UNIVERSITAT POLITECNICA DE CATALUNYA

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION TECHNOLOGIES

CONTRIBUTION OF PROVIDING ABSOLUTE QoS IN OBS NETWORKS

FINAL MASTER PROJECT

Presented to

UNIVERSITAT POLITECNICA DE CATALUNYA

Made by: Br. Mariana Escolar Díaz

Director: Dr. Xavier Hesselbach

January 29th, 2009

Contribution to Proving Absolute QoS in OBS Networks

Escolar Díaz, Mariana

marianae9@hotmail.com

This Final Master Project introduces a new strategy to provide QoS in IP/OBS networks, using routing with prioritization based on statistics, named RPBS. A new method is provided and subsequently validated. This proposal uses the feedback scheme in optical networks to provide statistical knowledge with the objective of finding a suitable route to reach each destination from a specific source node, with more chance of success. This yields a twofold outcome. First, the losses can be reduced in a big number due to statistics. Second, the delays are also reduced compared with other methods based on feedback scheme. These two improvements allow better QoS provision, supporting class differentiation and more efficient resources utilization. The benefits of this proposal are quantified and further compared against existent alternatives by simulations.

Keywords: OBS, Statistical Routing**,** WDM, QoS Provision, JET

Acknowledgements

First, I want to thank my parents, brother and sister, and Alfredo Vaamonde that without their support, patience and constant encouragement I could not have completed this important stage of my life.

In the same line, my eternal gratitude to Dr. Xavier Hesselbach, tutor of this work, and responsible for guiding and advising me during its execution.

I want to recognize the immense knowledge, experience and personal growth achieved during my long hours of working in this project.

Many were those who accompanied me in my academic career, and in my Master studies in Barcelona, specially my colleague and friend Maria Carolina Berrizbeitia, at this final stage of preparing my thesis. Very important people that supported and helped me, and managed to always keep me focus on what truly was useful and beneficial. My eternal gratitude to all of them.

Table of Contents

List of Illustrations

List of Tables

Introduction

I would like to introduce my Final Master Project with the following phrase, which was my inspiration to work in this subject: “Nowadays, the optical fiber is the best physical way for information transport and the light is the best source for its transportation”. [1]

At the beginning of the new millennium several trends can be observed in the field of communication networks. First, bandwidth requirement in networks seems to grow without limits. Internet protocol (IP) based data networks play a central role. This is not only due to the fact that data traffic has surpassed voice traffic but even more due to the exponential growth rate of IP traffic volumes. Second, more and more users and applications request quality of service (QoS) mechanisms from today’s communication networks. Third, optical technology continues to provide an exponential growth in fiber transmission capacities at higher rate than IP traffic growth. In this thesis, I will elaborate on these trends and show how they motivate *Optical Burst Switching* (OBS) as a new switching paradigm for future transport networks. [2]

Recently, OBS represents a balance between circuit and packet switching. It has opened up some exciting new dimensions in optical networking. OBS can provide improvements over wavelength routing in terms of bandwidth efficiency and core scalability via statistical multiplexing of bursts.

In the other hand, in the Optical Packet Switching (OPS) approach, each packet is sent into the network with its own header. This header is going to be earthier electronically or all-optically processed at each intermediate node while the packet is optically buffered. Also OPS may be seen as both the natural choice and conceptually ideal for the future all-optical network. But current optical technology is still immature and not able to overcome its demands. Finally, in order to provide optical switching for next-generation Internet traffic, OBS paradigm was proposed as a flexible yet feasible way.

An OBS core node consists of optical cross connect (OXC), electronic switching control unit and signaling processors. OBS end devices consist of an OBS interface which may be an IP router, ATM switch or Frame Relay switch. These devices are connected to an OBS ingress core node and collect and classify the traffic according to the address of destination end devices, and create the bursts with variable size data units.

A control packet that provides important information about the burst such as burst length and destination address is generated for each burst. This control packet is sent to the network before the burst goes through its path. It is processed at each node in electronic domain and the function of it, is to inform the intermediate nodes on the burst's path about the burst characteristics and to create an end-to-end optical link between the source and the destination.

The transmission of control packet and burst on different channels is one of the most important advantages of OBS. Since the complete burst is transmitted in optical domain, this situation provides high scalability of transmission rate and data format of user data.

The present Final Master Project is structured in five chapters. In the first chapter, the problem statement is formulated and the general and specific aims are introduced, as well the justification, viability and limitations of the investigation. The second chapter summarizes the theoretical fundaments that were used through the investigation, which facilitates the understanding of the OBS topic, and how it's possible to provide absolute QoS, using this technology.

In the third chapter, the methodology followed to achieve the goal initially raised it's exposed and developed, which aims to design a new strategy to provide absolute QoS in OBS/IP networks. It also outlines the activities to fulfill each aim.

In the fourth chapter, the results obtained in the investigation are presented and analyzed. Finally, the fifth chapter includes the conclusions reached during the project development. Additionally, it outlines some recommendations to be

followed by further research, to strengthen the settlement proposal, which are points of interest to this area of research.

Later in this chapter, the references that were consulted are presented. The source code that was used to simulate the model is also appended in Annexes A. The paper that was used for the results comparison, which complements the information throughout the document, is presented in Annexes B.

CHAPTER I

Problem Statement

The trends of existing networks aim towards service integration in scalable network architectures, with robust and high bandwidth. New Generation networks, enabling the establishment of multiple transport of broadband technologies and give it to the user portability in the services provision and applications. With the implementation of these technologies, resources, equipment and trained personnel must be acquired. These are also needed to make the transition from the old to the new system, to test, operate, assess and finally keep it operating.

There is a factor that occurs continually in the telecommunications field. This factor is the competition between equipment suppliers, trying to attract more customers for higher revenue. With constantly evolving and improving technology and infrastructure equipment to offer, supporting innovative applications and ensuring sales to secure economic benefits. As a result, many of the solutions they propose, in terms of infrastructure and transmitting data mechanisms, are not ready for the current market and they're not interoperable with all the other market equipments from different vendors.

This Master Final Project fits into the Telecommunications area and contemplates the design of a new strategy to provide absolute QoS in IP networks which are based on optical switching under quality parameters, in order to be taken as reference for new device creations and technology implementation.

In this present time, optical networks are not fully operational, and a way to get closer to them is using technology based on optical switching, which can be OBS technology, since the perfect jump to these networks, would be implementation of the OPS technology, which is currently very immature and to be able to go into it requires an intermediate study.

Additionally, the current technology equipment is not ready to migrate in an abrupt manner to these famous optical networks were speeds of up to a terabit

per second are expected , being this speed faster than light speed from one point to another. The problem is that with this current technology, the process can't be done at the same proportion, by which the desired speed could never be obtained.

Because of being at the beginning of the first generation of optical networks, were the electrical control plane still plays an important and strong role; it´s why this study it's based in providing QoS in this type of networks, without having to make use of large quantities of electrical domains and giving a step forward to the fully migration of Internet Networks powered optical domains.

I.1. General Aim

Design a strategy that allows the provision of absolute Quality of Service in Internet networks based on OBS.

I.2. Specific Aims

- Study and analyze previews works that are related in methods to provide quality of service in IP/OBS networks.
- Create a new strategy to provide QoS in IP/OBS networks and elaborate the model that will be simulated.
- Study and analyze the simulation software that will be used for the tests made to the model that is going to be presented.
- Create a simulation environment in which the proposed model could be analyzed.
- Compare the proposed model with other studies related on the same subject.
- Compile the results in a paper that could be published at a conference related to the topic.

I.3. Justification

This investigation is warranted in the positive impacts that will be on understanding and obtaining more knowledge in the optical Internet area. It is well known that in this era, were the bandwidth demand is increasing and diminishing fiber availability, network providers are moving towards a crucial milestone in network evolution: the optical network. Optical networks, based on the emergence of the optical layer in transport networks, provide higher capacity and reduced costs for new applications.

Finally, this study aims to help spread a better understanding of the subject, and provides important information to further researches.

I.4. Viability

This Final Master Project includes:

- The establishment and analysis of the current conditions in optical networks and the quality of service provided for these networks. It also clarifies the reason for which the study was based on OBS, and not in OPS.

- The design of a new strategy, which adds improvements in the OBS system, in terms of quality of service, based on previous studies.

- The implementations of tests, that will help demonstrate the adequate performance of this model.

- The Analysis of the results, based on the performance of other mechanisms used to deliver quality of service in optical networks, after having the three models tested under the same conditions.

I.5. Limitations

Much of the needed information during the development of this Special Master Project was hard to find, because it’s a topic that is being investigated recently, therefore many of the related works used to make the research have not been published yet .

The lack of knowledge of the NS-2 simulator, and not having the time to learn how to use it, did not allow the correct testing of the proposal in the same environment that related optical networks works, based on OBS.

CHAPTER II

Literature Review

This chapter presents the main theoretical aspects, which are related to the scope of the thesis, emphasizing concepts that served as the basis of its development. First of all, explaining the investigation background, the optical networks, the WDM (Wavelength Division Multiplexing) and the O/E/O conversion problems. In second place, establishing a framework for research to support the best features of OBS and how it works over IP networks. Finally, describing some related works and the simulation software that will be used for test the model that will be presented.

Within this perspective, the following is the outline of the Literature Review that supports this investigation.

II.1. Optical Networks

Optical networks are telecommunications networks with high capacity based on optical technologies and components that provide routing, grooming, and restoration at the wavelength level as well as wavelength-based services.

Synchronous digital hierarchy (SDH) and synchronous optical network (SONET) refer to a group of fiber-optic transmission rates that can transport digital signals with different capacities. SDH, that was defined by the European Telecommunications Standards Institute (ETSI) for Europe but now used everywhere outside of North America and Japan. SONET was defined by the American National Standards Institution (ANSI) and is used in North America. [3]

SONET and SDH were created with the purpose to obtain optical standards, which allows to standardized line rates, coding schemes, bit-rate hierarchies, and operations and maintenance functionality. Also SONET/SDH defined the types of network elements required, network architectures that could implement, and the functionality that each node must perform.

SONET and SDH often use different terms to describe identical features or functions, sometimes leading to confusion that exaggerates their differences. With a few exceptions, SDH can be thought of as a superset of SONET. The two main differences between the two:

- SONET can use either of two basic units for framing while SDH has one.
- SDH has additional mapping options which are not available in SONET.

An automatically-switched optical network (ASON) is a network which is based on technology enabling the automatic delivery of transport services. Specifically, an ASON can deliver not only leased-line connections but also other transport services such as soft-permanent and switched optical connections. ASON is a framework for protection switching. The reutilization can be articulated by GMPL or some implementation which adders the ASON standard.

Optical networks began with WDM which arose to provide additional capacity on existing fibers; this technology provided many virtual fibers on a single physical fiber. Like SONET/SDH, defined network elements and architectures provide the basis of the optical network. However, unlike these two standards, rather than using a defined bit-rate and frame structure as its basic building block, the optical network will be based on wavelengths. The components of the optical network will be defined according to how the wavelengths are transmitted, groomed, or implemented in the network. [4]

WDM systems are divided in different wavelength patterns, *conventional* or *coarse* and *dense* WDM. Conventional WDM systems provide up to 16 channels in the 3rd transmission window of silica fibers around 1550 nm. Dense WDM (DWDM) uses the same transmission window but with denser channel spacing. Channel plans vary, but a typical system would use 40 channels at 100 GHz spacing or 80 channels with 50 GHz spacing. Some technologies are capable of 25 GHz spacing (sometimes called ultra dense WDM).

At this time, the WDM technology works in a point to point architecture over big distances, in networks with SONET/SDH interface to higher layer, this

kind of scheme needs conversions between the optical and electrical domains, this means O/E/O conversions in each node, which don't let to use in an efficient way all the bandwidth. The signal is converted to the electrical domain, were the processing is doing, being transformed to the optical domain again before to be introduced in the exit fiber. With this is possible to say that the technology efficiency is given by the electronic devices capacity in the system. Also these devices have problems against the big sizes, the power consumption and the overheating, also the price of the O/E/O convertors.

Knowing the previous problem about the O/E/O conversions, this study is focused on finding a model that eliminates in some way these conversions and all the data that travels at the optical domain in "All Optical Networks" (AON), which will allow obtaining faster transmission rates. [5]

Optical technology plays an important role in transmission, but in the next generation this technology will play a stronger role in optical Internet. The studies in this topic are focused on three optical switching technologies: Optical circuit switching (OCS), optical packet switching (OPS), and finally optical burst switching (OBS).

Comparing these three optical switching technologies is possible to see that OBS is the most useful on this era, rather than OPS which is the most efficient, this is because:

- OBS provides efficiency and scalability by statically multiplexing of burst
- The setup latency of OBS for a burst payload is lower than OCS.
- OBS is easier to implement than OPS because it's less strict in requirements, in processing control signaling and achieving synchronization.
- Amortized overhead in routing and forwarding of an OBS router is smaller than that of on OPS router.

Finally it's possible to say that OBS is more attractive switching paradigm because has less stringent requirements for optical devices than OPS and is more

efficient than OCS in terms of wavelength utilization efficiency, that's why I choose for this study OBS technology. [6][7]

II.2. OPS Basic Concepts

A section is dedicated to explain OPS, because it's a long-term strategy to provide high-speed transmission up to Terahertz in IP networks, data transparency, and reconfigurability.

This technology is based on packets transmission and will allow the following benefits:

- Dynamic bandwidth allocation.
- Links will be occupied on demand.
- Alternative routes when congestion occurred.
- Packets from different sources can coexist on the same customer-to-network physical link without interference.
- Allows terminals operating at different bit rates to internet-work with each other.
- Routers will buffer the packets from a higher bit rate hosts.

But, as I had been explaining through this section, now a days, the optical networks are not ready for this technology and OBS can be a step forward to "All Optical Networks", so that's the reason why this Final Master Project is focused on OBS over OPS, even though OPS can provide more benefits than the chosen technology. [8]

II.3. OBS Basic Concepts

OBS is a switching burst technology between OCS and OPS, as can be seen in Table 1. The data is transported on variable sizes information unities, known as burst. If the size of the burst is too big, this technology will be similar to OCS and is the burst is smaller this will look more like OPS.

Technology	Granularity	Utilization	Complexity
OCS	Coarse	Low	Low
OPS	Fine	High	High, immature
OBS	Moderate	Moderate	Moderate

Table 1 Comparison of the switching technologies using WDM networks

Each sent data burst is preceded by a control packet that contains the routing information that the burst must follow in the network. After the control packet is sent for a dedicate channel and without confirmation, the ingress node sends the data burst, following the path done for the resource reservation in a unique direction.

In the OBS technology the control plane and the data plane are strongly separated, allowing the optical data switching, using only the optical domain. Meanwhile the resources accommodation is processed in the electrical domain. This is possible by reserving an optical channel for the control information or having a parallel electrical network for this purpose. The separation of both planes allows good flexibility, network management, scalability and adaptability. [9]

II.3.1. Network Architecture

OBS is a network formed by edge nodes and core nodes, which are interconnected by WDM links. The edge nodes have an electrical and an optical part, and these are the access interface between electronic and optical networks.

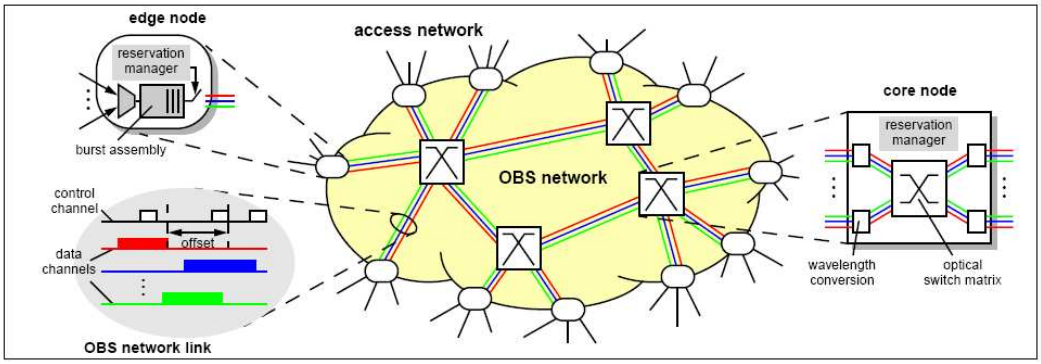

Figure 1 Node and network architecture for optical burst switching (extracted from [2])

The architecture of an edge node is possible to see in Figure 2, and these are in charge of:

- Performing bursts assemblies and disassemblies at the boundary of OBS networks, to support various network topologies.
- Initiating the connection establishment.
- Establishing the offset time between the control and data packets.
- Queuing packets from users.

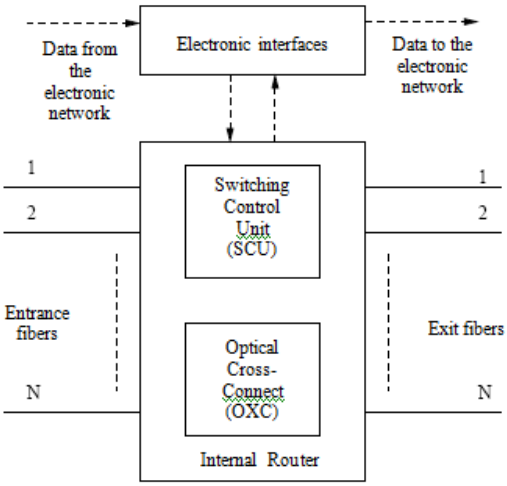

Figure 2 Architecture of an OBS edge node (extracted from [5])

Meanwhile the core nodes only have the optical part and these are in charge of signal propagation, burst switching and resources management.

OBS uses a burst which consists of a burst header, which is a burst control packet (BCP) and a burst payload or burst data (BD). BCP and BD are transmitted separately on different wavelengths with the BCP ahead in time and switched in electronic and optical domains at the OBS core routers they traverse, subsequently. [10]

II.3.2. Burst Assembly

To do the burst assembly process in the edge router, input packets are added and assembled into a burst in a logical way, as it can be seen in

Figure 3. This is important if it's wanted for considering Class of Services (CoS) based on the DiffServ technique.

