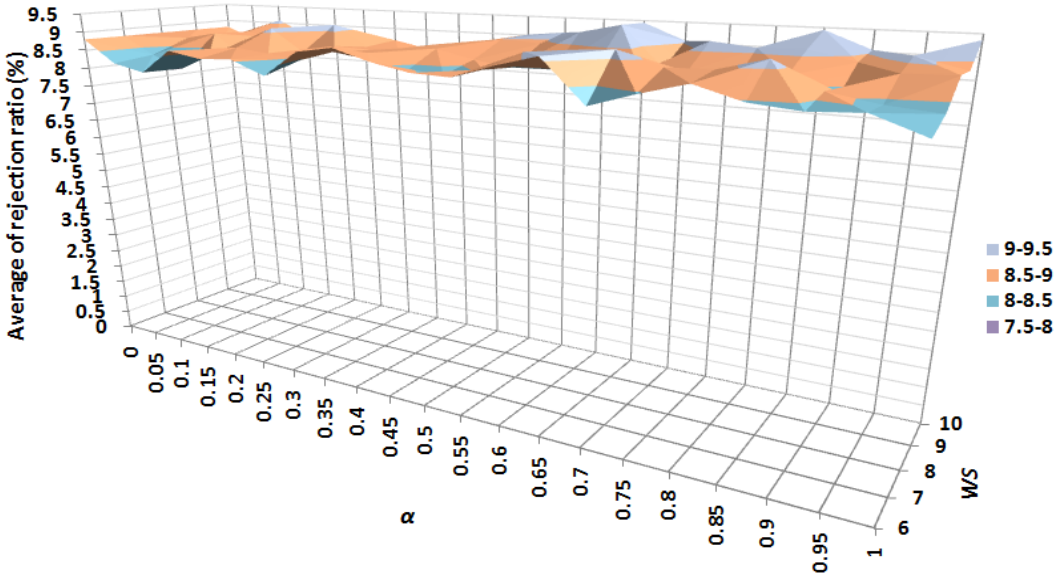

Figure A.4 Average of rejection ratio for the HL scenario (WSP_PFLRv.1).

Figure A.5 shows the average of rejection ratio for requests in the HL scenario using the CSPF_PFLRv.1 algorithm with different combination of *WS* and *α* parameters. The result shows that, the CSPF_PFLRv.1 algorithm has the least average of rejection ratio when *WS*equals8 and *α* equals 0.05.

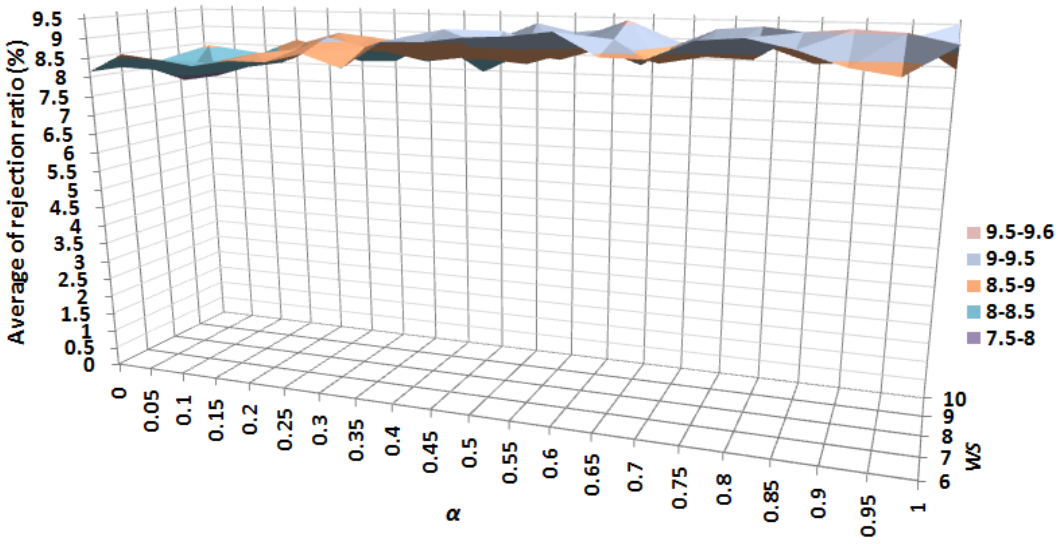

Figure A.5 Average of rejection ratio for the HL scenario (CSPF_PFLRv.1).

Figure A.6 shows the average of rejection ratio for requests in the HL scenario using the LIOA_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the LIOA_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals10 and α equals 0.15.

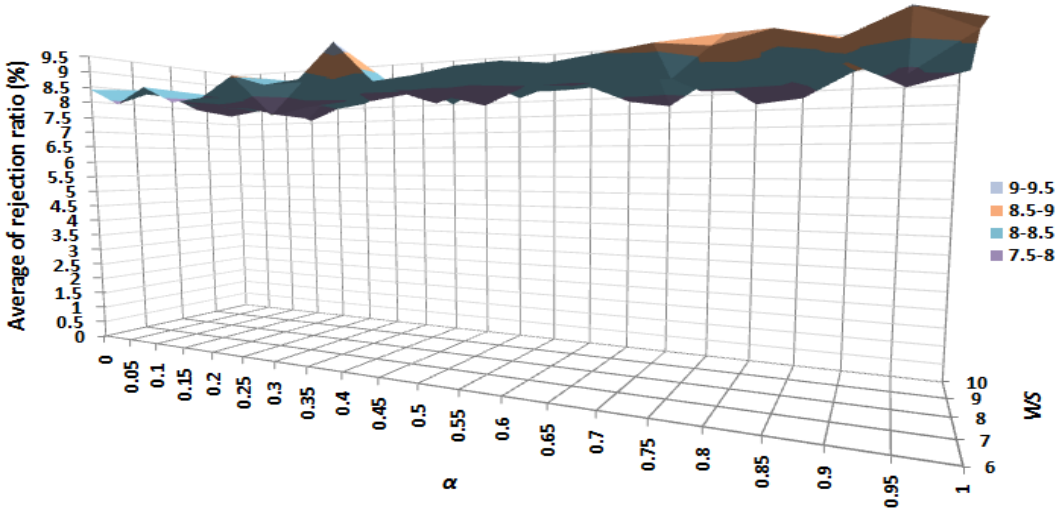

Figure A.6 Average of rejection ratio for the HL scenario (LIOA_PFLRv.1).

A.1.2 The COST266bt topology

A.1.2.1 The ML scenario

Figure A.7 shows the average of rejection ratio for requests in the ML scenario using the WSP_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the WSP_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals7 and α equals 0.2.

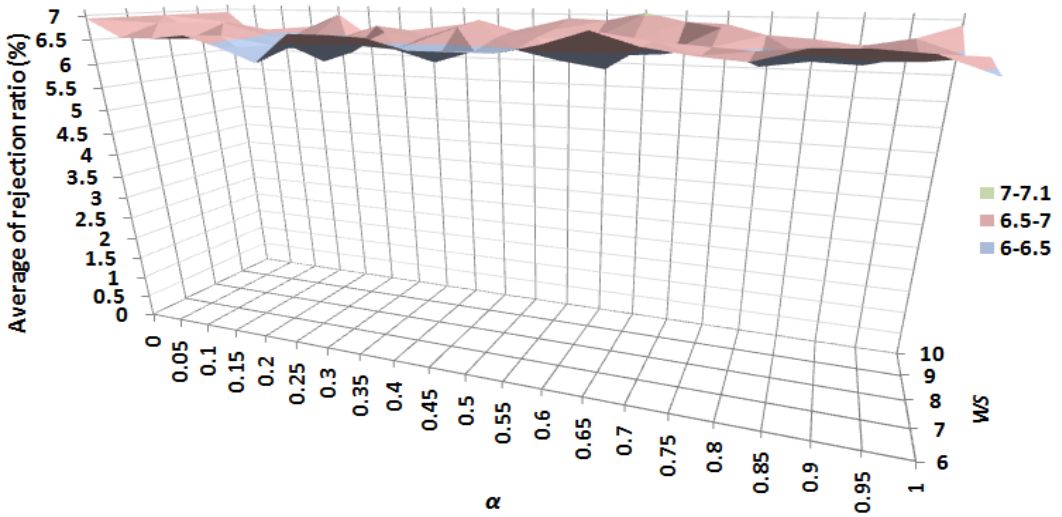

Figure A.7 Average of rejection ratio for the ML scenario (WSP_PFLRv.1).