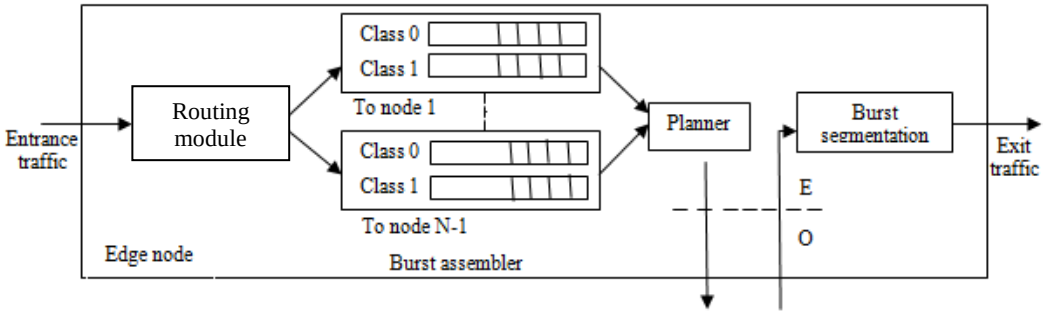

Figure 3 Electronic interface of an edge router (extracted from [5])

The received packets are collected and sorted by upper layers, based on their destination addresses and class of service. Then these are added until they form different bursts, as its shown in Figure 4. An edge node must have storage buffers considered as class of services and possible destination edge nodes.

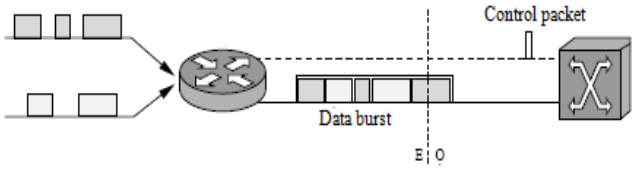

Figure 4 Burst conformation mechanism (extracted from [5])

The most common assembly techniques are timer-based and threshold-based. In the timer-based approach, a burst is created and sent into the optical network at periodic time intervals; therefore, the network may have variable length input bursts. A threshold is a limit on the number of packets contained in each burst before the burst is sent, thus, the network will have fixed-size input bursts. [11]

There is also a hybrid technique, were both techniques, time and size, are taking into account. The burst will be constituted when one of the two parameters exceeds its limits.

II.3.3. Connection Establishment

The optical bursts switching is an adaptation of an ITU-T standard (International Telecommunication Union Telecommunication Sector Standardization) about burst switching at Asynchronous Transfer Mode (ATM), known as ATM block transfer (ABT). The reserve resources protocols for the connection establishment from the two ABT versions are: Delayed Transmission (ABT/DT) and Immediate Transmission (ABT/IT).

II.3.3.1. ABT/DT

When a source has a burst to pass, the first attempt is to reserve bandwidth by sending a small request message. Each intermediate node that receives the request message will accommodate the necessary resources. If the route can be established, the destination will send a confirmation message to the source, informing that the burst can be send, and if not, a node that cannot accommodate the necessary resources will send a dissent packet, and the source will retransmitted the request message after certain time.

II.3.3.2. ABT/IT

The source transmits the burst without making resources reservation. At each node the burst will be delayed while commuting routes. If the resources reservation fails at any intermediate node, a dissent report is sent to the source, which initiates the burst retransmission after certain time. This is the adopted concept in optical networks and in order to implement it, some schemes are needed o complemented in the processing time. These are explained below. Also in Table 2 it’s possible to see the comparison between them.

II.3.3.2.1. Tell-and-Go (TAG)

In this scheme the source transmits the SETUP message and immediately after it, it transmits the optical burst. With this scheme, is possible to compensate the control header processing time and prevent the burst of going to the optical

switch before it has been set, introducing a fixed delay through FDLs at the entry ports.

II.3.3.2.1. Tell-and-Wait (TAW)

This scheme is the opposite of TAG. In this case, the SETUP message is propagated all the way to the receiving end-device, and each OBS node along the path processes the SETUP message and allocates resources within its switch fabric. A positive acknowledgment is returned to the transmitting end-device, upon receipt of which the end-device transmits its bursts. The burst will go through without been dropped at any OBS node.

The offset can be seen as being equal to the round trip propagation delay plus the sum of the processing delays of the SETUP message at each OBS node along the path. TAW suffers from high end-to-end packet delay.

II.3.3.2.2. Just-in-Time (JIT)

There is a delay called offset between the transmission of the control packet and the transmission of the burst. This scheme uses immediate configuration in which the OBS node allocates resources to the incoming burst immediately after it processes the SETUP message and is simpler to implement.

II.3.3.2.3. Just-Enough-time (JET)

In the JET scheme there is a delay between transmission of the control packet and transmission of the optical burst. This delay can be set to be larger than the total processing time of the control packet along the path. This way, when the burst arrives at each intermediate node, the control packet has been processed and a channel on the output port has been allocated. Therefore, there is no need to buffer the burst at the node.

This is a very important feature of the JET scheme, since optical buffers are difficult to implement. A further improvement of the JET scheme can be obtained by reserving resources at the optical burst switch from the time the burst

arrives at the switch, rather than from the time its control packet is processed at the switch. JET suffers from high packet loss. [12][13]

Mechanism	Use resource	End-to-End Delay	Resource reserve	DiffServ
TAG	Efficient	Short	Guaranteed	Not Supported
TAW	Deficient	Long	Guaranteed	Not Supported
JIT	Efficient	Medium	Guaranteed	Supported
JET	Efficient	Short	No Guaranteed	Supported

Table 2 Comparison between signaling protocols schemes (extracted from [14])

Finally I got to the conclusion that JET is the scheme with better performance for QoS provision. For that reason is why all the related works, in which this work was based, are related to this routing scheme.

II.3.4. Offset Time

Offset time is the time interval between the beginning of the control packet transmission/reception and the beginning of the associated burst transmission/reception.

The adequate choice of the offset time is a key point in the design of an OBS network, since it has a huge impact on the final performance, measured in terms of delay and data loss. The offset time can be calculated with a random or fixed value.

Sometimes in order to calculate the fixed offset time in distributed or centralized schemes, all the core nodes processing time plus the optical matrix configuration at the egress node, are needed. Also it's important to know the exact number of jumps between the source and destination nodes. This information is processed by the edge node. For the centralized schemes, the needed times, is the time the user needs to make the petition to the central planner and the time to compute the route and the signaling time. Meanwhile for the random offset time calculation, an algorithm based on the burst conformation time is used. [5]

II.3.5. Routing

The routing can be done in different ways. The most common are the hop-by-hop routing, routing through explicit calculated routes and Multiprotocol Label Switching (MPLS).

The hop-by-hop routing is based in a simple algorithm that consults routing table that stores the routes to particular network destinations, as in IP networks.

The routing through explicit calculated routes consists in establishing explicit routes based on protocols as Resource Reservation Protocol with Traffic Engineering (RSVP-TE) or Constraint-Based Label Distribution Protocol (CR-LDP). This routing type guarantees CoS, because it is possible to obtain some metrics as delay, jump numbers, blocking probability or bandwidth; but at the same time it has a downside which is the slow recovery against flaws.

MPLS semi permanents use routing decision labels in which a flow is designated for each data transmission. The possible routing options are divided into classes. The packet label is done in the ingress node according to the CoS and redirected to its respective egress node. Each node will consult the label of the entrance packet and will change for a new exit label. This protocol allows reducing the processing time of the control information, permitting better scalability and performance. Also MPLS allows building multipoint to point trees and establishing explicit routes without the inefficiency to transport the route in each packet.

II.3.6. Internal Commutation

As it has been said before, in OBS is used the kind of routing in only one direction without confirmation. These imply that the ingress nodes send burst without having reserve confirmation. This means that for a given node, bursts come from different input channels competing for the same output channel. This is a big problem for OBS because the use of storage is impossible or very limited, so it has to deal with it differently as in electrical networks. Some methods using

wavelength convertors, Fiber Delay Line (FDL) and methods of discard deflect and segmentation can be used.

II.3.6.1. Exploiting the Wavelength Domain

If two bursts compete for the same output channel, one of the signals can be converted to an optically wavelength of entry to another one of exit. The wavelengths conversion reduces the losses in an OBS network. This conversion can be total or partial. In the first case there is a converter for every possible wavelength, and in the second case the number of converters is less than the total number of wavelengths. Another advantage that wavelength converters offer is the ability to improve equity among routes with different number of jumps.

II.3.6.2. Optical Buffering

There is a limited way to store signals through optical FDL, which can increase the utilization and reduce the data loss. This is a mature technology and conceptually simple, but the devices are too bulky, adding additional delay.

II.3.6.3. Methods of discard, deflection and segmentation

The simplest way of resolving disputes is the discarding of bursts, if at the burst arrival to all output channels are busy this is discarded altogether. Another option for schemes with quality of service is to plan the incoming burst and discard a burst previously planned, according to its priorities. Before discarding some burst it’s possible to consider other mechanisms as:

II.3.6.3.1. Deflection routing

Which consists in sending the incoming burst for an exit port if there is no alternative channel available in the desired port. This mechanism can significantly reduce the losses, but it should be noted that if the burst was detour through a route with the highest number of jumps, congestion on the network can occur in terms of high loads of traffic. In addition, it should be noted the possibility that

the bursts are diverted to longer routes in the number of jumps from the ingress node, and set to a bigger offset time, which on the other hand increases the delay.

II.3.6.3.2. Segmentation

This technique consists, that bursts can be planned completely or partially by the division in the same segments. This technique may have a complication, but it achieves very good results, especially if it's combined with deflection. This mechanism reduces the data loss. [15]

II.3.7. Algorithms for channel planning

An entrant burst can be planned in more than a wavelength of the output port. That's why channel planner are needed because it takes into account the plans made on each channel to select an output channel. Some common algorithms are First Fit Unscheduled Channel (FFUC), Latest Available Unscheduled Channel (LAUC) and LAUC with Void Filling (LAUC-VF), which are described below.

II.3.7.1. FFUC

This algorithm keeps information about the last moment planned in each channel. Each time a burst arrives; it'll be looking in a certain order among the various channels and selecting the one that has a last planning moment smaller than the arrival time of the burst. Its main advantage is its computational simplicity, and its main drawback is the low usage. [16]

II.3.7.2. LAUC

The main idea is to increase utilization by minimizing existing gaps. With LAUC, the selected output channel is the one with the nearest planning time to the burst arrival time. [16]

II.3.7.3. LAUC-VF

It is similar to LAUC except it allows the gaps to fill with new bursts. The gaps between bursts are the spare capacity of the channel, and its effective use by the LAUC-VF has better benefits in terms of data loss than FFUC or LAUC. On the other hand, this algorithm is more complex because it needs more state information. [16]

II.4. OBS over IP networks.

In order to implement OBS over IP networks, it's important to remember that in Internet QoS is needed to support all kind of applications. The most important and hard to perform are the real time applications, were big delays can´t be allowed.

OBS is able to support multicast applications, which is an important aspect in Internet applications because these can be deployed in a more efficient way. In OBS, multicast is achieved through light bifurcation, which carries inherently signal lost. Therefore there is a limit on the number of times that a signal can be bifurcated.

II.4.1. QoS supported based on OBS

In Internet QoS is needed in order to assure the well performance of the different kind of applications, like the real time application that cannot tolerate big delays and in this way these should have higher priorities. Alternative solutions exist as class of services based on the offset time: CoS based on resources expropriation and CoS based on priority segmentation. Each concept is explained bellow.

II.4.1.1. CoS based on offset time

There is a study that uses this theory to obtain CoS using the offset time. They describe a provision scheme for QoS based on offset time, which achieves almost 100% isolation between different traffic classes, quantifying the extra time

needed for this purpose and for the measuring performance. According to the authors, the probability of data loss decreases when the interval between burst and control packet increase. The disadvantage of this system is the additional delay, which may be unacceptable in certain applications. [17]

II.4.1.2. CoS based on resource expropriation

New bursts with higher priority can abort made schedules. The additional delay for priority classes don't exist, but this is a technique that adds a lot of planning complexity, to overcome a number of drawbacks such as sophisticated signaling to allow the abort scheduling and with these to obtain a good channels use. [16]

II.4.1.3. CoS based on priority segmentation

While previous solutions provide differentiation at burst level, segmentation can be done at the packet's level. Thus, packets of different classes of service are added at different bursts, and when there is contention, the low priority burst will be segmented and will suffer greater data loss. [15]

II.4.2. Classes of Service Needed on Internet

In the Internet community and in resent research, it's indicated that only two classes are needed, the stream and elastic, in order to support QoS. [18] That's why previous studies based in providing QoS based on OBS, emphasizes in the analysis of loss burst probability, were high probability packets are affected by the offset time.

This work proposes a method of deciding the offset time for supporting the required QoS in optical Internet based generalized MPLS (GMPLS) over OBS. They consider that IP traffic is classified in two classes, the class 1 as high priority class, which has a priority offset time as the extra portion of the offset time, and the other class will be the class 2 which is the low priority class, without an extra offset time.

Once they have this classification, they use the Offset Time Decision Algorithm (OTD), which is able to decide on the appropriate priority offset time according to the required QoS, by using the reserved equation of the heuristic loss formula. For each requested burst lost rate (BLR), QoS can be guaranteed under some range of offered load. Their final conclusion about this study is that when the offered load is low enough to support the QoS without any offset time, the requests BLR is lower than the performance of classless, but when the traffic load changes, the OTD algorithm is able to fine the offset time needed to allow the existence of the two class of service for that QoS in the system. [11]

II.4.3. LOBS

OBS leads to an integrated IP/WDM solution that can further reduce redundancy and increase efficiency in an IP/WDM network. So it's possible to say that OBS facilitates the extension of the MPLS framework. This can be accomplished by increasing each OBS node with an IP/MPLS controller, although making it a labeled OBD (LOBS) node, which is similar to a label switched router (LSR).

Then, each control packet can be sent as an IP packet containing a label as a part of its control information along a pre-established LOBS path, similar to a label switched path (LSP). In addition, each data burst can be formed by assembling several IP packets from a common ingress LOBS node to a common egress LOBS node, and sent (on a separate data wavelength) along the same physical route as that traversed by the LOBS path used by the corresponding (labeled) control packet. In such a LOBS network, both explicit routing (ER) and constraint-based routing (CBR) developed within the MPLS or GMPLS framework can be extended to provide and engineer LOBS network resources. [19]

II.4.4. TCP over OBS

Currently TCP is the protocol most used on the Internet, and being OBS a strong candidate to support future IP backbone networks, it deserves some attention for the interaction between TCP and OBS. Recent studies show how

some OBS features affect the performance of the TCP protocol, especially in terms of its mechanism for controlling congestion.

In the Detti and Listanti work named “Impact of segments aggregation on TCP Reno flows in optical burst switching networks” these effects were studied and they got to the conclusion that the procedure for assembling packets in bursts, reduces the throughput due to increased delay (increase of the RTT (Round Trip Time) parameter of TCP. For this purpose it is called lost by delay, and it is remarkable for times of high assembling. On the other hand, the same assembling process offers benefits since it increases the amount of data sent between two losses. This effect is known as a gain correlation and it’s noticeable when the bandwidth of access to TCP source is medium / high.

A source which has low bandwidth in its IP network access and an assembly burst time smaller into the edge node, will gain little correlation and little lost by delay. The throughput in this case will be similar to that obtained in the scheme without burst assembler.

For a source with a relatively high access bandwidth and a high assembly burst time, all segments of the same TCP transmission window can be assembled in the same burst. In this case, it maximizes the gain by correlation, but it is also notable loss by delay.

The use of burst adaptive assembly algorithms adaptive get better throughput since the assembly time can be dynamically adjusted to the TCP congestion control mechanism. [20]

Optical Internet with JET-based OBS cannot achieve reasonable throughput in TCP layers because the size of the contention window for each TCP sessions will be decreased even though there is only one burst loss in the OBS network. [21]

II.5. Related Works

In this section some recent related works that were used in the investigation, are mentioned with the intention to show how this area is been studied during this era. They are ordered by publication date.

II.5.1. QoS Scheme in JET OBS Systems

The class differentiation in OBS networks, based on JET scheme can be implemented by assigning an extra offset time to high-class bursts. [22] To illustrate this scheme, it's considered a simple two-class OBS network. In Figure 5 "α_h" a high-class data burst and a "α_l" low-class data burst arrives at node *i* at nearly the same time and desires to be switched to the same outgoing channel. Since the high-class burst has a larger offset time, its control packet arrives first and successfully reserves the channel. When the control packet of low-class arrives, it sees that the channel has already been reserved. In this way, the high-class burst has implicit priority over the low-class burst.

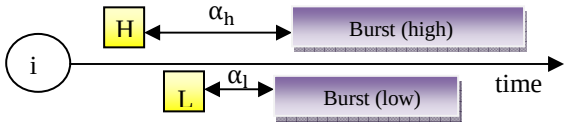

Figure 5. Illustration of how the high-class data burst successfully reserves the channel.

Despite having a shorter offset time, the low-class burst may still successfully reserve the bandwidth if its control packet arrives before the control packet of the high-class burst, as seen in Figure 6. However, this event can be impossible if the offset time of high-class is sufficiently large, which in this case the extra offset time is greater than the maximum size of the low-class burst. [23]

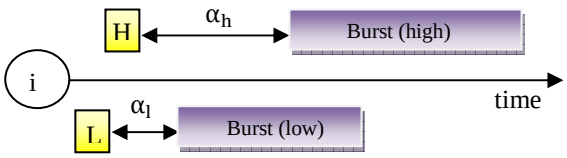

Figure 6. Illustration of how the low-class data burst successfully reserves the channel.

JET-based OBS protocol can achieve good bandwidth utilization by using delayed reservation compared to TAG-based OBS protocol; it has an intrinsic drawback in mesh networks, were the main problems are the high blocking rate, long burst latency and fairness problem.

II.5.2. A New Paradigm of Optical Burst Switching System for Lossless Transmission.

By: SeoungYoung Lee, In-Yong Hwang and Hong-Shik Park

This work proposes hybrid burst switching networks, in which networks circuit switching and burst switching schemes are mixed. To enhance network-wide multiplexing gain, they proposed source controlled burst transmission scheme. To apply this scheme, the routing table should be obtained by using the network cost function. After obtaining network-wide routing topology, the root node should be selected.

The function of root node is to schedule burst transmission and to determine cycle time for efficient transmission. The root node is selected among nodes which operate only as source nodes in routing paths. The child nodes have the ability of burst switching to the Egress Node (EN), meanwhile the VC nodes are without any switching ability and only cut through the burst by using wavelength converter. In Figure 5 is possible to see the nodes hierarchy.

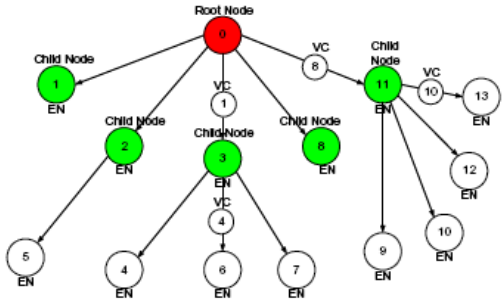

Figure 5 Hierarchy of nodes from source node 0 (extracted from [21])

With the proposed network structure, a burst loss free network can be deployed and the number of wavelengths can be reduced by the properties of statistical multiplexing gain. [21]

II.5.3. A New Absolute QoS Differentiation Scheme Supporting Best-Effort Class in OBS Networks

By Hongbo Lui and Hussein T. Mouftah.

In this paper they propose a reserve-and-preempt scheme (RPS) to provision best-effort class as much bandwidth as possible while keeping the loss rate of each priority-guaranteed class under its pre-set threshold. Simulation results show that the RPS outperforms existing schemes in terms of the loss rate of best-effort class and bandwidth utilization. [24]

II.5.4. Generalized Wavelength Sharing Policies for Absolute QoS Guarantees in OBS Networks

By Li Yang and George N. Rouskas

This paper presents a parameterized model for wavelength sharing which provides for isolation among different traffic classes while also making efficient use of wavelength capacity through statistical multiplexing. They developed a heuristic to optimize the policy parameters for a single link of an OBS network. Also it’s developed a methodology for translating the end-to-end QoS requirements into appropriate per-link parameters so as to provide network-wide guarantees. This approach can support a wide variety of traffic classes, and is effective in meeting the QoS requirements and keeping the loss rate of best-effort and overall low traffic.

The wavelength sharing policies for a single link is a technique for resource sharing among classes, but also offers each class varying degrees of protection from other classes. They assume that the unidirectional OBS link under study consists of W parallel wavelengths, and carries P classes of bursts. The

policies they consider manage the wavelength space by associating with each traffic class a pair of values that impose bounds on the use of the link's transmission resources by the class: *Wmax i*, referred to as *wavelength upper bound for class i*, is the maximum number of wavelengths that may be occupied simultaneously by bursts of class *i*, and, *Wmin i*, referred to as *wavelength lower bound for class i*, is the minimum number of wavelengths reserved by the link for class *i* bursts.

By specifying values for the pair of bounds (*Wmin i, Wmax i*) for each traffic class *i*, a policy may strike any desired balance between two conflicting objectives: QoS protection, through class separation, and efficient utilization, through sharing of wavelength resources. [25]

II.5.5. An Adaptive Reinforcement Learning-based Approach to Reduce Blocking Probability in Bufferless OBS Networks

By: Abdeltouab Belbekkouche, Abdelhakim Hafid

The performance of this work is about the use of deflection routing method in buffer-less OBS networks based on JET signaling. They propose a scheme that dynamically computes the value of offset time (OT) adapting to changes in the network state; instead of using a static value for OT.

In deflection protocols, every intermediate optical switch uses deflection routing only for contention resolution. To do so, if the default output link is reserved by another burst, the optical switch deflects the burst in question to any of other idle links.

They prohibit infinite loops due to infinite deflections by authorizing a limited number of deflections that a control packet may incur. For that purpose, they add an *NDef* field to each control packet in order to record the number of times it has been deflected, and an *NDmax* value that fixes the maximum number of authorized deflections. In addition to *SETUP* control packet, they define a new

kind of control packets called *NACK* which is sent if a burst is dropped due to Insufficient Offset Time (IOT) or if no available alternative link is found to deflect the data burst.

For the Offset Time calculation, they consider the control packet Processing Time (*PT*) on each optical switch of the path followed by the control packet from source to destination routers, the shortest path size, which is N_{SD}, and an explicit Extended Offset Time (*EOT*). So the Offset time will be calculated as: $OT = (N_{SD} \; x \; PT)$ x EOT

The EOT should not be too small, but also it should not be too large in order to alleviate increasing bursts' end-to-end delay. An adaptive scheme that computes the value of EOT dynamically is also proposed, which is an adaptive scheme in the framework of RL.

The Reinforcement Learning Model: is a learning system that accepts input from the environment, responds by selecting appropriate actions, and the environment evaluates the decisions by sending a rewarding or penalizing reinforcement feedback. Based on the value of the received reinforcement, the system updates its parameters so that good decisions become more likely to be made in the future, while bad decisions become less likely to occur.

Finally this approach is effective in reducing blocking probability at the expense of an acceptable overhead in burst end-to-end delay. [26]

II.5.6. Enhancing Bandwidth Utilization and QoS in Optical Burst Switched High-Speed Network

By: Amit Kumar Garg and R S Kaler

In this paper, the proposed scheme reduces the probability of burst contention by controlling the route at an edge router without contention resolution scheme at a core router. Each edge router selects a suitable route to the destination edge router autonomously by using feedback mechanism and prior-information

packets. They also focused on the number of resources a packet uses in the network to lower the overall resource usage and loss probability.

The Prioritized Scheduling Based on Hop-Counts (PSBHC) is a proposed prioritized scheduling scheme which is based on hop-counts, and considers several parameters, such as the packet's time in the network, the application's real-time demand, the number of resources used in the network and the numbers of resources left to use on its path in the network.

Each edge router keeps the information of all routes to each destination edge router. The priority is set for each route. The source edge router receives a feedback packet after sending a burst. When a burst is forwarded successfully, the destination edge router sends back the feedback packet that indicates the success of the transmission, whereas when a burst is discarded at an immediate core router, the core router sends back the feedback packet that indicates the failure of the transmission.

In order to improve the performance; the source edge router sends not only a burst but also some prior-information packets on the control channel. The transmission of these packets enables an edge router to find a suitable route without sending a burst payload and discarded burst numbers can also be reduced.

The simulation results showed that the proposed scheme performs better in terms of loss probability and resource waste than a best-effort network. The throughput is thus increased and the network resources are used more effectively. This is especially true for high loads and few wavelengths per fiber. By leveraging statistical multiplexing, re-configuration is minimized and bandwidth utilization is enhanced. [27]

II.5.7. Providing absolute QoS in an OBS/WR architecture

By: Bing Wu, Yang Qin, Feng Xie and Gang Feng

This paper analyzes the burst loss performance of the OBS/WR (wavelength routing) architecture and proposes a call admission control (CAC)

scheme for this architecture to provide absolute QoS, which guarantees the upper loss bounds for traffic flows. Based on that, they designed a dynamical priority allocation algorithm (DPAA) that works in conjunction with the CAC scheme to provide absolute burst loss guarantee while achieving lower blocking probability than the original CAC.

The proposed OBS/WR architecture employs OBS as its MAN part and WR as its WAN part. As can it be seen in Figure 6, these two parts are connected through hybrid conjunction nodes. Such hybrid conjunction nodes have similar architecture to the WR switching node, which will be configured for the connections for the MANs. With the OBS/WR architecture, incoming packets are aggregated at the edge router into data bursts.

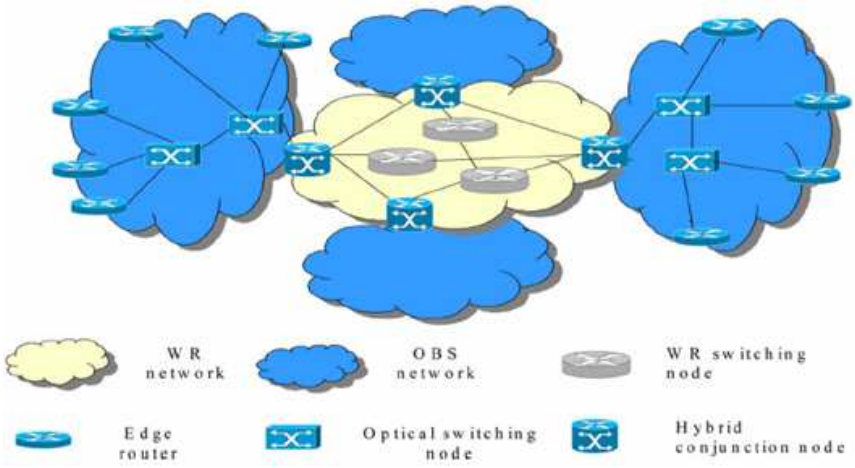

Figure 6 OBS/WR architecture (extracted from [28])

By integrating OBS and WR, some shortcomings appear. One of these is the possible burst loss in metropolitan OBS networks. In the MAN part, edge routers cannot be aware of the reservation status of the optical switching node (OSN) to avoid contentions, thus causing burst loss. To address this problem, they proposed a feedback scheme for the ingress metropolitan OBS parts of the OBS/WR architecture. The designed feedback scheme is composed of two functional components. The first component is with the OSN and the other is with the edge router. In the OSN part, the operation of the feedback scheme consists of two phases as follows:

Phase 1: Reservation. When a control packet arrives at an OSN, it attempts to reserve output bandwidth. If there is available wavelength for the time period at the desired output port, the reservation is deemed successfully. Otherwise, the reservation fails.

Phase 2: Feedback. If the reservation fails, the OSN will send a feedback packet to the source edge router of this data burst to initiate a retransmission.

The proposed feedback scheme of the edge router part is shown in Figure 7 and includes three phases as follows:

Phase 1: Assembling. All input packets are assembled into bursts according to their destinations and their class of service (CoS). The assembled data bursts are transmitted to the corresponding output unit according to their destinations.

Phase 2: Transmission. Each output unit maintains two queues, **Burst_Queue** (BQ) and **Sent_Queue** (SQ). All data bursts that arrive from the output unit of the assembling buffer are stored in the *transmission queue* for transmission. Once the output link is idle, the bursts in BQ are scanned and the burst that should be transmitted next is decided. Once a data burst in BQ is transmitted into the OBS networks, a copy of this burst should be inserted into the SQ.

Phase 3: Retransmission. All data bursts in SQ should be stored for a period of time, which is set to be the RTT from this ingress edge router to its destination egress edge router plus approximate processing time. If the edge router does not receive a feedback packet corresponding to a data burst during this period, the transmission of this data burst is considered successful and its copy will be removed from the SQ. Otherwise, this means that the transmission of the corresponding data burst fails and a tests should be performed to determine whether the data burst (in SQ) is eligible for retransmission. If the data burst is eligible for retransmission, its copy in SQ will be inserted into BQ and waits for retransmission. Otherwise, its copy will be dropped.

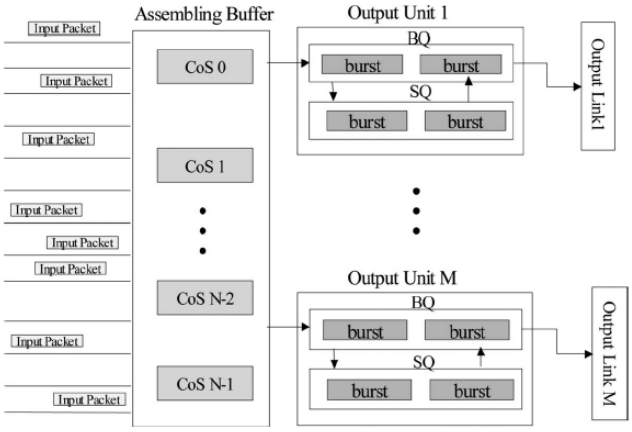

Figure 7 Schematic diagram of the feedback scheme of the edge router (extracted from [28])

An eligibility check is performed before each retransmission. They use the maximum tolerant number of retransmissions for this eligibility check to simplify the control process and minimize the feedback packet.

With the proposed CAC scheme, the edge router maintains a database, which contains the reservation status of the OSN. When a new traffic flow requires transmission, the CAC scheme gathers the necessary information about it, including the upper loss bound, the delay bound, the propagation delay, and the traffic load, from its control packet and decides whether to accept or reject the new incoming traffic flow. With this scheme, the OBS/WR architecture provides guaranteed QoS to traffic flows while maintaining high bandwidth utilization.

However, it is not the optimal solution because it blocks some unnecessarily traffic flow. To prevent this unnecessary block, in this paper, they proposed a DPAA. With the DPAA, when a new light switching path (LSP) requires establishment, the edge router allocates it as priority according to its delay bound and distance of the light path, then evaluates its loss performance and effects on existing LSPs. The new LSP is accepted if its loss bound can be guaranteed and does not violate existing LSPs' bounds. Otherwise, the priority of the new LSP will be adjusted to meet the requirements. If the requirements can be met with the new priority, the LSP is established with the new priority; if not, it is rejected. The simulations show that the proposed CAC scheme could guarantee

the burst loss threshold for LSPs. Meanwhile, the proposed DPAA achieves lower blocking probability than a regular CAC algorithm. [28]

II.6. Simulation Environment

The programming language chosen to simulate the proposed model is Java, this is because it's a portable language, allowing a flexible structure and it can be used in a simple way for the model creation and it's simulation in an Internet network.

Also, using this program language it's possible to simulate in a simple way, the related works chosen to do the comparison in order to get more convincing results.

The chosen network model to do the model study performance it's the 14-nodes NFSNET network, which is a USA network representation. It contains 14 nodes and 21 bidirectional fiber links to connect them, as shown in Figure 8. This network model was often used to evaluate related studies.

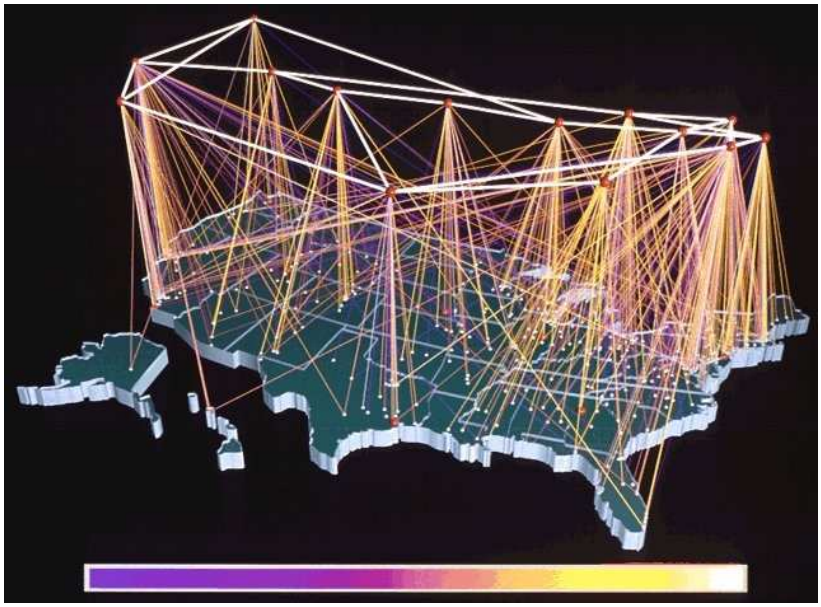

Figure 8. 14-nodes NSFNET over USA map (extracted from [29]).

In Chapter III it's explained in more details the model programming strategy and the simulation strategy, using Java.

CHAPTER III

Methodology and Development

This chapter develops the methodological framework that was used for the project. It describes in detail the methods, techniques, procedures and activities that helped meet the goals outlined in each phase.

In order to achieve mains defined in this project, a set of tools and techniques for collecting information were needed to review text bibliography and electronic contents. This activity allowed the established and defined of technical aspects needed to elaborate the new strategy studies.

The methodology consists of four (4) stages of development, as summarized in Figure 9, which allowed the creation of the new strategy model that provides QoS in OBS networks.

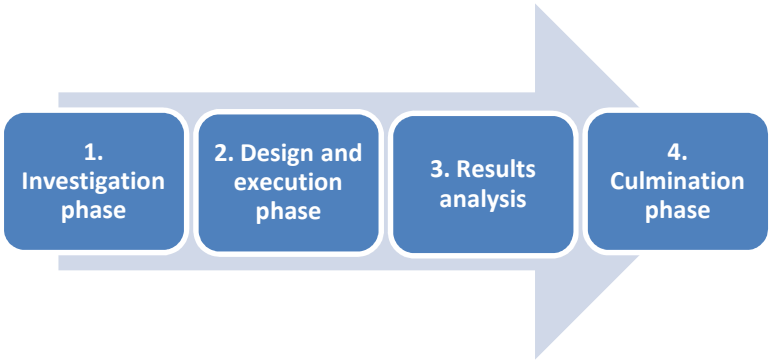

Figure 9. Methodology phases for the investigation developed.

III.1. Investigation Phase

This phase is the project starting point. For its performance, each one of the features and specifications of optical networks and the different mechanisms that allow QoS on them were studied and analyzed.

From now on, activities and preliminary results obtained in this stage are exposed.

III.1.1. Bibliography Review

This activity consisted of the review and consultation of theoretical sources in order to obtain a support that would enable an understanding of different topics related to the project, also enabling the development of the strategy, specifications and procedures to be carried out in the tests. For the documentation of this project, various authors and articles based in optical and electrical networks, were consulted; primarily from the IEEE Magazine, Yoo, M. and Qiao C. and other publications from the Optical Network Forum.

III.1.2. Related Works Documentation

The objective was to study and analyze each of the related works, then choose which ones were key points on my model creation and once this was done, use some of its strategies and modify in order to improve their works.

Research allowed identifying the similarities and differences, at a technical and logical level, between the consulted studies and this project.

Some of the more important related works used to create and test my strategy are exposed in Chapter II in the Related Works section.