Figure A.8 shows the average of rejection ratio for requests in the ML scenario using the CSPF_PFLRv.1 algorithm with different combination of *WS* and *α* parameters. The result shows that, the CSPF_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 6 and *α* equals 0.15.

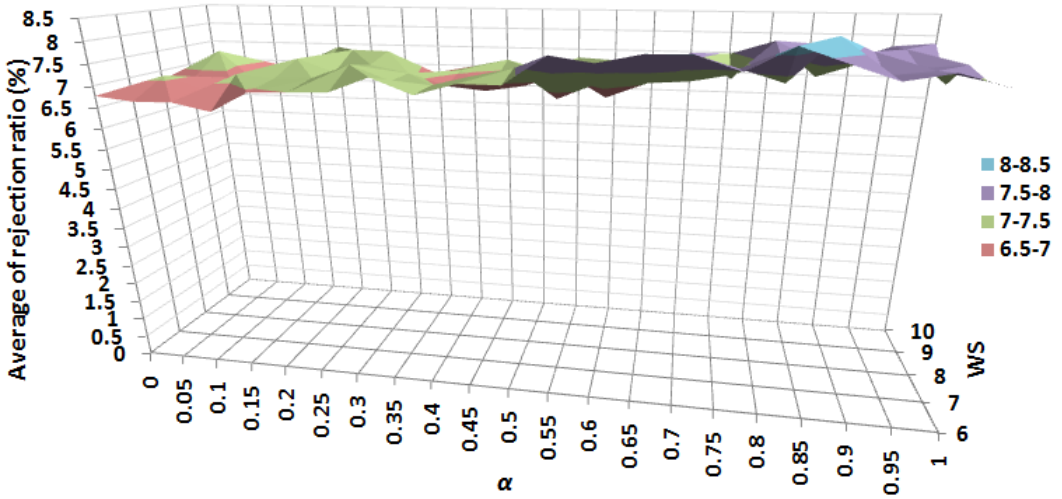

Figure A.8 Average of rejection ratio for the ML scenario (CSPF_PFLRv.1).

A.1.2.2 The HL scenario

Figure A.9 shows the average of rejection ratio for requests in the HL scenario using the WSP_PFLRv.1 algorithm with different combination of *WS* and *α* parameters. The result shows that, the WSP_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 10 and *α* equals 0.1.

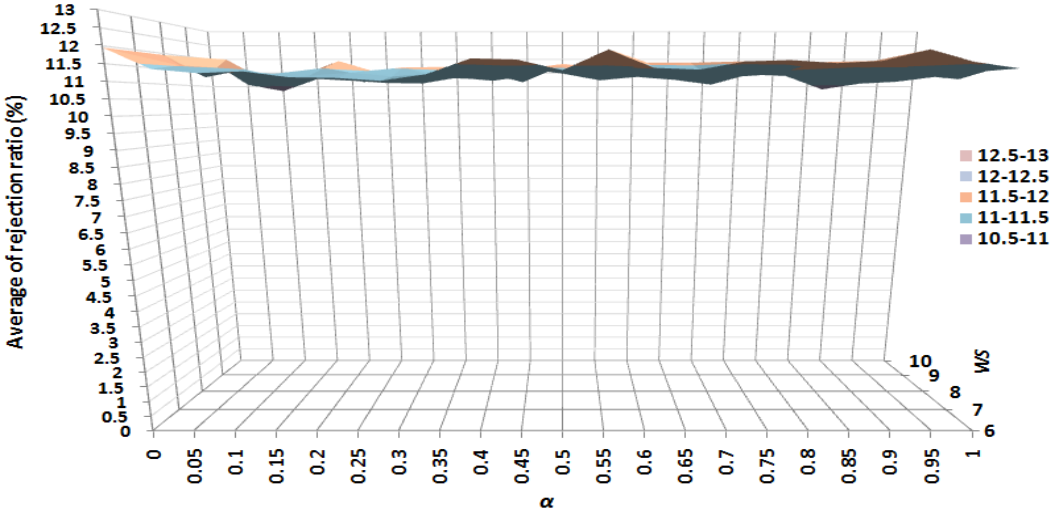

Figure A.9 Average of rejection ratio for the HL scenario (WSP_PFLRv.1).

Figure A.10 shows the average of rejection ratio for requests in the HL scenario using the CSPF_PFLRv.1 algorithm with different combination of *WS* and *α*

parameters. The result shows that, the CSPF_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 7 and α equals 0.4.

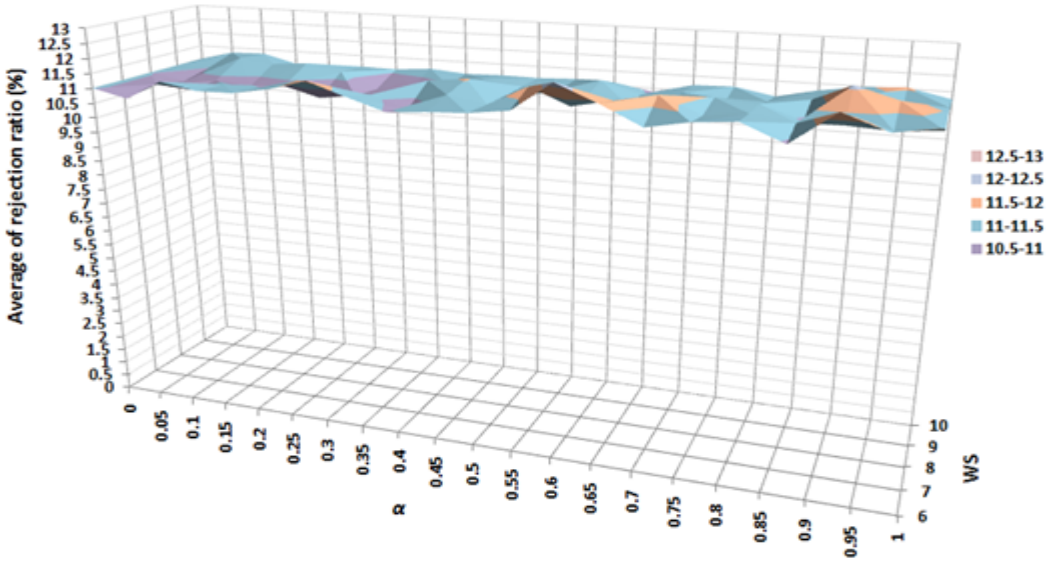

Figure A.10 Average of rejection ratio for the HL scenario (CSPF_PFLRv.1).

A.1.3 Internet2scenario

Figure A.11 shows the average of rejection ratio for requests in the Internet2 scenario using the WSP_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the WSP_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 6 and α equals 0.9.