III.2. Design and Execution Phase

In this phase, the strategy model which I created after all the literature review, it's explained. After its explanation, the environment used for the test is described. It's important to say that in order to go to the execution phase; the model was revised by the tutor of this work Xavier Hesselbach, who followed the strategy buildup step by step.

III.2.1. New Strategy Creation

First of all, the new strategy found in order to provide absolute QoS is described bellow and the different phases that conforms it are also explained. A scheme of it, its shown in Figure 10.

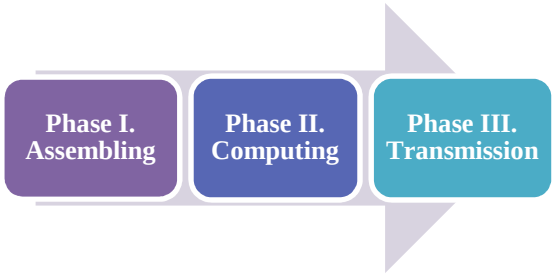

Figure 10. Phases of the New Strategy.

III.2.1.1. Phase I. Assembling

In this phase, the electrical packets will arrive and will be transformed into bursts. These data bursts will be assembled according to the same CoS and destination node. These will also have a fixed size depending at which CoS it belongs. For data burst that belong to CoS_0 will have a length equal to 32KB, meanwhile the burst length for CoS_1 will be twice the burst length of CoS_0, which is equal to 64KB. For the following *CoS* it's two times the previous *CoS*, so for the CoS_{k-1} the burst length will be $2 \times CoS_{k-2}$, were the higher priority *CoS* will be represented for CoS_0 and the lower priority for the CoS_{k-1}. The total number of classes will be $k \in [0, k-1]$. A scheme of the assembling process can be seen in Figure 11.

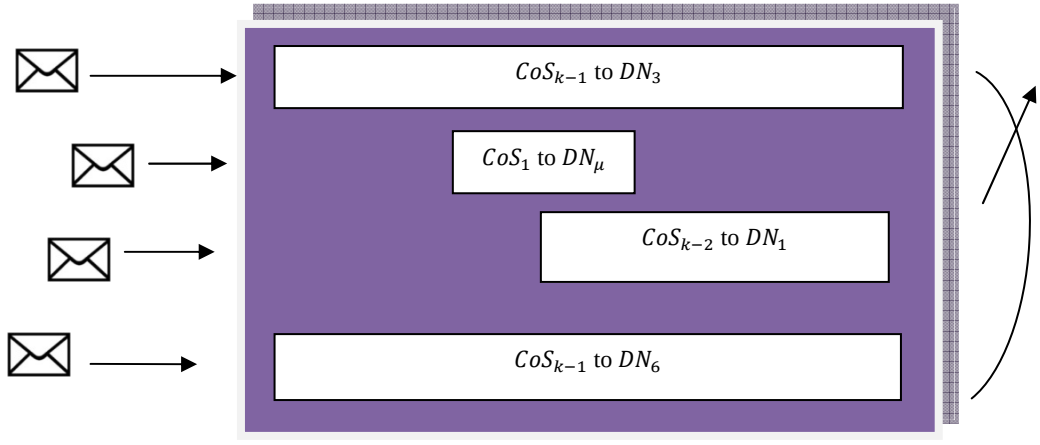

Figure 11. Assembling Process

III.2.1.2. Phase II. Computing

When a BCP packet of a specific data burst arrives to an edge router, which is assigned depending to its destination node, a route will be assigned and also an OT depending of its *CoS*. In order to do this phase, every edge router keeps information from all the available routes to reach other edge routers. Also each route will be associated to a priority P and a N_f value that will indicate the number of times that this route has been chosen and also an H value that indicates the number of nodes were the burst must pass to reach the destination node. The P parameter will be a number between$[0,1]$, meanwhile the N_f will be a real number. [6]

An example of the routing table, were all the routes to reach a specific destination node, are shown in Table 3. This will be adjusted any time a data burst has expired in the *waiting queue.* A success occurs when the time expires, without receiving a feedback packet, as shown in Equation 1.a. Meanwhile if the *NACK* packet is received before, a failure is calculated, as shown in Equation 1.b. The N_f value will be calculated in the same way for both cases, as shown in Equation 1.c.

Route	*P* factor	N_f factor	*H*
3-4-7-2-9	0,98	6	5
8-6-7-5-2-7-9	0,98	4	7
12-6-4-6-9	0,65	3	5
1-4-5-6-7-3-9	0,34	8	7
3-6-5-9	0,12	1	4

Table 3. Example of the existing *Routing Table* in each edge router

$$P = P \qquad P = \frac{P \times N_f}{N_f + 1} \qquad N_f = N_f + 1$$

Equation 1.a. P calculation assuming a success. 1.b. P calculation assuming a failure. 1.c. N_f calculation. (from left to right) (extracted from [6])

The route will be chosen following the next priorities: The first parameter to see is the P value. The route with the biggest value will be chosen if more than

one route has the same value. The second parameter to take into account will be the smaller *H* value.

When a route has been chosen, it's going to be introduced into the BCP packet, which contains also the information about the source edge router, destination edge router and the *CoS.*

After the route is selected, the offset time must be calculated, as shown in shown in Equation 2, and set, which will be carried by BCP packets and it will let know the intermediate OBS nodes for how long they have to reserve the wavelength channel for its specific data burst.

$$OT_{CN} = \left(t_{bcp} + t_p + t_c + EOT_{CN}\right) \times h_i \qquad EOT_{CN} = \left(\sum_{i=0}^{k-1} BL_i - 2BL_{CN}\right) \times \frac{8}{b_{core}}$$

Equation 2.a. Offset Time calculation, Equation 3.b. Extra-Offset time calculation (from left to right)

Were:

OT_{CN}: The offset time for each class number

t_{bcp}: The delay time to transmit the *BCP* packet

t_p: The delay time to process the *BCP* packet

t_c: The matrix switching time

EOT_{CN}: The extra offset time calculation for each class number

h_i : Represents the number of jumps that the burst needs to do before arriving to the *i* node, from the source node.

CN: The number class, were $CN \in [0, k-1]$

BL_i: The burst length size according to the CoS

b_{core}: The output bandwidth for each wavelength channel

H: The total number of jumps from the source edge router to the destination edge router.

By knowing the OT for each *CoS* data burst, it's possible to obtain the Time-to-Live (*TTL)* for each packet, which can be calculated with the Equation

1Equation 4, Twice this time will be the Round Trip Time (RTT) that the copy will be placed into the *waiting queue*.

$$TTL = H \times \left[OT + \left(BL \times \frac{8}{b_{core}} \right) \right]$$

Equation 4. Time-to-Live calculation

III.2.1.3. Phase III. Transmission Phase

This phase is composed of two functional components. The first component is the OSN (Optical Switch Node) and the other is the edge router. In each component, two phases will be explained.

III.2.1.3.1. The OSN Component

In the OSN part, the operation of the feedback scheme consists of two phases as follows:

- **Reservation phase:**

When a control packet arrives at an OSN, it attempts to reserve output bandwidth, a time period in advance for the data burst that is expected to arrive later at a specific offset time. If there is available wavelength for the period at the desire output port, the reservation is deemed successfully. Otherwise, the reservation fails, and the second phase takes place.

- **Feedback phase:**

The OSN will send back a feedback packet to the source edge router of this data burst to, initiate the transmission as it corresponds, if the number of feedback of this data burst doesn’t exceed the maximum number of retransmissions. The feedback packet will contain information about the sequence number of the collision data burst and the source edge router’s address.

III.2.1.3.2. The Edge Router Component

The edge router will be in charge of the data bursts transmission and retransmission operations. It's performance can be seen in Figure 12.

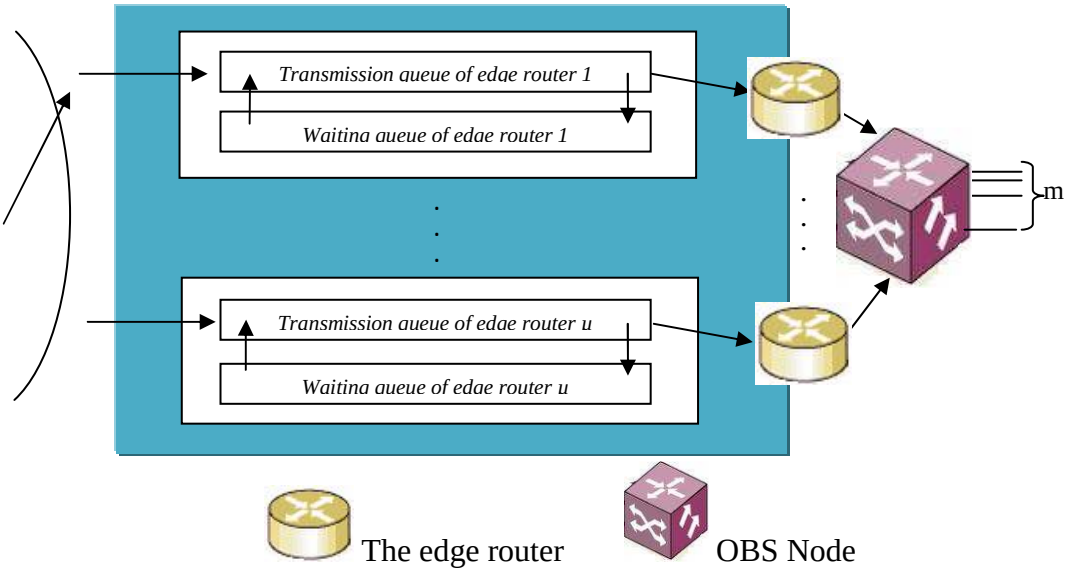

Figure 12. The Edge Router Function

- **Transmission:**

Each output unit maintains two queues, the *transmission queue* and the *waiting queue*, as seen in Figure 13. All data bursts that arrive from the output unit of the computing phase are stored in the *transmission queue* for transmission.

This queue will perform an earliest deadline first (EDF) scheduling algorithm for scheduling the bursts, and the theory of this discipline is that the first burst to be sent will be the one with the earliest time to live. This scheduling algorithm was chosen in order to keep the *CoS* priority and also because it's often used in real-time operating systems.

Once the bursts are scheduled, and the output link is idle, the burst will be sent to the optical network and a copy will be stored in to the *waiting queue* for a period of time, which is specified in the next section.

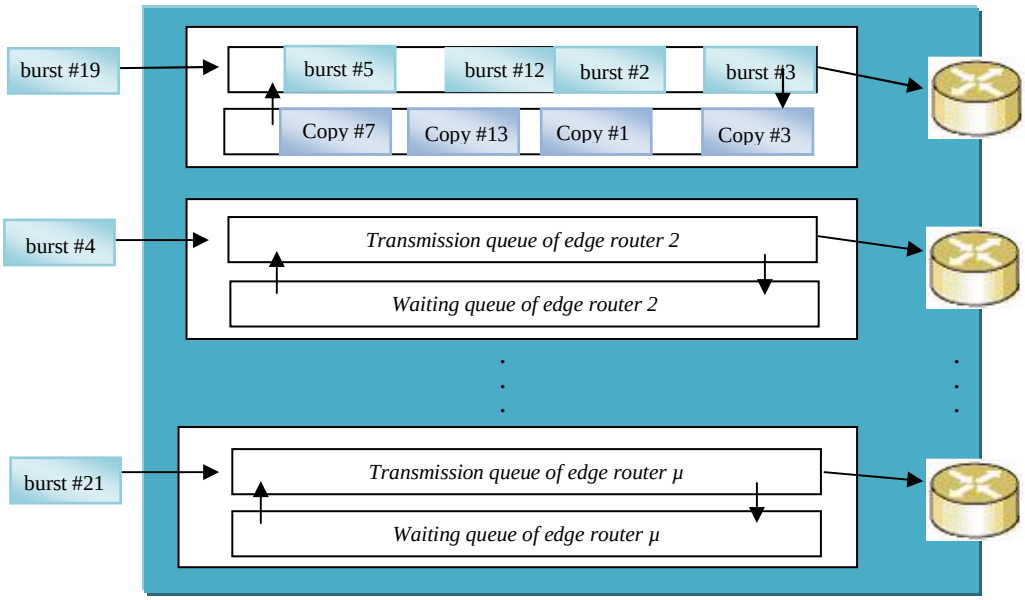

Figure 13. Transmission and Retransmission scheme

- **Retransmission:**

As I said before all the data burst that are sent to the optical networks are stored in the *waiting queue* for a period of time equal to the RTT from the source edge router to the destination edge router, plus a processing time.

If the source edge router receives a feedback packet before the established time expires, the system takes this as a failure and the *waiting queue* will send the copy to the *transmission queue* and remove from its queue, this is done, after its retransmissions numbers are evaluated without exceeding the N_{max} established parameter. Otherwise, if the edge router doesn't receive a feedback packet in this period of time, the system will consider that the data burst arrived to the destination node and the *waiting queue* will drop its copy.

III.2.2. Execution Phase

Following, the structure and performance of the designed model test is presented. In order to study the proper functioning of the model, I will make a series of tests to the system. First, the model operation will be tested and then the simulation of an optical network. In both scenarios, the Java programming language was used. Each test is explained in a programming point of view.

III.1.1.1. Operation Model Tests

In order to study if the model is working correctly, it will be divided into sections and each one will be tested separately. Once, it has been proofed that each one works correctly, the model will be tested altogether.

III.1.1.1.1. Test #1: BCP Packet Creation

- **General Main:**

The General main is to study if the control packet creation is performed in a correct way inside the model, taking electric traffic packets as input, and having control packets ready to go to the transmission queue as output.

- **Values Involved:**

In this fist section, the programming was based in classes; the done classes were the following: *phase1*, *phase2* and *phase2b* .These three (3) classes involved methods that were in charge of the bcp packet construction. This will be explained later. Also other kind of classes are created, which are: *packetelec*, *packets*, *packest2* and *packestbcp*; these types of classes have the objects properties creation, which are going to be passed from one class to other. Another created class in the project was the one in charge of the electronic traffic creation and finally *Controller*, which is the main program, which it's going to be in charge of controlling the model operation. The diagram is shown in Figure 14.

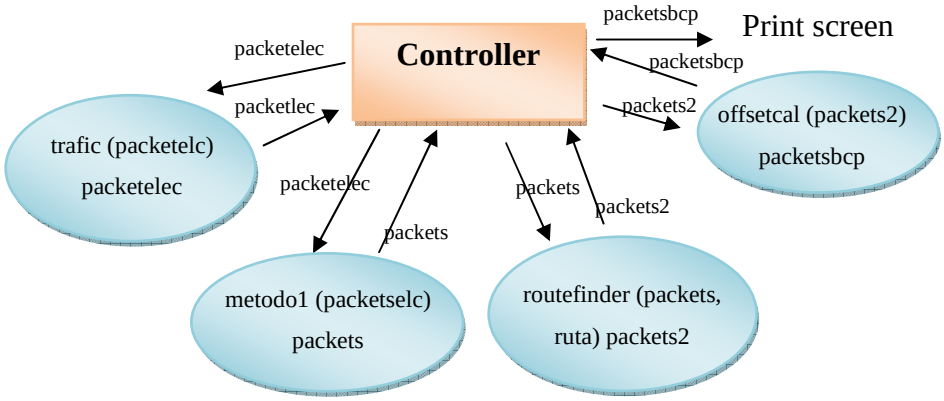

Figure 14. Model Diagram for the BCP packets creation.

Traffic: This phase will be in charge of creating electronic packets, assigning the destination node, CoS and data information. All these packets have a fixed value, which depend of the CoS, being the CoS1 the smallest, and the following one twice the size of the one before. The returned packet will be also an electronic packet named packetele, with modifies parameters.

Phase1: This phase will be in charge to take an electronic packet, which comes from the *electronic traffic* class and will be introduced in *metodo1*. *T*his method will remove the destination node and CoS information and will classify the packets according to the parameters and start the bcp packet creation, adding the source address (SA), destination address (DA), CoS and sequence number (Nseq). The return bcp packet is named packet2.

Phase2: This phase will be in charge to take a packet, which comes from *phase1* class and a route which will come from *ruta* class. For this kind of test a static value will be used which will be later modified for future testing. These two values are introduced into *routefinder.* This method will add the route that the bcp packet must follow, according to the destination node information and will add these parameters to the previous bcp packet, which are the route to follow, the route priority (P). The number of feedbacks (Nf) that this route has to suffer and the number of jumps (H) that the burst will experiment. The return bcp packet is named packet2.

Phase2b: this phase will be in charge to take a packet2, which comes from the *phase2* class, and introduced to the *offsetcal.* This method will use the CoS information and the number of jumps that the burst must follow, to calculate the offset time (OT), the time to live (TTL), also, the number of the feedbacks counter (N) is set to cero, all these values are going to be added to the bcp packet. Finally, after this class is executed, a bcp packet of the packetbcp, it's returned as shown in Figure 15.

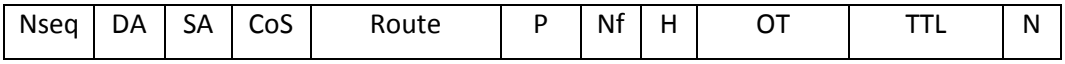

Figure 15. BCP packet

Were:
Nseq: It's the sequence packet number
DA: It's the packet destination address
SA: It's the packet source address
CoS : It's the packet CoS
Route: It's the route that the burst must follow
P: It's the route priority
Nf: It's the route number of feedbacks
N: It's the route number of hopes
OT: It's the packet offset-time
TTL: It's the burst time to live
N: It's the packet number of feedbacks

In Table 4 is possible to see the used values in this section with its respective unit in the user point of view.

Values	Unit
Packetelec, Packets, Packets2, Packetsbcp	packets/u.t.
OT, TTL	u.t.

Table 4. Units of used values in test #1

- **Methodology:**

In order to see if this model section is working, a first test is done, using fixing parameters in the *ruta* class, and after each phase a print of the set values is done. Just one traffic of each class will be introduced.

The introduced route value will be "1-2-3-7-8-9" with a priority of "0.78", number of feedbacks equal to "7" and number of jumps "6". The first packet will have as destination node "0" and CoS "1". Finally I should have a resultant bcp packet as shown in Figure 16.

1	0	0	1	1-2-3-7-8-9	0.78	7	6	0.0001981	0.0013422	0

Figure 16. Resultant BCP packet with DN=0 and CoS=1

Using the same methodology a bcp packet of a burst with CoS2 and different parameters, as shown in Figure 17, will be also tested. And finally a bcp packet of a burst with CoS3 and different parameters, as also shown in Figure 18, will be also tested.

2	3	3	2	1-10-3-12	0.98	4	4	0.0001725	0.0008948	0

Figure 17. Resultant BCP packet with DN=3 and CoS=2

3	5	5	3	2-3-7	0.89	12	3	0.0001213	0.0006711	0

Figure 18. Resultant BCP packet with DN=5 and CoS=3

- **Results:**

After compiling and running the first test, the results were the following:

```
the electronic packet will be 0 1 aaaa
the packet before phase 1 will be 0 1
the packet after phase 1 will be 1 0 0 1
the packet before phase 2 will be 1 0 0 1
the route assignment in phase 2 will be 1-2-3-7-8-9 0.78 7 6
the packet after phase 2 will be 1 0 0 1 1-2-3-7-8-9 0.78 7 6
the packet before phase 2b will be 1 0 0 1 1-2-3-7-8-9 0.78 7 6
the bcp packet after phase 2b will be 1 0 0 1 1-2-3-7-8-9 0.78 7 6
1.981E-4 0.0013422 0 2
the bcp packet will be 1 0 0 1 1-2-3-7-8-9 0.78 7 6 1.981E-4
0.0013422 0
```

It's possible to see that the bcp packet was complete as expected. After compiling and running the second test in order to get a bcp packet for a burst CoS2, the results were the following:

```
the electronic packet will be 3 2 aaaa bbbb
the packet before phase 1 will be 3 2
the packet after phase 1 will be 2 3 3 2
the packet before phase 2 will be 2 3 3 2
the route assignment in phase 2 will be 1-10-3-12 0.98 4 4
the packet after phase 2 will be 2 3 3 2 1-10-3-12 0.98 4 4
the packet before phase 2b will be 2 3 3 2 1-10-3-12 0.98 4 4
the bcp packet after phase 2b will be 2 3 3 2 1-10-3-12 0.98 4 4
1.725E-4 8.948E-4 0 4
the bcp packet will be 2 3 3 2 1-10-3-12 0.98 4 4 1.725E-4 8.948E-
4 0
```

With these results, it's possible to see that the bcp packet was complete as expected. After compiling and running the last test for the bcp packet creation of a burst with CoS3, the obtained results were:

```
the electronic packet will be 5 3 aaaa bbbb cccc dddd
the packet before phase 1 will be 5 3
the packet after phase 1 will be 3 5 5 3
the packet before phase 2 will be 3 5 5 3
the route assignment in phase 2 will be 2-3-7 0.89 12 3
the packet after phase 2 will be 3 5 5 3 2-3-7 0.89 12 3
the packet before phase 2b will be 3 5 5 3 2-3-7 0.89 12 3
the bcp packet after phase 2b will be 3 5 5 3 2-3-7 0.89 12 3
1.213E-4 6.711E-4 0 6
```

```
the bcp packet will be 3 5 5 3 2-3-7 0.89 12 3 1.213E-4 6.711E-4 0
```

- **Conclusions:**

Finally, with these results, the model test was done, and it's possible to say that it works correctly for the bcp packet creation, for each class and destination node. Knowing these, it's possible to test the other methods and classes that involved the complete model operation.

III.1.1.1.2. Test #2: Routes Table and Calculation

- **General Main:**

The General main is to study if the Route Table is working correctly inside the model, looking if the set route is the one that has the best performance.

- **Values Involved:**

None of the used parameters in this section have any value unit from the user or programming point of view.

- **Methodology:**

In order to test this section, a network topology was established with the idea to get real routes to go from one source node (SN) to a destination node (DN), through optical nodes (ON). In Figure 19 the routes to go from SN0 to all DN, are shown.

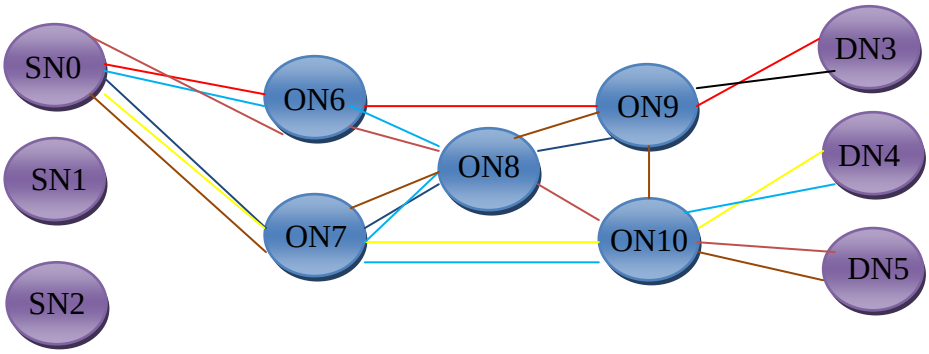

Figure 19. Established Network Topology, from SN0 to all DN

Having this topology, it's possible to set at least two routes to go from one source address to a destination node. To do this test section, the routes from SN0 to each DN are established with a priority P, the number of feedbacks Nf, and the number of optical nodes jumps H, as seen in Table 5.

SN	DN	ROUTE	P	Nf	H
0	3	6 – 9	0.96	3	2
		7 - 8 – 9	0.81	2	3
0	4	7 – 10	0.56	5	2
		6 - 8 - 7 – 10	0.86	1	4
0	5	6 - 8 – 10	0.9	3	3
		7 - 8 – 9 – 10	0.03	6	4

Table 5. Routetable for SN 0 to all DN

The expected results on this section are to obtain the routes with the higher P value after assigning the DN, and also all the route values for this specific priority. The purpose to obtain these values is to set the best route that the burst must follow.

- **Results:**

To find the route from SN0 to DN3, after executing the class *route0,* the obtained results were:

```
The possible routes to get to the destination node 3 from the
source node 0, are:
Route   P    Nf    H
6-9   0.96  3.0   2.0
7-8-9 0.81  2.0   3.0
The selected route will be:
6-9   0.96  3.0   2.0
```

To find the route from SN0 to DN4, after executing the class *route0,* the obtained results were:

```
The possible routes to get to the destination node 4 from the
source node 0, are:
Route        P     Nf    H
6-8         0.56  5.0   2.0
6-8-7-10    0.86  1.0   4.0
The selected route will be:
6-8-7-10    0.86  1.0   4.0
```

Finally, to find the route from SN0 to DN5, after executing the class *route0,* the obtained results were:

```
The possible routes to get to the destination node 5 from the
source node 0, are:
Route        P     Nf    H
6-8-10      0.9   3.0   3.0
7-8-9-10    0.03  6.0   4.0
The selected route will be:
6-8-10      0.9   3.0   3.0
```

- **Conclusions:**

With these results, it's possible to say that the obtained route tables are the same as the one shown in Table 5, and the chosen route is the expected one, which is the route with the higher P value.

III.1.1.1.3. Test #3: Transmission Queue

Once the bcp packet is done, it will pass to the transmission queue and wait for an idle link to be transmitted. This queue will work using as scheduler algorithm, the EDF. In order to simulate it, the *Bubble Sort* mechanism was implemented using the TTL value, which can be obtained in the bcp packet.

- **General Main:**

The General main is to study if the transmission queue is working correctly in its 3 phases, which are the push phase, when a new control packet arrives, the Bubble Sort mechanism and the pop phase.

- **Values Involved:**

The transmission queue will be tested. This is composed by three (3) different classes: *push, pop* and *cola* with the methods: *agregar, sacar* and *bubble* respectively, which are going to be explained later. Also there is a class named *colaor,* which is the queue properties creation and the transmission queue representation. And in order to control all these classes a main named *Controller2* was created, which works as seen in Figure 20 .

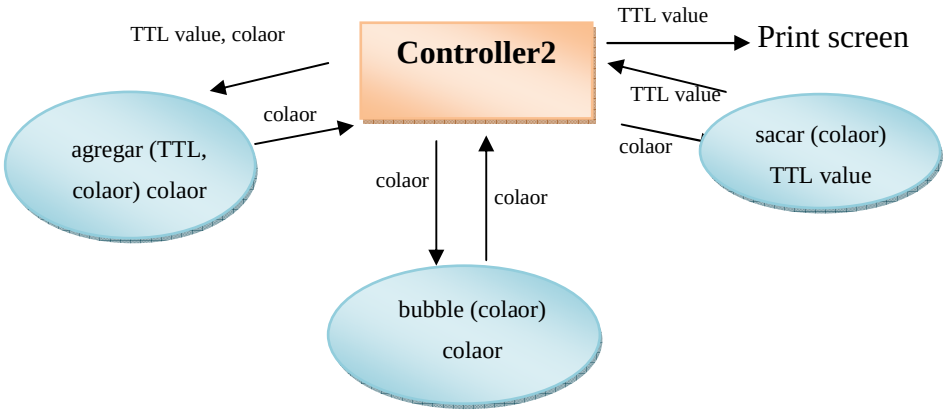

Figure 20. Model Diagram for the model transmission queue.

Push: This class will be in charge to take the *colaor* (transmission queue) and a *packetsbcp*, which comes from the *phase2b* class, and removed the TTL value in order to introduce into the *agregar* method, which will add the TTL value to the previous *colaor*. Finally this class, will return the new *colaor*.

Cola: This class will be in charge to take the c*ola*, which comes from the *push* class, and introduce it into the *bubble* method, which works by repeatedly stepping through the list to be sorted, comparing two items at a time and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps *colaor*.

Pop: This class will be in charge to take the *colaor*, which comes from the *cola* class, and introduce it into the *sacar* method, which will take the first queue element, which is the one with the smaller TTL value. Finally this class will return the TTL value of the bcp packet that can be transmitted.

In Table 6 is possible to see the used values in this section with its respective unit in the user point of view.

Values	Unit
Packetsbcp	packets/u.t.
TTL	u.t.

Table 6. Units of used values in test #3

- **Methodology:**

In order to test this model section, some parameters are going to be introduced in the transmission queue, which as I said before is represented by the class colaor, using the *agregar* method, which is the push mechanism.

Once a new value is introduced, the EDF algorithm is done, in which the *bubble* method is used for organizing all the existent packets in the *transmission queue*, which at the same time, will be waiting a pop while using the *sacar* method. What it's expected, is that every time a packet is created, it should be introduced in the transmission queue and the EDF algorithm should be executed. Once the networks are ready to send a packet, a pop will occur to the packet with smallest TTL value.

- **Results:**

```
Pushing ttl value = 7.8
The transmission queue after the first push will be 7.8
Pushing ttl value = 3.8
The transmission queue after the second push will be 3.8 7.8
Pushing ttl value = 7.0
The transmission queue after the third push will be 3.8 7.0 7.8
Pushing ttl value = 17.8
Pushing ttl value = 0.8
Pushing ttl value = 12.6
Finally, after six pushes and apply the edf mechanism, the queue
will be 0.8 3.8 7.0 7.8 12.6 17.8
Popping the first value
If the output link is idle, the bcp packet with the ttl= 0.8 is
the one to be pop
```

- **Conclusions:**

As seen in the obtained results, the transmission queue is growing up each time that a push is performed, and the EDF algorithm is also performed after each push, as it was expected. Also it's possible to see that the output expected value, after a pop is performed is the expected, which is the 0.8 value, which is the smallest value in the transmission queue. With these results it's possible to say that the transmission queue works correctly and it was expected.

III.1.1.1.4. Test #4: All the Model together without the feedback scheme

Finally, this will be the last test done to the model performance. In this section all the models will work together, and when the output link is ready to send the data burst, the bcp packets will be created for their transmission.

- **General Main:**

The general main is to study the performance of the model altogether by joining all the previous phases which already have been tested.

- **Values Involved:**

In Table 7 is possible to see the used values in this section with its respective unit in the user point of view.

Values	Unit
Packetelec, Packets, Packets2, Packetsbcp	packets/u.t.
OT, TTL	u.t.

Table 7. Units of used values in test #4

- **Methodology:**

The transmission queue will be performed in the following way: electric traffic will be created, and then their respective bcp packets. These packets will be sent to the *transmission queue* and will be waiting for an idle link, which is simulated using the pop function. Once the packet is ready to be sent, all the necessary bcp packets are generated for sending the data burst. The number of bcp packets is equivalent to the number of optical links in which the data burst must pass.

It's also important to say that in this test, the network that is going to be used for the simulation is the one seen in Figure 21, with the idea of creating the adequate route table for the control parquets which will be travelling through the network.

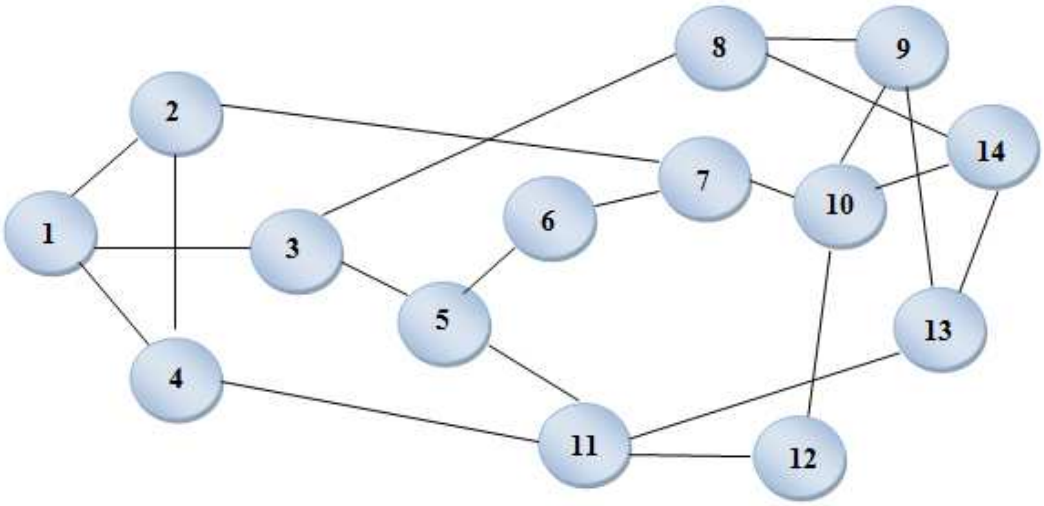

Figure 21. 14-Nodes NSFNET represented with numbers.

For each source node, I will have a routing table, and at least 2 routes to reach the destination as shown in Table 8, were is possible to see the routing table for source node 1.

Destination Node	Route	Hops
1	1	0
2	1-2	1
	1-4-2	2
3	1-3	1
	1-4-11-5-3	4
4	1-4	1
	1-2-4	2
5	1-3-5	2
	1-4-11-5	3
6	1-3-5-6	3
	1-2-7-6	3
7	1-2-7	2
	1-3-5-6-7	4
8	1-3-8	2
	1-2-7-10-9-8	5
9	1-3-8-9	3
	1-2-7-10-9	4
10	1-3-8-9-10	4
	1-3-5-6-7-10	5
11	1-4-11	2
	1-3-5-11	3
12	1-4-11-12	4
	1-3-5-11-12	4

13	1-4-11-13	4
	1-3-5-11-13	4
14	1-3-8-14	3
	1-3-5-11-13-14	5

Table 8. Routing Table for Source Node 1

The idea is to create, in a random way, electric traffic and their respective control packets, then all these packets are going to be introduced, one by one, in the *transmission queue.* Then, after a period of time, a pop class is simulated and the expected result is to obtain the control packet with the smaller TTL value and create all the respective bcp packets to send to the optical network. In order to be able to follow each packet, is important to see the control packet format show in the first test. Remember that the sequence number is given for the first number that appears in each packet. Also a format for the bcp packets that are going to be send to the optical network is shown in Figure 22.

Nseq	DA	SA	CoS	Link to reserve	DNO	DNFO	OT	N

Figure 22. BCP packet for each optical node

- **Results:**

The obtained result is the following:

```
The electric packet will be 1.0    2     1     aaaa
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.96  3.0   2.0
1-4-2       0.81  2.0   3.0
The selected route will be:
1-2         0.96  3.0   2.0
the bcp packet will be 1.0 2 1   1 1-2 0.96 3.0 2.0 1.981E-4
4.474E-4 0

The electric packet will be 2.0    3     1     aaaa  bbbb
The possible routes to get to the destination node 3 from the
source node 0, are:
Route         P    Nf    H
1-3         0.96  3.0   2.0
1-4-11-5-3  0.81  2.0   5.0
The selected route will be:
1-3         0.96  3.0   2.0
the bcp packet will be 2.0 3 1 2 1-3 0.96 3.0 2.0 1.725E-4 4.474E-
4 0
```

```
The electric packet will be 3.0    4     1     aaaa  bbbb  cccc
      dddd
The possible routes to get to the destination node 4 from the
source node 0, are:
Route         P    Nf    H
1-4         0.56  5.0   2.0
1-2-4       0.86  1.0   3.0
The selected route will be:
1-2-4 0.86  1.0   3.0
the bcp packet will be 3.0 4 1 3 1-2-4 0.86 1.0 3.0 1.213E-4
6.711E-4 0


The electric packet will be 4.0    5     1     aaaa
The possible routes to get to the destination node 5 from the
source node 0, are:
Route         P    Nf    H
1-3-5       0.9   3.0   3.0
1-4-11-5    0.03  6.0   5.0
The selected route will be:
1-3-5 0.9   3.0   3.0
the bcp packet will be 4.0 5 1 1 1-3-5 0.9 3.0 3.0 1.981E-4
6.711E-4 0


The link is idle and ready for transmission:
the extracted packet will be 2.0 3 1 2 1-3 0.96 3.0 2.0 1.725E-4
4.474E-4 0
the first bcp packet to send to the optical node will be 2.0 3 1 2
1-3 1 3 1.725E-4 0
the second bcp packet to send to the optical node will be 2.0 3 1
2 1-3 3 0 3.45E-4 0

The link is idle and ready for transmission:
the extracted packet will be 1.0 2 1 1 1-2 0.96 3.0 2.0 1.981E-4
4.474E-4 0
the first bcp packet to send to the optical node will be 1.0 2 1 1
1-2 1 2 1.981E-4 0
the second bcp packet to send to the optical node will be 1.0 2 1
1 1-2 2 0 3.962E-4 0


The link is idle and ready for transmission:
the extracted packet will be 4.0 5 1 1 1-3-5 0.9 3.0 3.0 1.981E-4
6.711E-4 0
the first bcp packet to send to the optical node will be 4.0 5 1 1
1-3-5 1 3 1.981E-4 0
the second bcp packet to send to the optical node will be 4.0 5 1
1 1-3-5 3 5 3.962E-4 0
the third bcp packet to send to the optical node will be 4.0 5 1 1
1-3-5 5 0 5.943E-4 0
```

- **Conclusions:**

The results are the expected. It was possible to see, that the bcp packets were created, based on electronic packets as the proper way, and they were

introduced in the transmission queue. Once it was possible to remove a packet from the transmission queue, to send it to the networks, the chosen one, was the control packet with the smaller TTL value. Once this packet is found in a data base, all the bcp packets needed to send to the optical network, (which are the same amount that the h value) are created and ready to send to its respective optical node. Also it's possible to see that the ot value for each of these new bcp packets are different and their value is the number of jumps that the data burst must do before arriving to the optical network were they must go, by the ot value contained by the control packet.