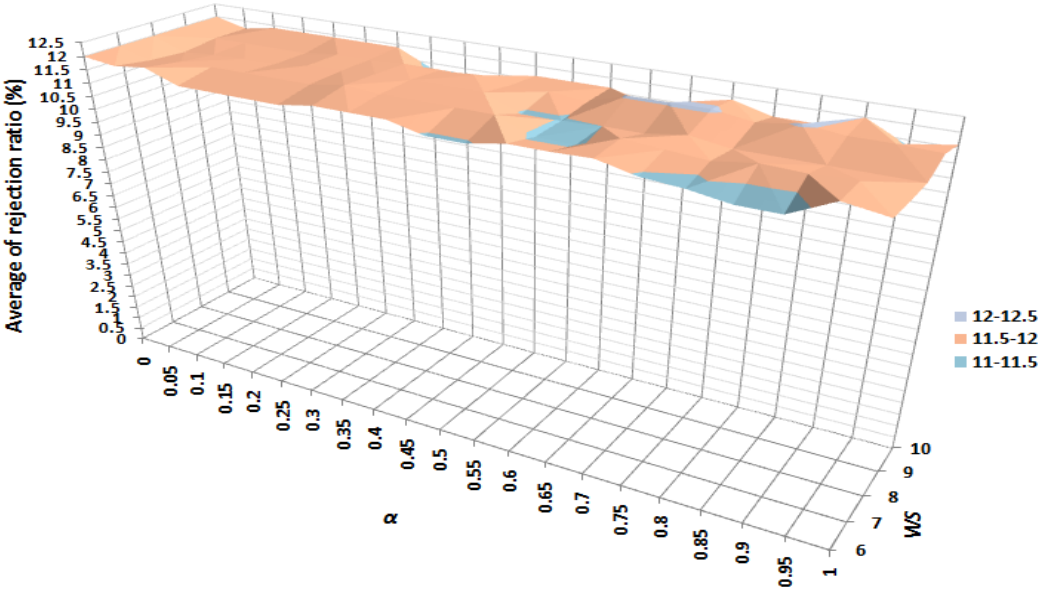

Figure A.11 Average of rejection ratio for the real scenario (WSP_PFLRv.1).

Figure A.12 shows the average of rejection ratio for requests in the Internet2 scenario using the LIOA_PFLRv.1 algorithm with different combination of *WS* and α parameters. The result shows that, the LIOA_PFLRv.1 algorithm has the least average of rejection ratio when *WS* equals 9 and α equals 0.8.

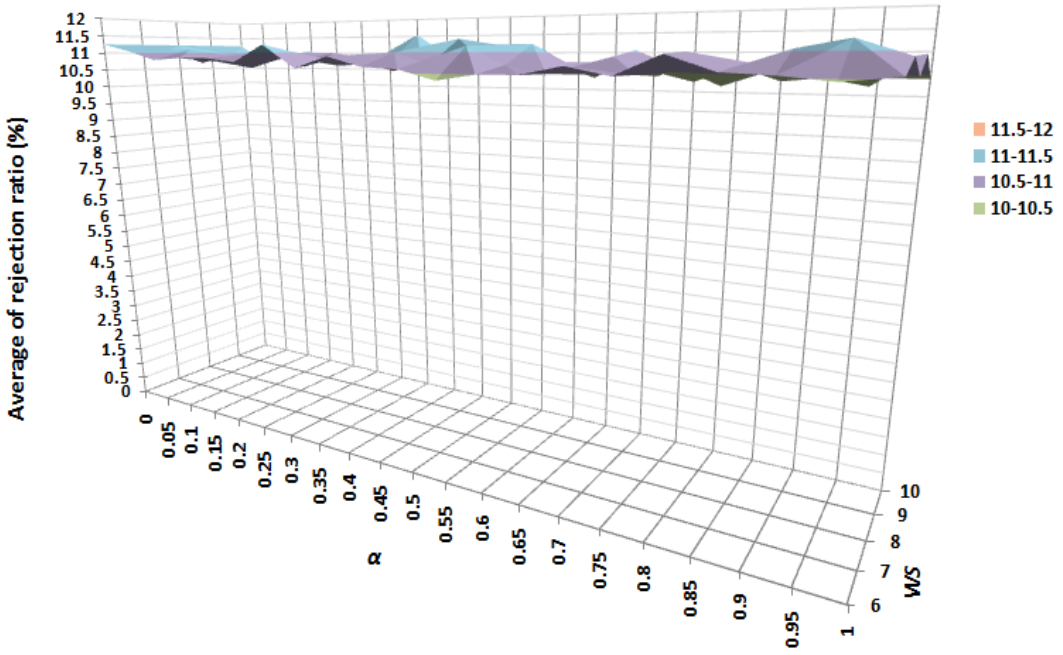

Figure A.12 Average of rejection ratio for the real scenario (LIOA_PFLRv.1).

A.2 The PFLR v.2 algorithm

In the next experiments, analysis studies are performed on two thousands of requests, which are requested after the first four thousands to focus on the steady state, in order to select the best values of *ETh* and α parameters. The aim is to select the best combination of *Eth* and α parameters that have the least average of rejection ratio.

A.2.1 The MIRA topology

A.2.1.1 The ML scenario

Figure A.13 shows the average of rejection ratio for requests in the ML scenario using the WSP_PFLRv.2 algorithm with different combination of *ETh* and α parameters. The result shows that, the WSP_PFLRv.2 algorithm has the least average of rejection ratio when *Eth* equals 51 and α equals 0.25.

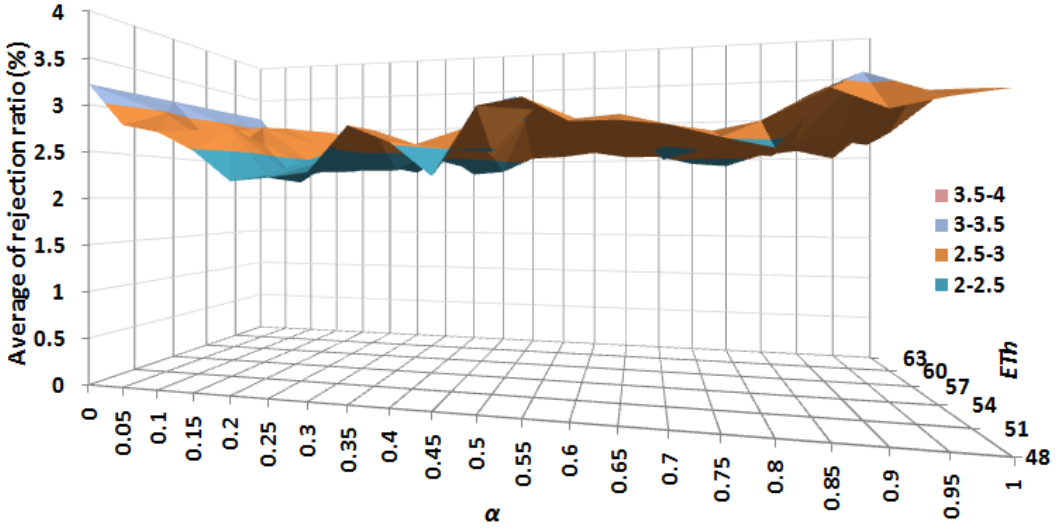

Figure A.13 Average of rejection ratio for the ML scenario (WSP_PFLRv.2).

Figure A.14 shows the average of rejection ratio for requests in the ML scenario using the LIOA_PFLRv.2 algorithm with different combination of *ETh* and *α* parameters. The result shows that, the LIOA_PFLRv.2 algorithm has the least average of rejection ratio when *Eth* equals 60 and *α* equals to 0.2.