III.1.1.1.5. Test #5: Table Adjustment by Feedback

- **General Main:**

The General main is to study if the routing tables are being adjusted after the feedback scheme is performed, obtaining a failure or success, when a bcp packet tries to reserve an optical link in the network.

- **Values Involved:**

In this section I will work with a new type of packets, which are the NACK packets, that perform if a link cannot be reserved. This packet's unit is the packets/u.t. Also, some new classes play an important role in this section. The *waiting queue* was created, which is able to take a packet that is popped from the transmission queue, using the *WQpush* class, and stored until the TTL time is expired or a NACK packet is received.

Once the packet expires its time, the waiting queue will remove the packet and send a result that indicates a success. But in the other case, if a NACK is received the waiting queue, it will remove the packet and send a result that indicates a failure, this method is performed by the *WQpop* class.

Once these results are obtained, the adjustable class will be in charge to recalculate the routes and change the old routing table for the new one, in order to get better performance based on statistical results.

- **Methodology:**

In order to study if the table adjustment is doing well, I will generate in a random way success and failures results that the *waiting queue* should take and use to remove the bcp packets from its queue and send information for the readjustment of the routing table. The study link in this case will be the link 1-2, and the results could be seen in the table adjustment any time a failure o success is indicated.

- **Results:**

The obtained result is shown next:

```
The electric packet will be 1.0    2     1     aaaa
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.7   3.0   2.0
1-4-2       0.81  2.0   3.0
the bcp packet will be 1.0 2 1   1 1-4-2 0.81 2.0 3.0 1.981E-4
6.711E-4 2
the extracted packet will be 1.0 2 1 1 1-4-2 0.81 2.0 3.0 1.981E-4
6.711E-4 2
The stored packets in the WQ are: 1.0
A success occurs in route: 1-4-2
The route to change will be 32
the old priority is 0.81 and the Nf were 2.0
the new priority will be 0.81 the new Nf is 3.0

The electric packet will be 2.0    2     1     aaaa  bbbb
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.7   3.0   2.0
1-4-2       0.81  3.0   3.0
the bcp packet will be 2.0 2 1   2 1-4-2 0.81 3.0 3.0 1.725E-4
6.711E-4 4
the extracted packet will be 2.0 2 1 2 1-4-2 0.81 3.0 3.0 1.725E-4
6.711E-4 4
The stored packets in the WQ are: 2.0 1.0
A failure occurs in route: 1-4-2
The route to change will be 32
the old priority is 0.81 and the Nf were 3.0
the new priority will be 0.6075 the new Nf is 4.0

The electric packet will be 3.0    2     1     aaaa  bbbb  cccc
     dddd
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
```

```
1-2         0.7   3.0   2.0
1-4-2       0.6075      4.0   3.0
the bcp packet will be 3.0 2 1  3 1-2 0.7 3.0 2.0 1.213E-4 4.474E-
4 6
the extracted packet will be 3.0 2 1 3 1-2 0.7 3.0 2.0 1.213E-4
4.474E-4 6
The stored packets in the WQ are: 3.0 2.0 1.0
A success occurs in route: 1-2
The route to change will be 31
the old priority is 0.7 and the Nf were 3.0
the new priority will be 0.7 the new Nf is 4.0

The electric packet will be 4.0    2     1     aaaa
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.7   4.0   2.0
1-4-2       0.6075      4.0   3.0
the bcp packet will be 4.0 2 1  1 1-2 0.7 4.0 2.0 1.981E-4 4.474E-
4 2
the extracted packet will be 4.0 2 1 1 1-2 0.7 4.0 2.0 1.981E-4
4.474E-4 2
The stored packets in the WQ are: 4.0 3.0 2.0 1.0
A failure occurs in route: 1-2
The route to change will be 31
the old priority is 0.7 and the Nf were 4.0
the new priority will be 0.5599999999999999 the new Nf is 5.0

The electric packet will be 5.0    2     1     aaaa  bbbb
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.559 5.0   2.0
1-4-2       0.607 4.0   3.0
the bcp packet will be 5.0 2 1  2 1-4-2 0.6075 4.0 3.0 1.725E-4
6.711E-4 4
the extracted packet will be 5.0 2 1 2 1-4-2 0.6075 4.0 3.0
1.725E-4 6.711E-4 4
The stored packets in the WQ are: 5.0 4.0 3.0 2.0 1.0
A success occurs in route: 1-4-2
The route to change will be 32
the old priority is 0.6075 and the Nf were 4.0
the new priority will be 0.6075 the new Nf is 5.0

The electric packet will be 6.0    2     1     aaaa  bbbb  cccc
      dddd
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.559 5.0   2.0
1-4-2       0.607 5.0   3.0
the bcp packet will be 6.0 2 1  3 1-4-2 0.6075 5.0 3.0 1.213E-4
6.711E-4 6
the extracted packet will be 6.0 2 1 3 1-4-2 0.6075 5.0 3.0
1.213E-4 6.711E-4 6
The stored packets in the WQ are: 6.0 5.0 4.0 3.0 2.0 1.0
A failure occurs in route: 1-4-2
The route to change will be 32
```

```
the old priority is 0.6075 and the Nf were 5.0
the new priority will be 0.50625 the new Nf is 6.0

The electric packet will be 7.0    2    1    aaaa
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P    Nf    H
1-2         0.559 5.0   2.0
1-4-2       0.506 6.0   3.0
the bcp packet will be 7.0 2 1  1 1-2 0.5599999999999999 5.0 2.0
1.981E-4 4.474E-4 2
the extracted packet will be 7.0 2 1 1 1-2 0.5599999999999999 5.0
2.0 1.981E-4 4.474E-4 2
The stored packets in the WQ are: 7.0 6.0 5.0 4.0 3.0 2.0 1.0
A success occurs in route: 1-2
The route to change will be 31
the old priority is 0.5599999999999999 and the Nf were 5.0
the new priority will be 0.5599999999999999 the new Nf is 6.0
```

The importance of these results is the routing evolution. This is why these results are shown in

Route	P	Nf	H
1-2	0.7	3	2
1-4-2	**0.81**	**2**	**3**
Success in route 1-4-2			
1-2	0.7	3	2
1-4-2	**0.81**	**3**	**3**
Failure in route 1-4-2			
1-2	**0.7**	**3**	**2**
1-4-2	0.607	4	3
Success in route 1-2			
1-2	**0.7**	**4**	**2**
1-4-2	0.607	4	3
Failure in route 1-2			
1-2	0,55	5	2
1-4-2	**0.607**	**4**	**3**
Success in route 1-4-2			
1-2	0,55	5	2
1-4-2	**0.607**	**5**	**3**
Failure in route 1-4-2			

1-2	**0,55**	**5**	**2**
1-4-2	0.506	6	3

Table 9, were the chosen route is highlighted in bold letters.

Route	P	Nf	H
1-2	0.7	3	2
1-4-2	**0.81**	**2**	**3**
Success in route 1-4-2			
1-2	0.7	3	2
1-4-2	**0.81**	**3**	**3**
Failure in route 1-4-2			
1-2	**0.7**	**3**	**2**
1-4-2	0.607	4	3
Success in route 1-2			
1-2	**0.7**	**4**	**2**
1-4-2	0.607	4	3
Failure in route 1-2			
1-2	0,55	5	2
1-4-2	**0.607**	**4**	**3**
Success in route 1-4-2			
1-2	0,55	5	2
1-4-2	**0.607**	**5**	**3**
Failure in route 1-4-2			
1-2	**0,55**	**5**	**2**
1-4-2	0.506	6	3

Table 9. Routes adjustment, after feedback

- **Conclusions:**

With these results is possible to see that the route is being adjusted in a dynamic way after a failure or success occurs. So the next packets that need to use this link are going to travel for the route with better probability to obtain a success, reducing the amount of loss packets in the network.

III.1.1.1.6. Test #6: All the model with the Feedback Scheme

- **General Main:**

The general main is to see if all models work in a proper way, when the feedback scheme is incorporated to the previous model section.

- **Values Involved:**

The packets are working in the same unit that before, which is packets/u.t., and no new elements are introduced in this section. But a new method is introduced, which is called *sacar*, which is in the *WQpop* class, which is the one able to find the packet that needs retransmission and evaluates if the maximum number of retransmission expired or in other case, if the packet is able to be retransmitted. In this case, the packet will be extracted and sent to the route calculation, in order to get a new route with better probability to reach the destination. Then the packet will go to the *transmission queue* and compete with the other packets that are waiting for an idle output link.

- **Methodology:**

In this section, packets are generated and waiting for transmission, but after success a failure is generated in a random way, so the packet will need retransmission. This case will be performed until the packet expire its maximum feedback number and until it can be retransmitted.

With this test it will be possible to see how the packet goes to the *transmission queue* once it's popped from the *waiting queue* and a failure occurs.

- **Results:**

The obtain results are showing next:

```
A new electric packet arrives
The electric packet will be 1.0    2     1     aaaa  bbbb  cccc
      dddd
The selected route will be:
1-2   1.0   1.0   2.0
```

```
A success occurs in route: 1-2

A new electric packet arrives
The electric packet will be 2.0    3     1     aaaa  bbbb  cccc
      dddd
The selected route will be:
1-3   1.0   1.0   2.0
A success occurs in route: 1-3

A new electric packet arrives
The electric packet will be 3.0    4     1     aaaa  bbbb
The selected route will be:
1-4   1.0   1.0   2.0
A success occurs in route: 1-4

A new electric packet arrives
The electric packet will be 4.0    5     1     aaaa  bbbb  cccc
      dddd
The selected route will be:
1-3-5 1.0   1.0   3.0
A failure occurs in route: 1-3-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 0

A packet that needs retx arrives
The selected route will be:
1-4-11-5    0.99  1.0   4.0
A failure occurs in route: 1-4-11-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 1

A packet that needs retx arrives
The selected route will be:
1-3-5 0.5   2.0   3.0
A failure occurs in route: 1-3-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 2

A packet that needs retx arrives
The selected route will be:
1-4-11-5    0.495 2.0   4.0
A failure occurs in route: 1-4-11-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 3

A packet that needs retx arrives
The selected route will be:
1-3-5 0.3333333333333333     3.0   3.0
A failure occurs in route: 1-3-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 4

A packet that needs retx arrives
The selected route will be:
1-4-11-5    0.33  3.0   4.0
A failure occurs in route: 1-4-11-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 5

A packet that needs retx arrives
The selected route will be:
1-3-5 0.25  4.0   3.0
A failure occurs in route: 1-3-5
A failure occurs in packet 4.0 with SA 1 DA 5 CoS 3 and N 6
```

```
It’s not possible to send the packet, because the Nf>Nmax, it
belongs at CoS= 3

A new electric packet arrives
The electric packet will be 5.0    6     1     aaaa
The selected route will be:
1-2-7-6     1.0   1.0   4.0
A success occurs in route: 1-2-7-6
```

- **Conclusions:**

With these results is possible to see the retransmission behavior in the model when feedback is received. It’s also possible to see that the model is studying if the number of retransmissions hasn’t expired and if it’s able to retransmit it. Also it’s possible to see the routing table evolution after feedback is received.

III.1.1.1.7. Test #7: Infinite Loop test

- **General Main:**

The General main is to study if the model is able to work in a correct way after an infinite loop is performed.

- **Values Involved:**

The same than the ones used in test #6.

- **Methodology:**

A billion (1.000.000.000) of random packets are generated, were the 20% is from CoS1, 30% of CoS2 and 50% of CoS3. The destination node is random, as also the feedback that comes from the network. This traffic will be following all the model phases and I will evaluate the routing table evolution, and the number of losses packets in the network that couldn’t be retransmitted.

Also it's important to say that all the routes are initialize with a P=1 and Nf=0. These values are going to be evaluated at the end of the simulation to see how the table evolution is.

- **Results:**

After 18 hours of simulation and 56.876.884 packets sent to the network, the simulation was stopped in order to see how it was doing. The obtained results were the following:

```
There are 2843849 packets of CoS 1 dropped after exceeding the
maximum number of retransmissions.
There are 3791799 packets of CoS 2 dropped after exceeding the
maximum number of retransmissions.
There are 7583597 packets of CoS 3 dropped after exceeding the
maximum number of retransmissions.
```

And the resulting routing tables for reaching the other 13 nodes in the network (from node 2 to 14) from source node 1 are shown next:

```
The possible routes to get to the destination node 2 from the
source node 0, are:
Route         P                    Nf        H
1-2      7.499104535815914E-5  8294996.0   2.0
1-4-2    7.499098665929411E-5  1184500.0   3.0
The selected route will be:
1-2      7.499104535815914E-5  8294996.0   2.0

The possible routes to get to the destination node 3 from the
source node 0, are:
Route                   P              Nf        H
1-3         5.637136193179703E-5    7362203.0   2.0
1-4-11-5-3  5.6371336975134035E-5   2117293.0   5.0
The selected route will be:
1-3   5.637136193179703E-5    7362203.0   2.0

The possible routes to get to the destination node 4 from the
source node 0, are:
Route                   P              Nf        H
1-4         4.1801858020806004E-5   6079523.0   2.0
1-2-4       4.180186127681104E-5    3399975.0   3.0
The selected route will be:
1-2-4 4.180186127681104E-5    3399975.0   3.0

The possible routes to get to the destination node 5 from the
source node 0, are:
Route                   P              Nf        H
```

```
1-3-5       2.9455690096125736E-5   2578668.0   3.0
1-4-11-5    2.9455687104723174E-5   6900833.0   4.0
The selected route will be:
1-3-5 2.9455690096125736E-5   2578668.0   3.0
```

The possible routes to get to the **destination node 6** from the source node 0, are:

```
Route                   P              Nf        H
1-3-5-6     6.457915154671931E-5    1138871.0   4.0
1-2-7-6     6.457915500320636E-5    8340629.0   4.0
The selected route will be:
1-2-7-6     6.457915500320636E-5    8340629.0   4.0
```

The possible routes to get to the **destination node 7** from the source node 0, are:

```
Route                   P              Nf        H
1-2-7       5.274028311655753E-5    8016218.0   3.0
1-3-5-6-7   5.274025527296737E-5    1463282.0   5.0
The selected route will be:
1-2-7 5.274028311655753E-5    8016218.0   3.0
```

The possible routes to get to the **destination node 8** from the source node 0, are:

```
Route                        P              Nf        H
1-3-8             5.252114884005992E-5    7265643.0   3.0
1-2-7-10-9-8      5.252114941679983E-5    2213853.0   6.0
The selected route will be:
1-2-7-10-9-8      5.252114941679983E-5    2213853.0   6.0
```

The possible routes to get to the **destination node 9** from the source node 0, are:

```
Route                   P              Nf        H
1-3-8-9     3.113945383104975E-5    2035272.0   4.0
1-2-7-10-9  3.113946376051689E-5    7444230.0   5.0
The selected route will be:
1-2-7-10-9  3.1139467943550426E-5   7444230.0   5.0
```

The possible routes to get to the **destination node 10** from the source node 0, are:

```
Route                         P              Nf        H
1-3-8-9-10        7.17832219029852E-5     9027932.0   5.0
1-3-5-6-7-10      7.178322176320651E-5    451563.0    6.0
The selected route will be:
1-3-8-9-10  7.17832219029852E-5     9027932.0   5.0
```

The possible routes to get to the **destination node 11** from the source node 0, are:

```
Route                   P              Nf        H
1-4-11      5.993080842098082E-5    6221641.0   3.0
1-3-5-11    5.993079063784855E-5    3257854.0   4.0
The selected route will be:
1-4-11      5.993080842098082E-5    6221641.0   3.0
```

The possible routes to get to the **destination node 12** from the source node 0, are:

```
Route                   P              Nf        H
1-4-11-12   4.517502596938565E-5    6717279.0   4.0
1-3-5-11-12 4.5175015187883036E-5   2762220.0   5.0
The selected route will be:
1-4-11-12   4.517502596938565E-5    6717279.0   4.0

The possible routes to get to the destination node 13 from the
source node 0, are:
Route                   P              Nf        H
1-4-11-13   3.280196017817118E-5    2403421.0   4.0
1-3-5-11-13 3.280195962213575E-5    7076078.0   5.0
The selected route will be:
1-4-11-13   3.280196017817118E-5    2403421.0   4.0

The possible routes to get to the destination node 14 from the
source node 0, are:
Route                        P              Nf        H
1-3-8-14          6.382785693951887E-5    7603549.0   4.0
1-3-5-11-13-14    6.382784467594472E-5    1875950.0   6.0
The selected route will be:
1-3-8-14    6.382785693951887E-5    7603549.0   4.0
```

- **Conclusions:**

With these results is possible to see that the model is performed well after a lot of simulation time. But it was possible to see that the route priorities are obtaining numbers so small that it can have problems in bigger simulations or in real life. That is why a new strategy should be proposed in adjusting the routing tables after a period of time.

Also it was possible to see that the chosen route for transmission is always the one with the best priority, as it was expected.

In order to solve this problem, a new strategy was designed, which is to normalize the routes probabilities. All the routes priorities to go to the same destination node should add 1.

III.1.1.2. Network Simulation

In this section, the final proposed model will be simulated in the 14-nodes NSFNET network, as the one shown in Figure 23, were it's possible to see the representation in 2D of the network, with the American States that made part of it.

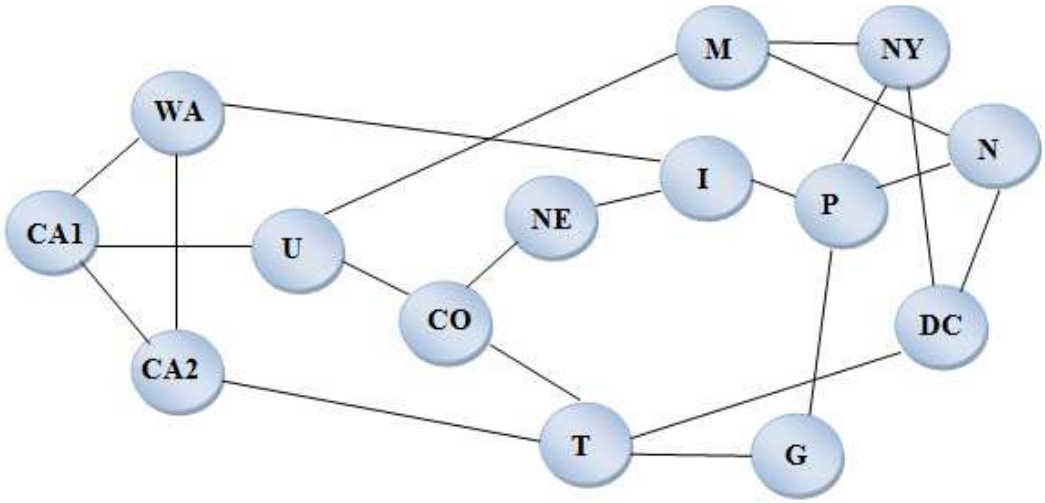

Figure 23. 2D Representation of the 14-Nodes NSFNET

I will choose these 14-nodes NFSNET, in which I will have 14 core nodes plus 14 edge routers, which means that for each core node I will have associated 1 edge routers. Each link will contain 4 wavelength channels at 10Gbps each. The number representation of this network used in this work is shown in Figure 24.

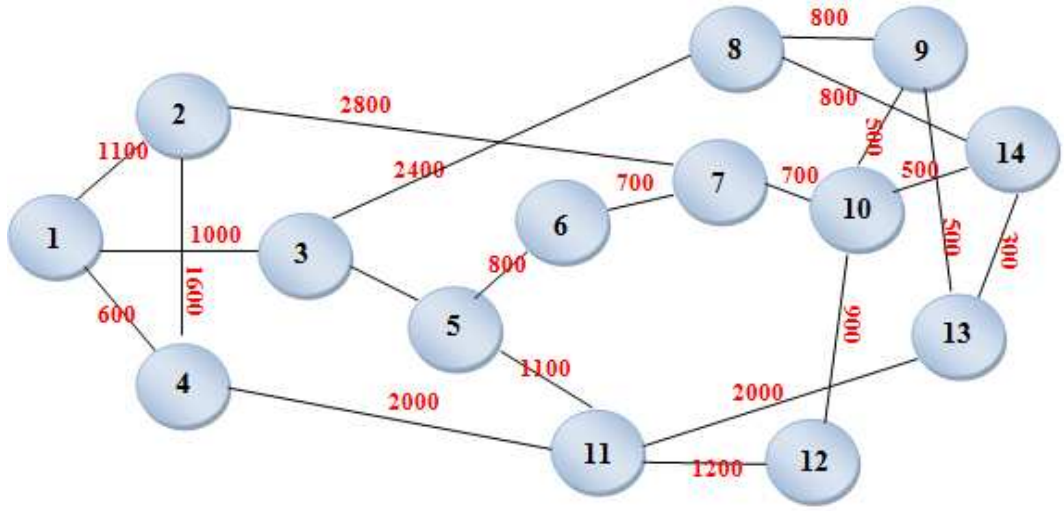

Figure 24. 14-Nodes NSFNET represented with numbers (units in Km).

With evaluation purposes, I simulated the 14-nodes NFSNET topology, assuming that each link carries 4 bidirectional wavelengths at 10Gbps. For the traffic characteristics, it's considered that 20% of the traffic is from CoS, 30% from CoS1 and 50% from CoS3.

For the traffic characteristics, I considered that uniformly distributed burst arrives following a Poisson process with rate 1/λ. Burst length packets have a length of 40kB, so the transmission time onto the link will be 32µs.

Regarding hardware devices, it's assumed that OBS nodes are equipped with full wavelength conversion; a non-blocking matrix and enough number of add/drop ports. The BCP processing time and the matrix switching time were set to 10µs and 2.5µs respectively.

These conditions were established with the purpose of obtaining comparative results with previous works. These characteristics are extracted from the paper "Controlled Loops: A New QoS Differentiation Frameworks for Delay-Sensitive OBS Networks" by Jordi Perelló, Salvatore Spadaro, Jaume Comellas and Gabriel Junyet (2008 IEEE).

III.1.1.2.1. Test #8: Proposed Model (RPBS) Performance in the Network

- **General Main:**

The idea of this test is to establish some networks parameters, to see the performance of the model in these conditions; which is the idea to evaluate other related works, to see if with the proposed model, the packets lost in the network are reduced.

- **Values Involved:**

The same than the ones used in test #6.

- **Methodology:**

The proposed model will be tested during ten million (10.000.000) iterations. The generated traffic is divided by classes giving 50% to CoS3, 30% to CoS2 and 20% to CoS1. The destination nodes (DN) for this traffic is independent

to the CoS and it's generated in a random mode, from DN2 to DN14, being the source edge node (SN) the SN1.

- **Results:**

From the obtained results, the important part to extract in order to compare, are shown next:

```
The amounts of new packets of CoS1 transmitted are 3135606
The amounts of new packets of CoS2 transmitted are 4509370
The amounts of new packets of CoS3 transmitted are 9555869

The amount of packets of CoS1 with N=1, were 11483 and with N=2
3206
The amount of packets of CoS2 with N=1, were 150561 with N=2 3246
with N=3 365 with N=4 4652
The amount of packets of CoS3 with N=1, were 31664 with N=2 6982
with N=3 439 with N=4 398 with N=5 67 with N=6 9966

In the model were 3206 CoS 1 packets that couldn't be transmitted
In the model were 4652 CoS 2 packets that couldn't be transmitted
In the model were 9966 CoS 3 packets that couldn't be transmitted
```

- **Conclusions:**

With these results, the number of packets transmitted for each CoS can be calculated, knowing that the generated traffic is 50% for CoS3, 30% for CoS2 and 20% for CoS1. In previous tests it was demonstrated that it can be generated with these characteristics. So having the amount of transmitted and lost packets in the network, the losses percentage can be calculated, to compare with the results in the following tests, which are done for comparing the two models.

III.1.1.2.2. Test #9: JET Performance

- **General Main:**

The idea of this test is to evaluate how many packets can be sent and how many will be lost if the feedback scheme isn't performed in order to compare with my proposal.

- **Values Involved:**

The same than the ones used in test #6.

- **Methodology:**

The same traffic, network and time conditions than in test #8 is generated in order to get the results to be compared. The idea is that in this test each time a link cannot be reserved, the burst will be lost because retransmission isn't possible. With these results I could compare the performance of my model and see if it brings improvements.

- **Results:**

As in test #9, these were extracted from the obtained results:

```
The amounts of new packets of CoS1 transmitted are 3539078
The amounts of new packets of CoS2 transmitted are 4718772
The amounts of new packets of CoS3 transmitted are 9437543
In the model were 48130 CoS 1 packets that couldn't be transmitted
In the model were 65591 CoS 2 packets that couldn't be transmitted
In the model were 133069 CoS 3 packets that couldn't be
transmitted
```

- **Conclusions:**

With these results, the number of packets transmitted for each CoS can be calculated. So having the amount of packets transmitted and lost packets in the network, the percentage of losses in this model without feedback can be calculated and compared with the results in tests #8 and #10.

III.1.1.2.3. Test #10: Performance with Feedback, but without Statistical route tables adjustment.

- **General Main:**

The general main in this study is to see the functionality of this model, which is a related work presented in the Literature Review, in the related works

section to see its performance and evaluate if with the new proposal some improvements are added.

- **Values Involved:**

  The same than the ones used in test #6.

- **Methodology:**

The same traffic, network and time conditions than in tests #8 and #9 is generated in order to get results that can be compared. This model will perform retransmissions with the same conditions that my model, which means, for CoS1 the Nmax=2, for CoS2 the Nmax=4 and finally, for CoS3 the Nmax=6. But the routes will be assigned without following a priority. These routes are assigned randomly for each packet.

- **Results:**

  The obtained results are the following:

```
The amounts of new packets of CoS1 transmitted are 2924581
The amounts of new packets of CoS2 transmitted are 4406535
The amounts of new packets of CoS3 transmitted are 9623945

The amount of packets of CoS1 with N=1, were 43038 and with N=2
29336
The amount of packets of CoS2 with N=1, were 42199 with N=2 34563
with N=3 9303 with N=4 44260
The amount of packets of CoS3 with N=1, were 122318 with N=2
100188 with N=3 10347 with N=4 17377 with N=5 29091 with N=6 96829

In the model 29336 CoS 1 packets couldn't be transmitted
In the model 44260 CoS 2 packets couldn't be transmitted
In the model 96829 CoS 3 packets couldn't be transmitted
```

- **Conclusions:**

With these results, the number of packets transmitted for each CoS can be calculated. So having the amount of transmitted and lost packets in the network, the percentage of losses in this model can be calculated and compared with the

results in tests #8 and #9. The analysis comparison of these three (3) models is shown in Chapter IV.

III.1.1.2.4. Test #11: Burst Loss Probability for different offered load

- **General Main:**

The general main is to study the performance of 3 algorithms, when they are tested at different traffic loads.

- **Values Involved:**

The offered load ρ, which is $\rho = \frac{\lambda}{\mu}$, were λ is the burst arrival rate, the parameter that is going to be modified to obtain the different ρ values, and 1/μ is the burst distribution, which was established before it will be 32μs always.

- **Methodology:**

The idea is to obtain the loss probability for different ρ values. So the algorithms are going to be tested for 10.000.000 iterations at different λ. Once the results are obtained, these are going to be compared with a previous work, to see if the results are similar and can be taken as reference.

- **Results:**

The obtained results can be seen in Table 10.

1/λ	1/ μ	Offered Load ρ	Losses RPBS	Losses WS	Losses JET
0,0001	0,000032	0,32	81	173	26710
0,00008	0,000032	0,40	981	2173	526710
0,000068	0,000032	0,47	2262	7588	1226479
0,000056	0,000032	0,57	3375	13985	1760717
0,000044	0,000032	0,73	4707	19971	1960618
0,000038	0,000032	0,84	5321	23403	2131831

Table 10. Obtained Results for different traffic loads

- **Conclusions:**

With these results is possible to see that for the RPBS model the losses are much smaller than for the JET model, and the model without statistical adjustment is in between, closer to RPBS. In Chapter IV these results are going to be analyzed.

III.1.1.2.5. Test #12: Three Models Performance, based on Network Failures

- **General Main:**

The general main is to see the functionality of these three models, when link failures appear in the Network with a random probability.

- **Values Involved:**

The same than the ones used in test #6.

- **Methodology:**

The same traffic conditions than in previous tests are generated in order to get results that can be compared. Each model will work as it was previously explained. But the network conditions are going to be modified, based in random link failures, giving different probabilities to each type of link. These failures are reported to the model and it will count how many packets could not be retransmitted after the simulation, as the same as its respective class.

- **Results:**

From the obtained results, the important part to extract in order to compare, will be the number of loss packets after the maximum number of retransmissions is exceeded in each class, as its shown next:

For RPBS:

```
The amount of new packets of CoS1 transmitted is 2179775
```

```
The amount of new packets of CoS2 transmitted is 2906368
The amount of new packets of CoS3 transmitted is 5812735
The amount of packets of CoS1 retransmitted is 956194
The amount of packets of CoS2 retransmitted is 1600456
The amount of packets of CoS3 retransmitted is 3750053
The amount of losses in the network is 496159
In the model were 135452 CoS 1 packets that couldn't be
transmitted
In the model were 143036 CoS 2 packets that couldn't be
transmitted
In the model were 217671 CoS 3 packets that couldn't be
transmitted
```

For the JET scheme:

```
The amount of new packets of CoS1 transmitted is 3540442
The amount of new packets of CoS2 transmitted is 4720591
The amount of new packets of CoS3 transmitted is 9441181
In the model were 1360961 CoS 1 packets that couldn't be
transmitted
In the model were 2813772 CoS 2 packets that couldn't be
transmitted
In the model were 3902977 CoS 3 packets that couldn't be
transmitted
The amount of losses in the network is 8077710
```

For the model with feedback but without statistical adjustment:

```
The amount of new packets of CoS1 transmitted is 1634766
The amount of new packets of CoS2 transmitted is 2179689
The amount of new packets of CoS3 transmitted is 4359379
The amount of packets of CoS1 retransmitted is 1289440
The amount of packets of CoS2 retransmitted is 2225183
The amount of packets of CoS3 retransmitted is 5265711
The amount of losses in the network is 741279
In the model were 204784 CoS 1 packets that couldn't be
transmitted
In the model were 216503 CoS 2 packets that couldn't be
transmitted
In the model were 319992 CoS 3 packets that couldn't be
transmitted
```

- **Conclusions:**

With these results, the number of packets transmitted for each CoS can be calculated. So having the amount of transmitted and lost packets in the network, the percentage of losses in this model can be calculated and see if the three models are working as previous. The analysis comparison of these three (3) models is shown in Chapter IV.

III.3. Results Analysis

In this project phase, the obtained results in the different tests applied to the new model and the analysis of them will be done in Chapter IV, which is the specific section were this point can be explained with details.

With the obtained results it was possible to see that the proposed model adds new improvements to related works using the feedback mechanism to adjust the routing table. With this use the burst must follow a statistical way to choose the routes in order to perform a better quality of service in optical networks.

III.4. Culmination Phase

This phase is set to define the closure of the final master work. It will carry out the conclusions and recommendations for the design and implementation of the suggested model for providing QoS in OBS/ IP networks in order to present the proposal and give a step forward in optical networks studies. The conclusions will be developed in Chapter VI while proposing recommendations to be considered for new studies and research in the same subject.

CHAPTER IV

Analysis Results

In this section an analysis of the results is done, based on the tests done in Chapter III, and the established mains in the beginning of the work, which is to obtain a new model that is able to provide QoS in IP/OBS networks.

The first analysis will be done for the tests #11, were the three models were tested at different traffic loads. In Table 11 it's shown the obtained results for each algorithm after 10.000.000 iterations, and in Figure 26 the performance can be seen in charts.

Offered Load ρ	Loss probability RPBS	Loss probability WS	Loss probability JET
0,32	8,10E-06	1,73E-05	2,67E-03
0,40	9,81E-05	2,17E-04	5,27E-02
0,47	2,26E-04	7,59E-04	1,23E-01
0,57	3,38E-04	1,40E-03	1,76E-01
0,73	4,71E-04	2,00E-03	1,96E-01
0,84	5,32E-04	2,34E-03	2,13E-01

Table 11. Loss probability for different traffic loads.

Based on these results and making a comparison with the paper "Controlled Loops: A New QoS Differentiation Framework for Delay-Sensitive OBS Networks", it's possible to say that the obtained results for the JET algorithm simulated in that paper, specifically shown in Fig. 3 of this paper, which is annexed at the end of this work, the standard JET was used for the BE class, and also shown in Figure 25, following the same performance and traffic conditions used in the simulation of this study; giving validation to their work and also to the obtained results for the three performances of this study. [35]

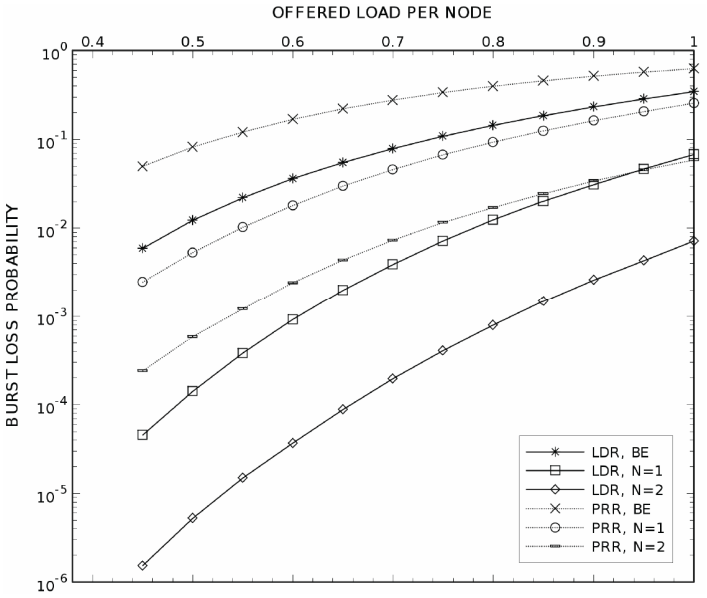

Figure 25. Reference of Offered load Vs. Burst Loss Probability. (Extracted from [35])

Also is possible to see that when the traffic load is small, the three algorithms converge towards the same loss probability, but as ρ increases, the loss probability for the three algorithms separates as RPBS always maintains the lower number of losses in the Network.