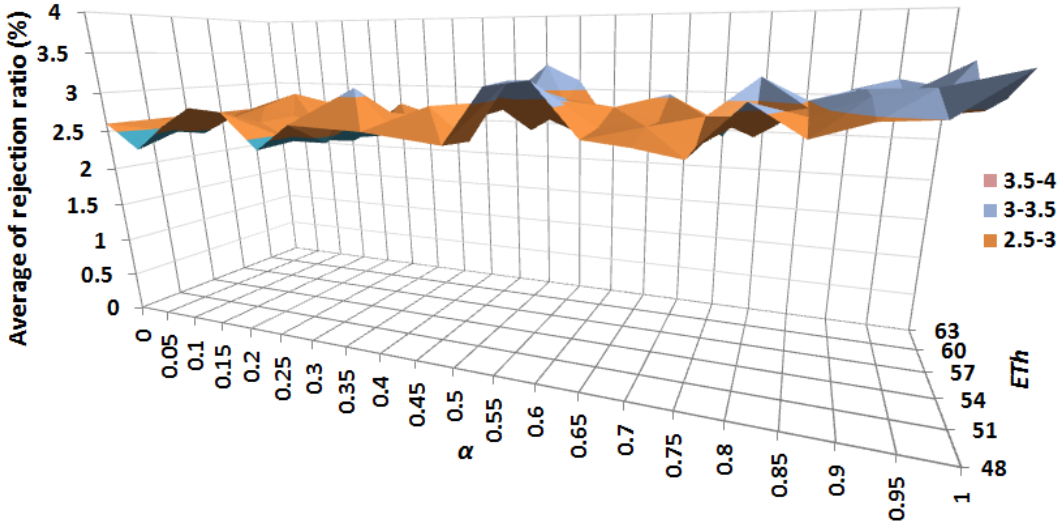

Figure A.14 Average of rejection ratio for the ML scenario (LIOA_PFLRv.2).

A.2.1.2 The HL scenario

Figure A.15 shows the average of rejection ratio for requests in the HL scenario using the WSP_PFLRv.2 algorithm with different combination of *ETh* and *α* parameters. The result shows that, the WSP_PFLRv.2 algorithm has the least average of rejection ratio when *ETh* equals to 51 and *α* equals to 0.05.

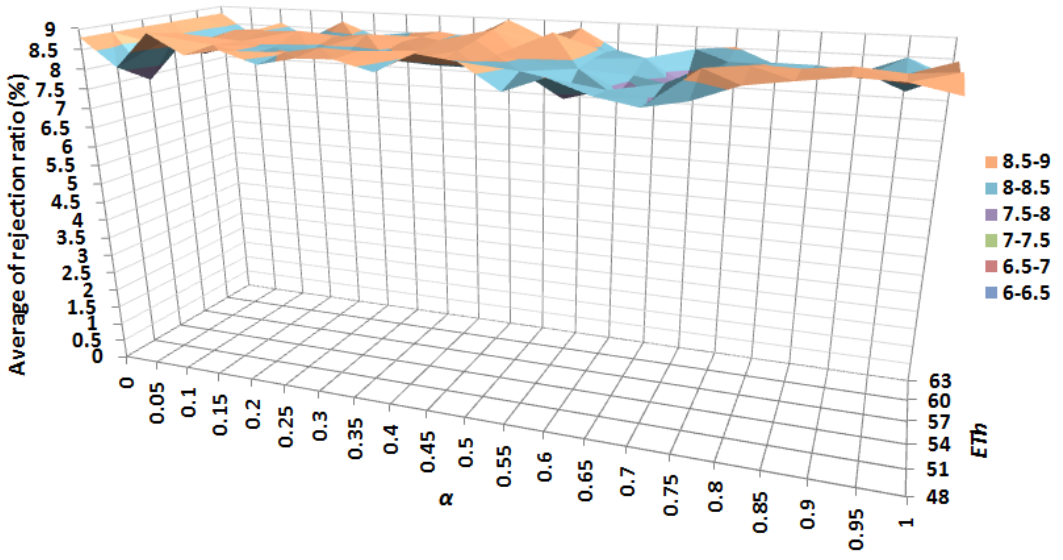

Figure A.15 Average of rejection ratio for the HL scenario (WSP_PFLRv.2).

Figure A.16 shows the average of rejection ratio for requests in the HL scenario using the LIOA_PFLRv.2 algorithm with different combination of *ETh* and α parameters. The result shows that, the LIOA_PFLRv.2 algorithm has the least average of rejection ratio when *Eth* equals 54 and α equals 0.1.

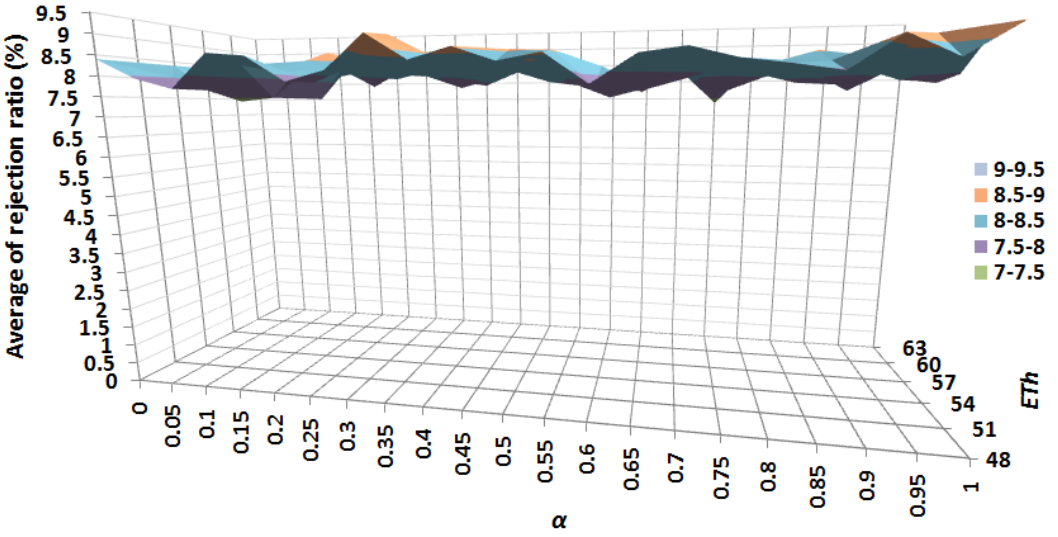

Figure A.16 Average of rejection ratio for the HL scenario (LIOA_PFLRv.2).

A.2.2 The Internet2 scenario

Figure A.17 shows the average of rejection ratio for requests in the Internet2 scenario using the WSP_PFLRv.2 algorithm with different combination of *ETh* and α parameters. The result shows that, the WSP_PFLRv.2 algorithm has the least average of rejection ratio when *Eth* equals 1800 and α equals 0.8.

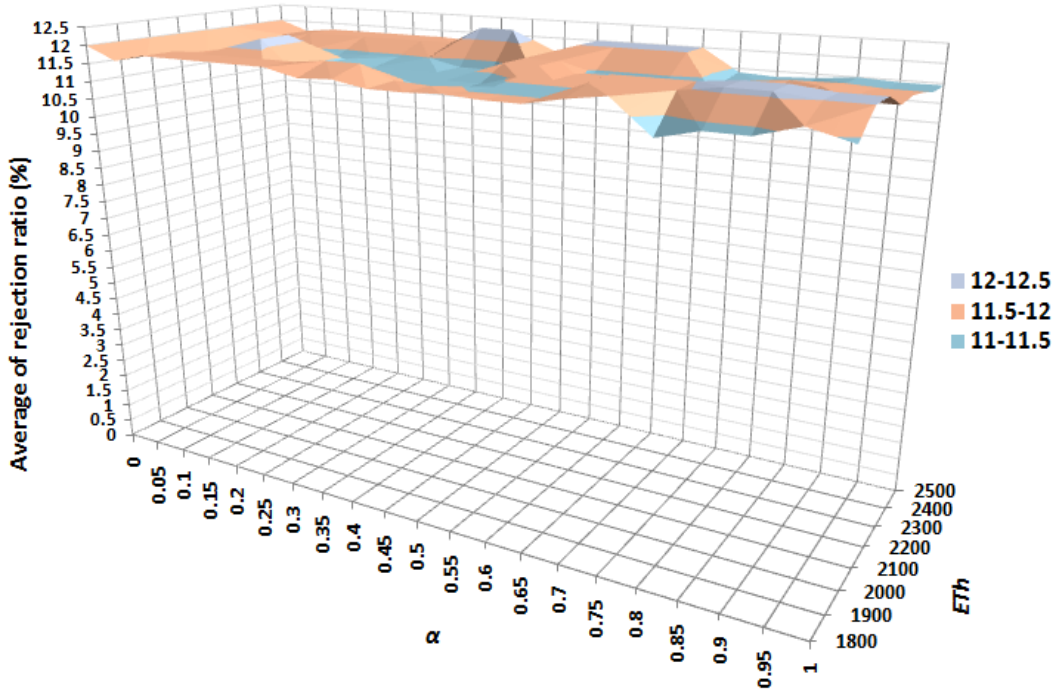

Figure A.17 Average of rejection ratio for the real scenario (WSP_PFLRv.2).

Figure A.18 shows the average of rejection ratio for requests in the Internet2 scenario using the LIOA_PFLRv.2 algorithm with different combination of *ETh* and α parameters. The result shows that, the LIOA_PFLRv.2 algorithm has the least average of rejection ratio when *Eth* equals 2500 and α equals 0.65.

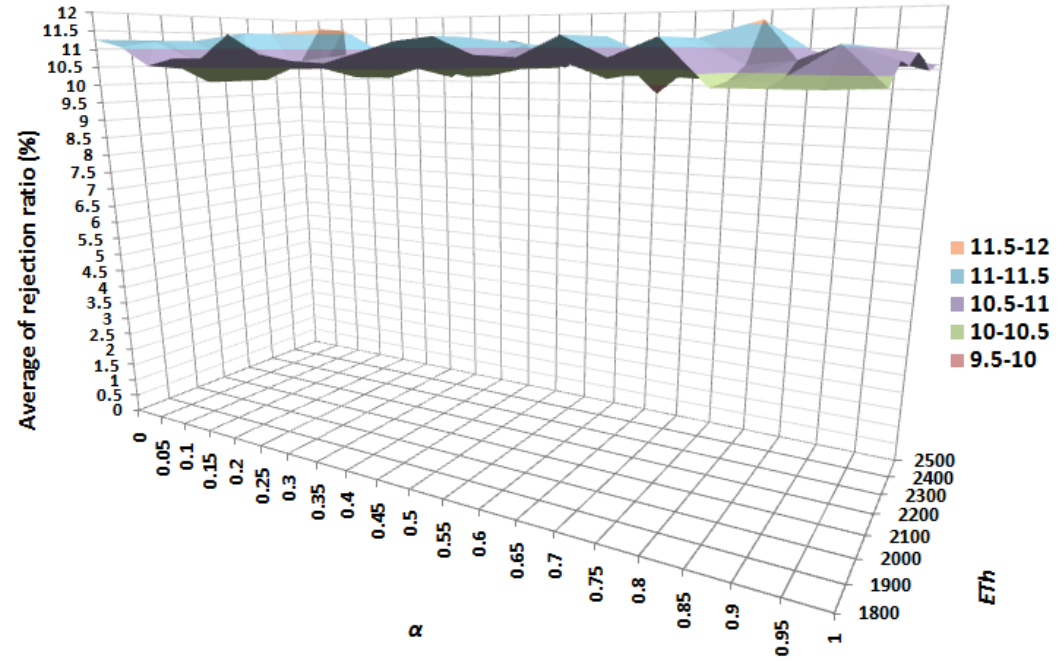

Figure A.18 Average of rejection ratio for the real scenario (LIOA_PFLRv.2).

A.3 The PDR algorithm

In the next experiments, analysis studies are performed on two thousands of requests, which are requested after the first two hundred of time unit to focus on the steady state, in order to select the best values of *ETh* and α parameters. The aim is to select the best combination of *ETh* and α parameters that have the least average of rejection ratio.

A.3.1 The MIRA topology

A.3.1.1 The ML scenario

Figure A.19 shows the average of rejection ratio for requests in the ML scenario using the PDR algorithm with different combination of *ETh* and α parameters. The result shows that, the PDR algorithm has the least average of rejection ratio when *Eth* equals 60 and α equals 0.6.

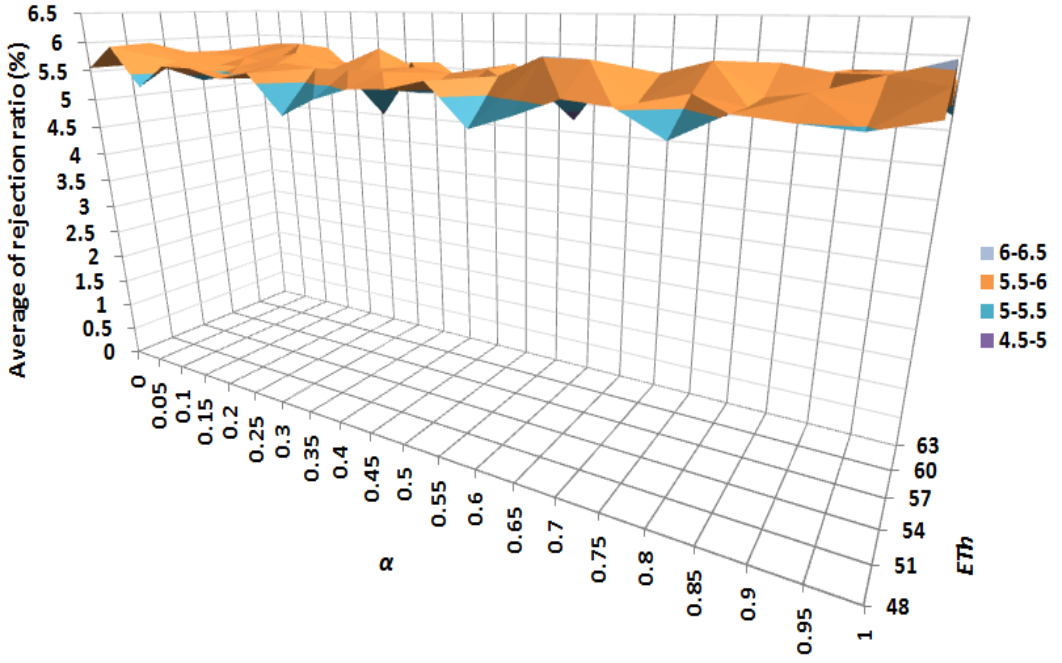

Figure A.19 Average of rejection ratio for the ML scenario (PDR).

A.3.1.2 The HL scenario

Figure A.20 shows the average of rejection ratio for requests in the HL scenario using the PDR algorithm with different combination of *ETh* and α parameters. The result shows that, the PDR algorithm has the least average of rejection ratio when *Eth* equals 54 and α equals 0.5.

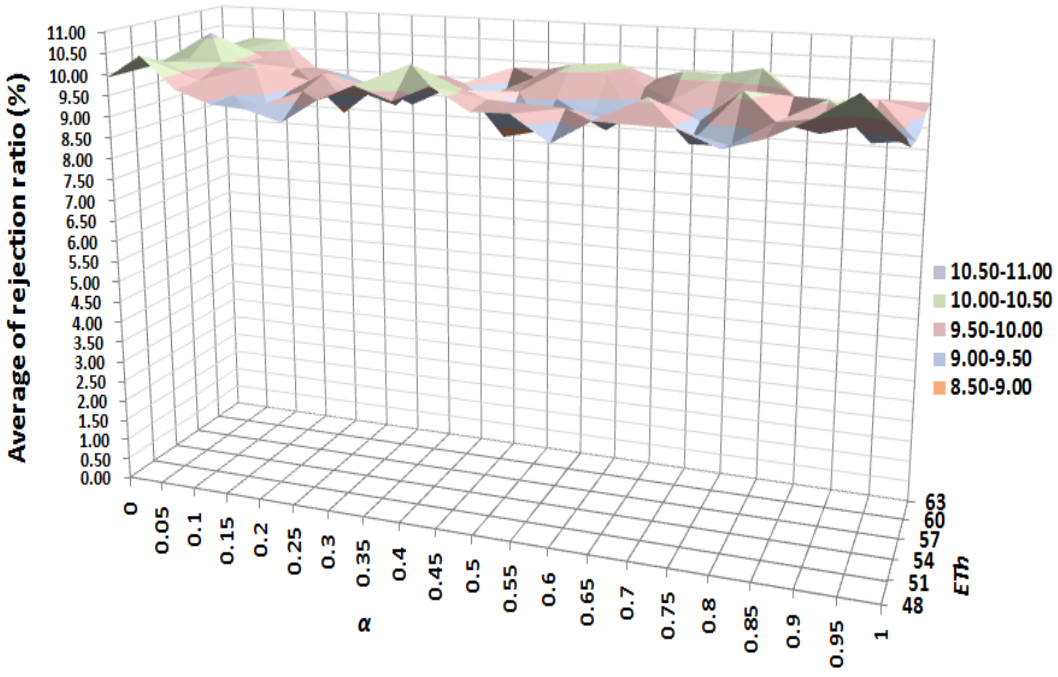

Figure A.20 Average of rejection ratio for the HL scenario (PDR).

A.3.2 The Internet2 scenario

Figure A.21 shows the average of rejection ratio for requests in the Internet2 scenario using the PDR algorithm with different combination of *ETh* and *α* parameters. The result shows that, the PDR algorithm has the least average of rejection ratio when *Eth* equals 1800 and *α* equals 0.25.

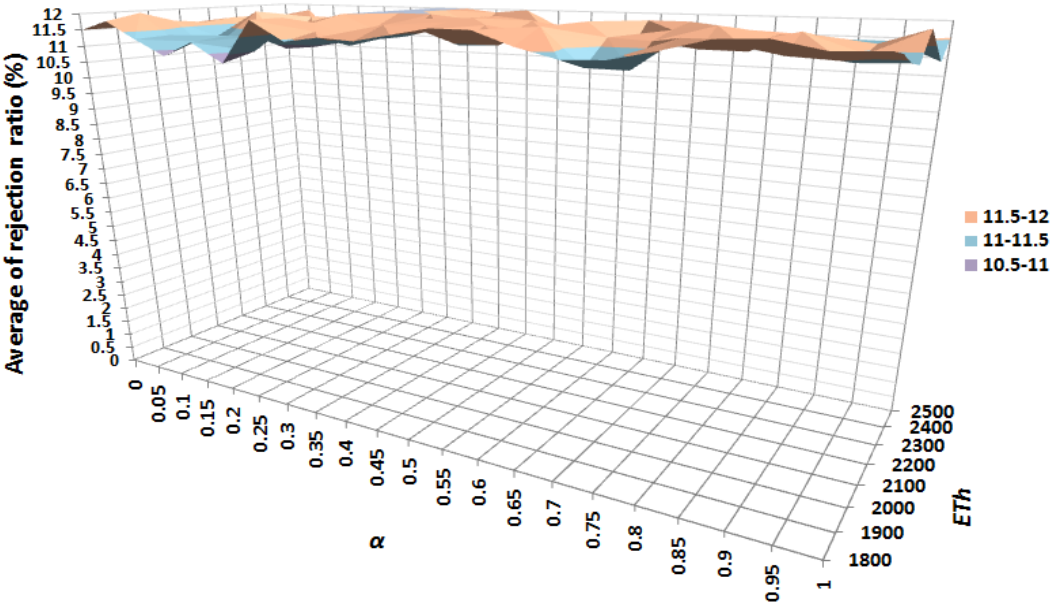

Figure A.21 Average of rejection ratio for the real scenario (PDR)

Appendix B - The prediction accuracy of proposed algorithms

During this appendix, the analysis studies for the PFLRv.2 and PDR algorithms are introduced in order to test the prediction accuracy of proposed algorithms for different load scenarios and with different routing algorithms. Firstly, the focus is on the PFLRv.2 algorithm. After that, the prediction accuracy of the PDR algorithm is presented.

- Network topology:-
  - The experiments are done on MIRA [36] network topology that is often used in evolution of many routing algorithms. The MIRA topology has 15 nodes and 28 links (see Figure 5.1).
  - In the MIRA topology, the thicker links have a capacity of 4800 capacity units while the thinner links have a capacity of 1200 capacity units.
- Generated traffic:-
  - All possible combination of source and destination pairs is considered.
  - The request capacities are randomly distributed among 5-50 capacity units.
  - The arrival of requests follows a Poisson distribution with mean (λ) and the holding time of the requests is based on an exponential distribution with mean ($1/\mu$).
  - Three different network loads are considered. In the first load scenario, called Light Load (LL), (λ/μ) equals (10×47) = 470. In the second, called Moderate Load (ML), (λ/μ) equals (15×35) = 525. In the third, called Heavy Load (HL), (λ/μ) equals (20×29) = 580.
  - In each load scenario, three different traffics are generated with different seeds and with the same (λ/μ) values.
  - Ten thousands of requests are generated. However, to focus on the steady state of network load, the performance of routing algorithms is evaluated after four thousands of requests.
- Performance study:-
  - The prediction error is calculated in every scenario.

B.1 The PFLR v.2 algorithm

Two analysis studies are presented in this section. Firstly, the prediction accuracy of the PFLRv.2 algorithm is presented for different load scenarios. Then, the focus is on the prediction accuracy of PFLRv.2 algorithm for different routing algorithms.

B.1.1 The prediction accuracy for different load scenarios

In the following section, the prediction accuracy of the PFLR algorithm is presented for different load scenarios. Figure B.1 shows the average and variance of prediction errors for different load scenarios with respect to the WSP_PFLRv.2 algorithm. The results show that, the average of prediction error is decreased when the network load is heavier because the changes on the reserved BW within the network links are decreased when the network load is heavier.

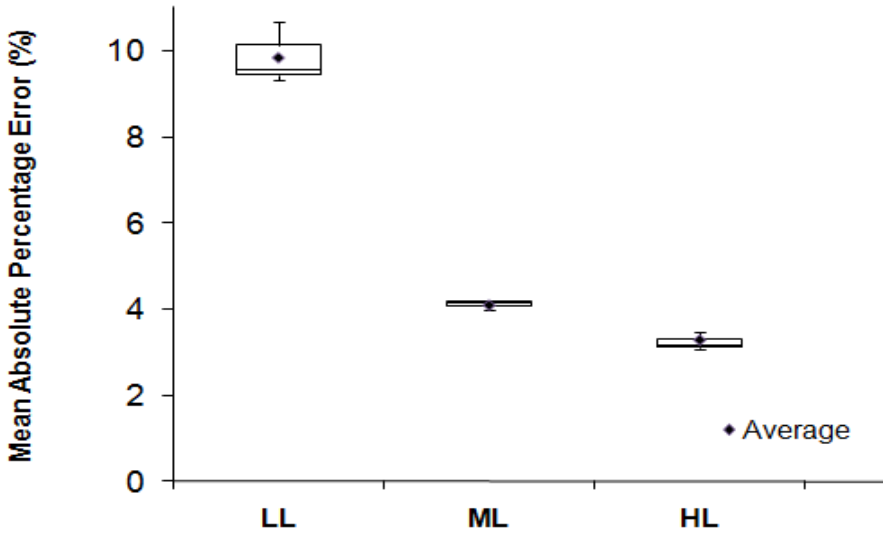

Figure B.1 The prediction error for different load scenarios (PFLRv.2).

Depending on the results in Figure B.1, the average of prediction accuracy for the proposed predictor ranges from 90.15% to 96.70% depending on the network load scenarios, since the Prediction Accuracy (PA) is computed according to the following equation:

$$PA = 1 - Averge\ of\ Mean\ Absolute\ percentage\ Error \qquad \text{Equation (B.1)}$$

The prediction errors and the network load scenarios have this relationship because the PFLRv.2 algorithm is event-based approach. In case of the heavy load scenario, there are no wide changes within the link loads because the links are

loaded most of the time. Therefore, the error prediction is the smallest in this case. In contrast to the heavy load scenario, there are wide changes in the link loads for the light load scenario because the links have a lot of available BW most of the time.

As described in the parameters adaptation process of the PFLRv.2 algorithm, when the prediction error increases above a specific threshold, the training process is triggered. Since the average of prediction error is decreased when the network load is heavier, the number of requests between the sequent training processes is decreased when the load is heavier (see Figure B.2).

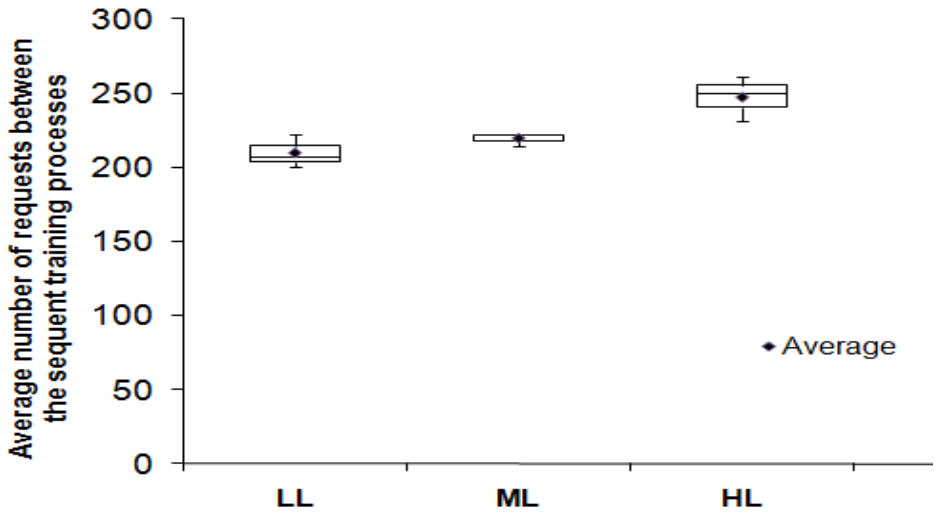

Figure B.2 Average number of requests between the sequent training.

B.1.2 The prediction accuracy for different routing algorithms

Figure B.3 shows the prediction error for different load scenarios with respect to different routing algorithms. The main continuation of this study is to prove the accuracy of the proposed predictor regardless the routing algorithm. This study aims to compare the prediction accuracy of MHA_PFLRv.2 algorithm and other BW-based routing algorithms such as WSP and LIOA algorithms.

The MHA algorithm does not depend on the available BW to compute the routes, but depends on the number of hops between the source and the destination. In other words, the MHA algorithm does not influence the prediction process and the prediction process has not effect on the routing decisions. The results show that, the prediction accuracy of the PFLRv.2 algorithm, in case of the (non-influenced)

MHA routing algorithm, ranges approximately in the same range of (influenced) WSP and LIOA routing algorithms.

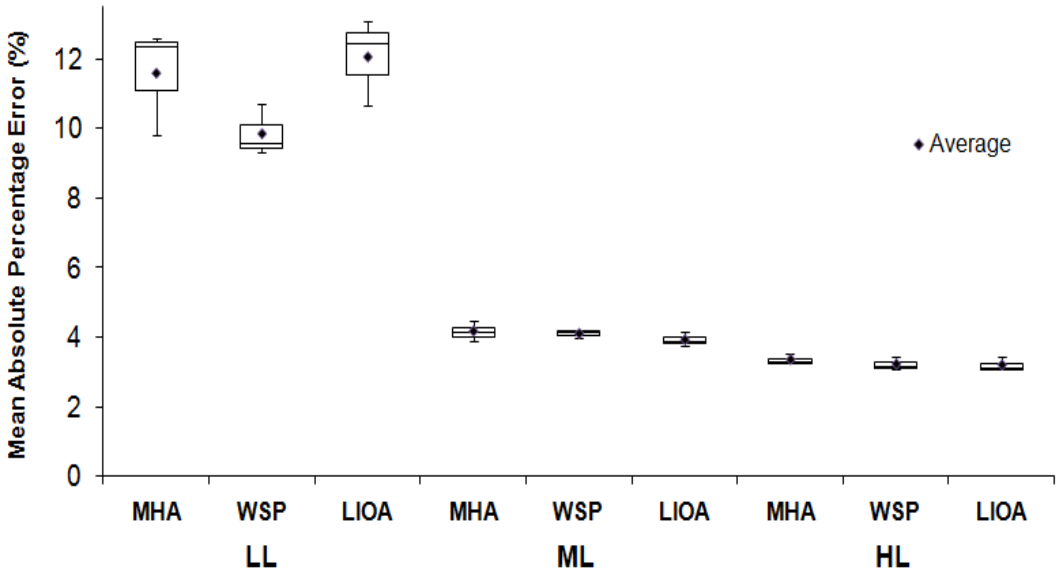

Figure B.3 The prediction error for different routing algorithms (PFLRv.2).

B.2 The PDR algorithm

In the following section, the prediction accuracy of the PDR algorithm is presented for different load scenarios. Figure B.4 shows the average and variance of prediction errors for different load scenarios with respect to the PDR algorithm. The results show that, the average of prediction error is decreased when the network load is heavier. Depending on the results in Figure B.4, the average of prediction accuracy for the proposed predictor ranges from 92.14% to 94.1% depending on the network load scenarios.

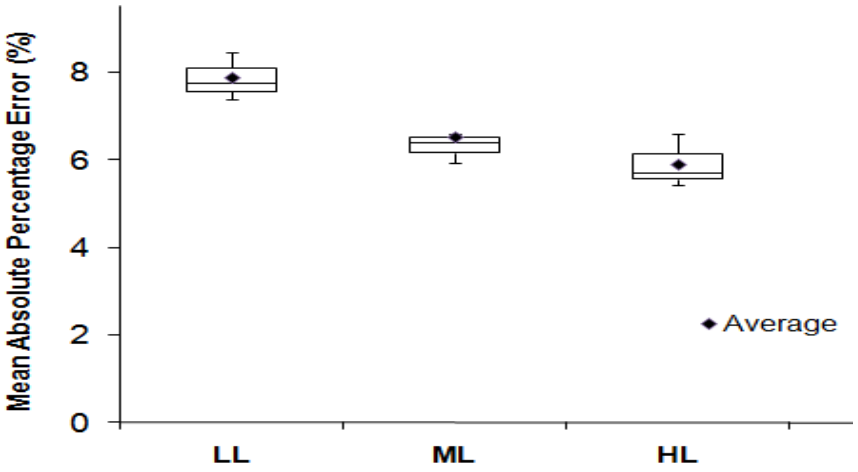

Figure B.4 The prediction error for different load scenarios (PDR).

Appendix C - The stabilization of statistic results

In this section, the experiment results of the ML scenario for ten simulation runs (rather than five) with different seeds within the MIRA topology are presented. The main objective of the following study is to stabilize the statistic result of the PFLRv.1 algorithm. The following experiment is preformed with the simulation details that are described in section 5.2.1.

Figure c.1 shows the rejection ratio of requests for the ML scenario within the MIRA topology. The average of results shows that, the WSP_PFLRv.1algorithm rejects 12.40% less requests than the normal WSP algorithm. Also, the CSPF_PFLRv.1 algorithm rejects 7.04% less requests than the normal CSPF algorithm.

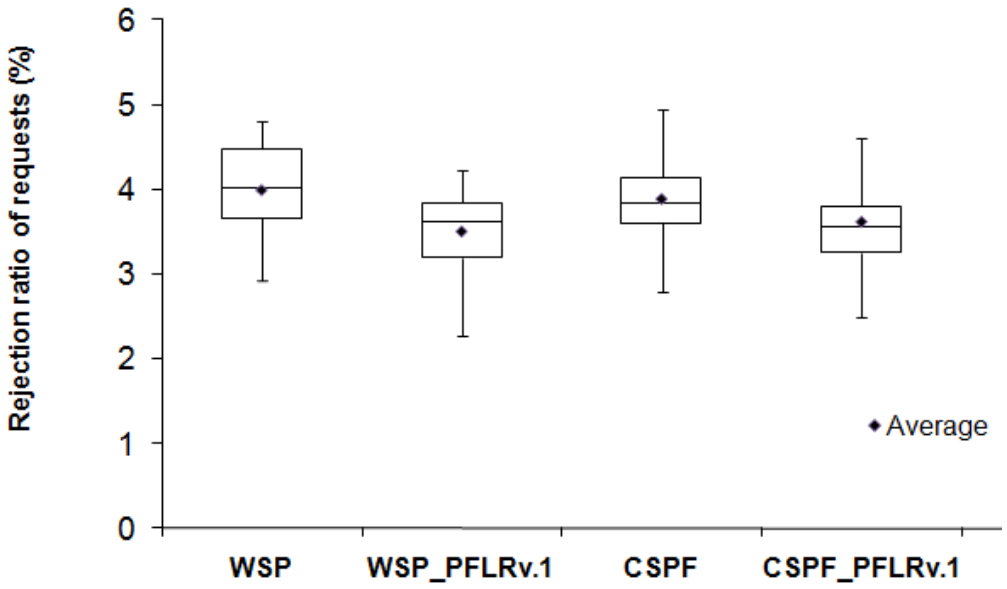

Figure C.1 The rejection ratio of requests for the ML scenario (Ten times).

Figure c.2 shows the bandwidth blocking rate for the ML scenario in the MIRA topology. The average of results shows that, the WSP_PFLRv.1 algorithm rejects 12.60% less bandwidth than the normal WSP algorithm. Also, the CSPF_PFLR algorithm rejects 6.50% less bandwidth than the normal CSPF algorithm.

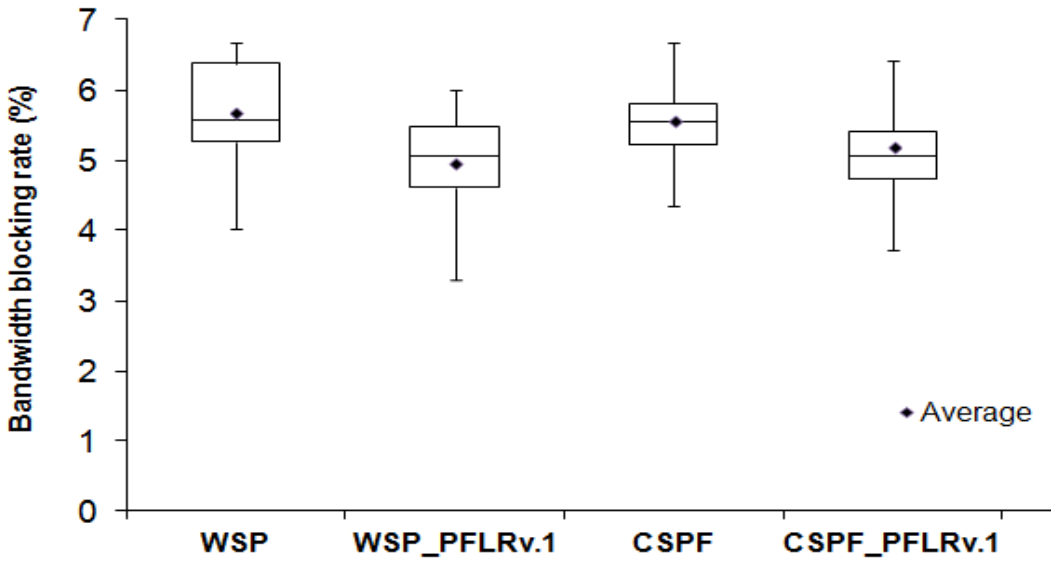

Figure C.2 The bandwidth blocking rate for the ML scenario (Ten times).

The comparison between the results of Figure 5.4 (five simulation runs) and Figure c.1 (ten simulation runs) shows that, both results have the same range and have approximately the close conclusions. Therefore, all the experiments in the chapter 5 are done based on five simulation runs.

Declaration of Independent Work

I hereby declare that the thesis entitled: “Dynamic Routing Optimization Using Traffic Prediction", submitted to Faculty of Computer Science and Electrical Engineering at Rostock University for obtaining the degree of doctor of Engineering, is the result of my own independent and original work.

Also, I hereby declare that the submitted thesis makes use of no other sources or materials other than those referenced, and that quotations and paraphrases obtained from the work of others are indicated as such.

Rostock, 2014 M.Sc. Abutaleb Turky

Theses

1) The efficiency of Traffic Engineering (TE) schemes mainly depends on routing optimization. Additionally, the provided Quality of Service (QoS) depends on the accurate measurement of the available BW.
2) In the current dynamic routing algorithms, the state of network links is represented by specific weights. These weights are used to compute the best paths between the source and destination pairs.
3) Most routing algorithms use the available BW information to represent the link weights. However, due to the varying nature of the available BW, this is not an efficient approach to represent the link utilization.
4) The new research direction is to perform the estimation of the link utilization in the future based on the actual traffic profile and use the estimated values of traffic to enhance the routing performance.
5) In the first contribution, a new efficient routing maintenance approach, called Predicting of Future Load-based Routing (PFLR), is introduced for optimizing the routing performance.
6) PFLR algorithm runs with any routing algorithm whose computations depend on the residual BW. With the use of PFLR algorithm, the future status of the network link loads will be considered.
7) Considering of future network link loads has a big impact in reducing the interference between the requests in the future and so reduces the network congestions and at the same time leads to increase the network utilization.
8) The main idea of PFLR algorithm is combing the predicted link load with the current link load with an effective method in order to optimize the link weights and so enhance the routing performance.
9) The proposed approach uses the Artificial Neural Network (ANN) for building an adaptive traffic predictor in order to predict the future link loads.

10) Additionally, the proposed algorithm has the ability to adapt the parameters of the prediction model, such as the length of prediction step and the prediction validity period, in order to efficiently estimate the link traffics.
11) According to different simulation scenarios, the bundled routing algorithms with PFLR reduces the rejection ratio of requests, minimizes the bandwidth blocking rate and reroutes the requests upon link failure in an optimal way.
12) The second contribution is introducing a new efficient TE algorithm, called Prediction-based Decentralized Routing (PDR) algorithm, which is fully decentralized and self-organized approach.
13) PDR algorithm is a member of ant colony routing class. In PDR algorithm, an ant uses a combination of the link state information and the predicted link load instead of the ant's trip time to determine the amount of pheromone to deposit.
14) Using the link state information helps the routing algorithm to efficiently achieve the BW guarantee of the provided QoS. Additionally, the considering of future of network link loads leads optimizes the routing performance.
15) PDR algorithm uses a similar prediction mechanism to the PFLR algorithm but with local-based implementation. Additionally, the PDR algorithm has the ability to locally adapt the prediction validity period depending on the prediction accuracy in order to efficiently predict the link traffics.
16) PDR algorithm is compared with centralized and decentralized routing algorithms. In general, PDR algorithm performs considerably better than the comparative algorithms with respect to various performance comparison criteria.