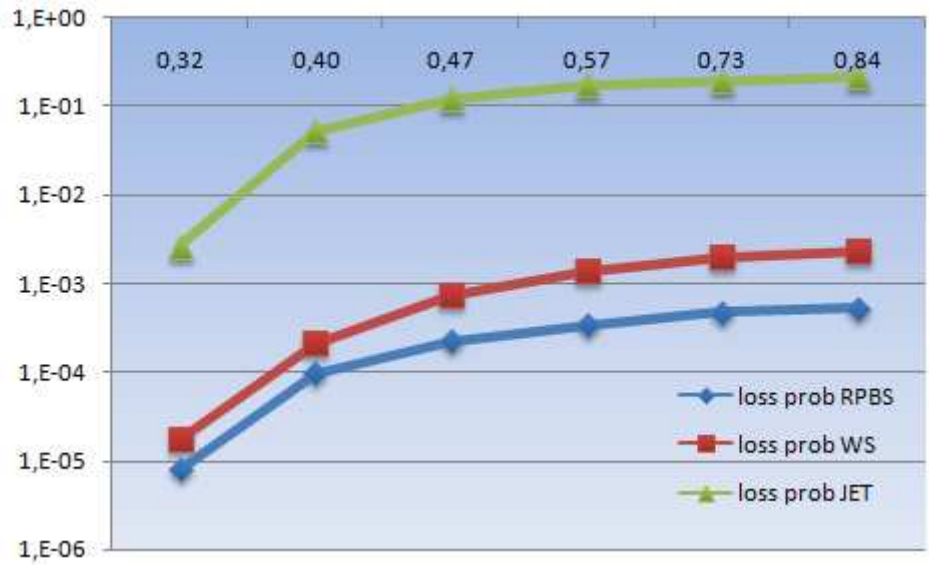

Figure 26. Burst loss probability in a 3-class scenario.

Once it was possible to verify that the obtained results in the simulation can be taken as valid. The comparison of the results can be done with the previous obtained results in the various tests.

The obtained results in tests #8, #9 and #10 are summarize and shown in Table 12, also in Figure 27, Figure 28, Figure 29 is possible to see the received & losses per algorithm.

	CoS1	CoS2	CoS3
Send RPBS	3135606	4509370	9555869
Received RPBS	3132400	4504718	9545903
Lost RPBS	3206	4652	9966
Send WS	2924581	4406535	9623945
Received WS	2895245	4362275	9527116
Lost WS	29336	44260	96829
Send JET	3539078	4718772	9437543
Received JET	3490948	4653181	9304474
Lost JET	48130	65591	133069

Table 12. Obtained Results in tests #8, #9 and #10

The received packets are all the packets that are able to reach the final destination, meanwhile the lost packets are the ones that are not able to reach the destination node. For RPBS and WS models the losses in the network are all the packets that could not reach the destination, so all the packets that are retransmitted are also taken as losses.

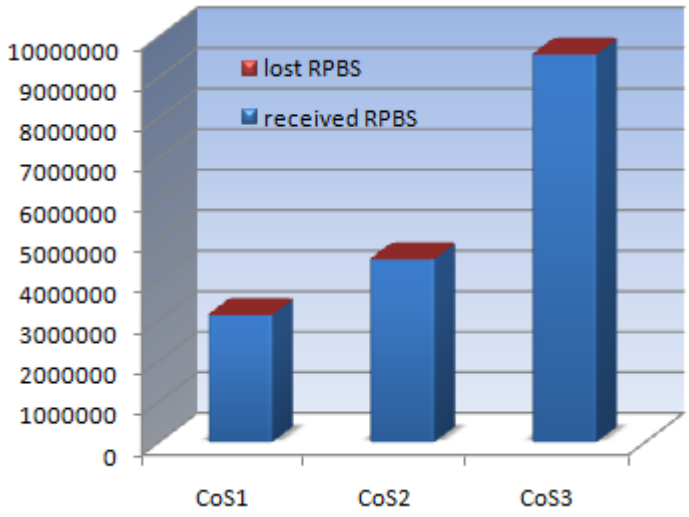

Figure 27. RPBS results, received & lost packets per each CoS.

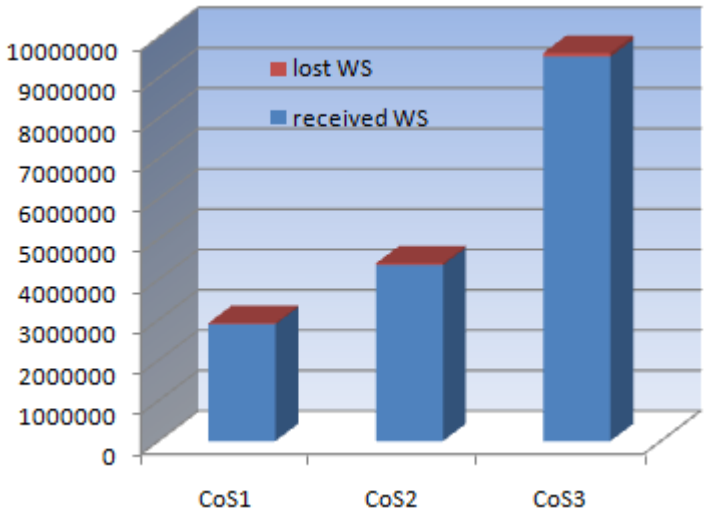

Figure 28. WS results, received & lost packets per each CoS.

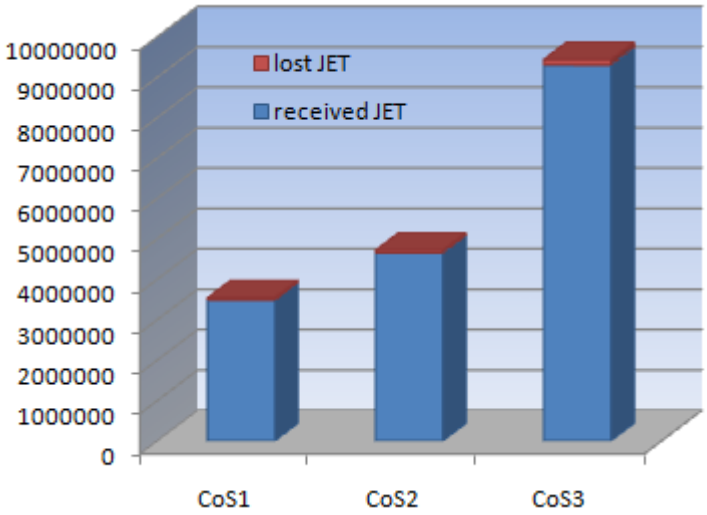

Figure 29. JET results, received & lost packets per each CoS.

In Figure 30, is possible to see the percentage of all the packets sent to the network per model and per class, were is possible to see that for CoS1 the amount of sent packets is 20%, for CoS2 30% and finally for CoS3 50%. And in Figure 31 it's possible to see the losses in packets units that models suffer per each CoS.

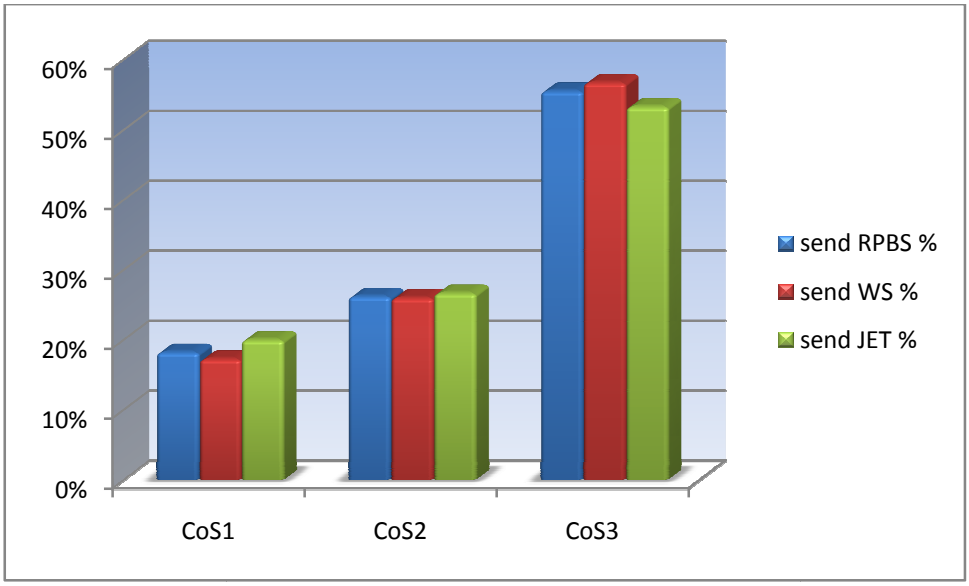

Figure 30. Percentage of the send packets to the network for the three models.

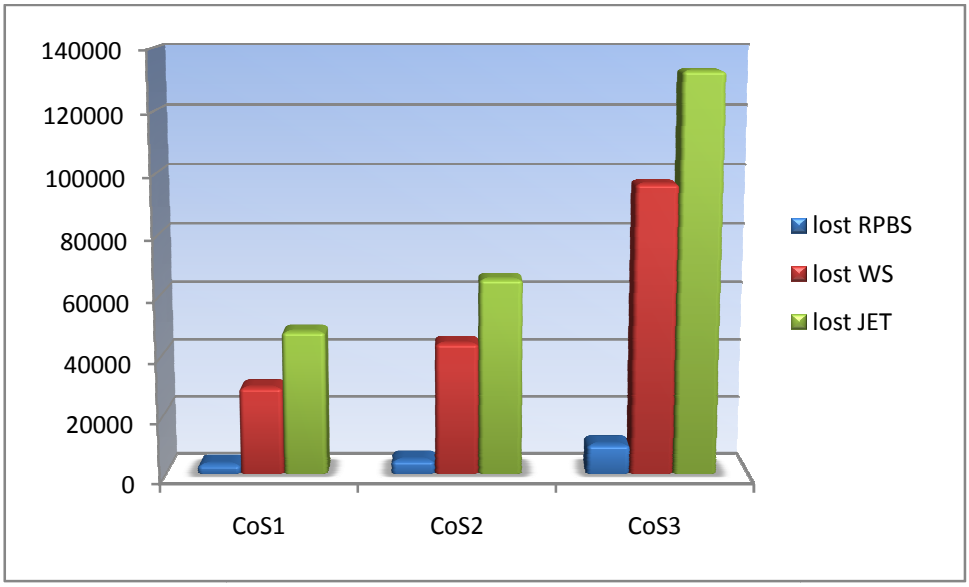

Figure 31. Losses in the Network per model.

The final and comparative results are shown in Table 13 and in Figure 32, were it's possible to see that RPBS improve the performance of the WS model, in terms of packets losses in the network.

Model	Send	Lost	Received	Loss Probability
RPBS	17200845	17824	17183021	1,04E-03
WS	16955061	170425	16784636	1,01E-02
JET	17695393	246790	17448603	1,39E-02

Table 13. General Performance in the Network

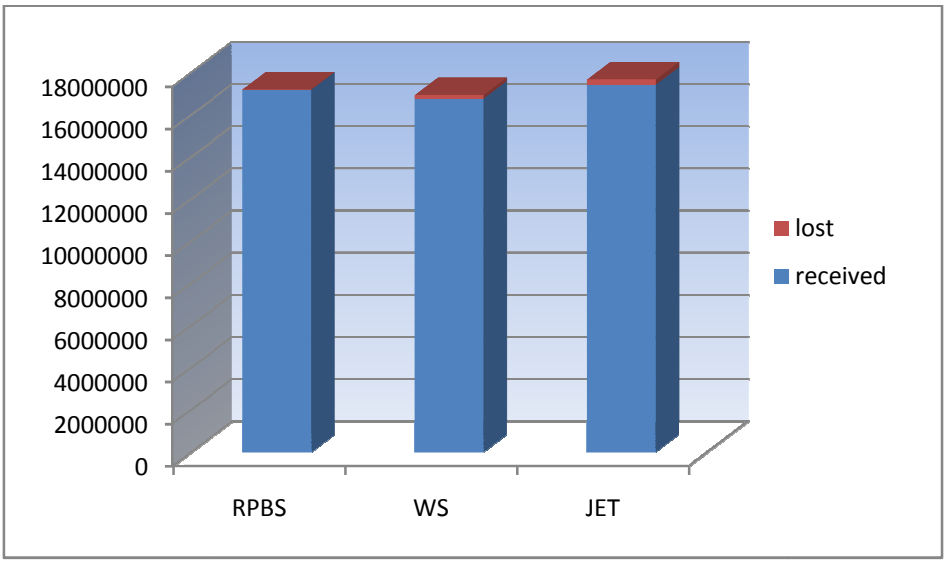

Figure 32. General Performance in the Network per Model.

Another important point to take in consideration in the model study is the route's evolution. This is because in this model the route was chosen by taking into consideration the statistical evolution of them. The routes are always normalized and the addition of their priorities is always one.

In this case, the evolution of three destinations is studied. These were chosen randomly, and the results were 7, 10 and 14. The evolution of the priorities of possible routes for reaching each destination is graphically seen in Figure 33, Figure 34 and Figure 35 respectively. Due to the programming software restrictions, only the results from the iteration 10883781 could be obtained, which represents the final stage of the routes.

With these graphics it’s possible to see the resultant priorities after each iteration of each possible route to reach a specific destination.

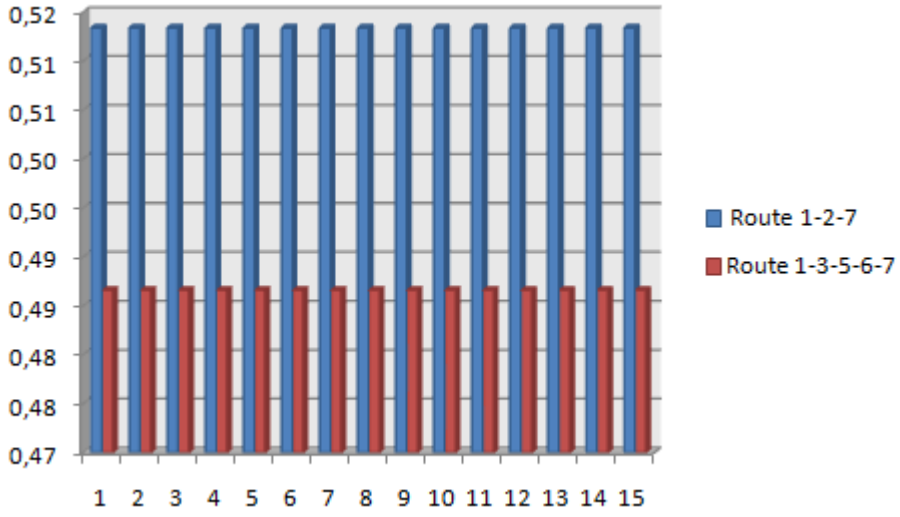

Figure 33. Routes evolution from node 1 to node 7. Expressed as Priority Vs. Iterations

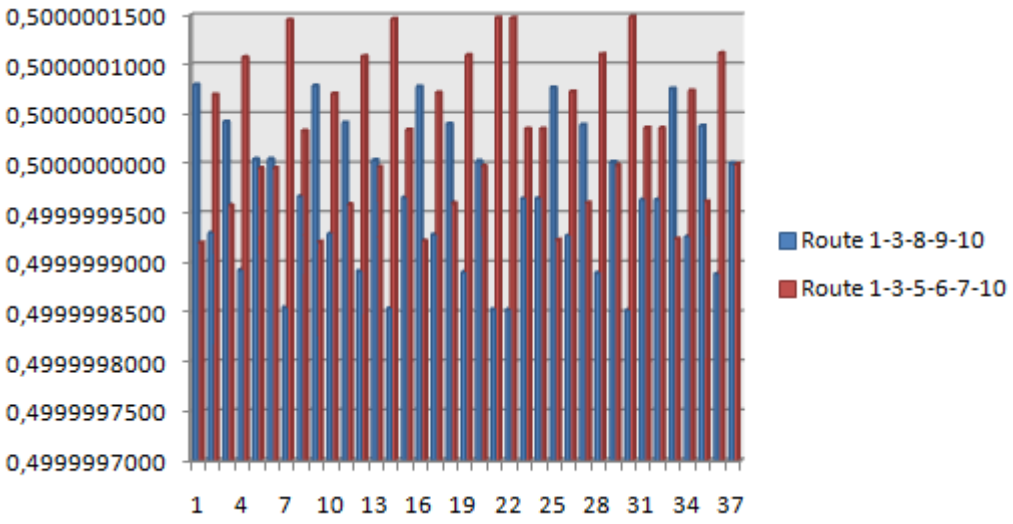

Figure 34. Routes evolution from node 1 to node 10. Expressed as Priority Vs. Iterations

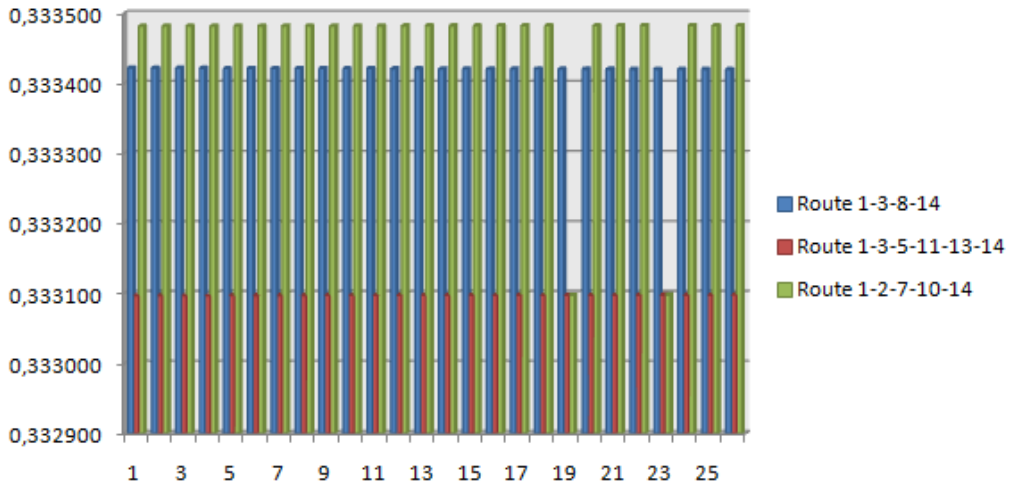

Figure 35. Routes evolution from node 1 to node 7. Expressed as Priority Vs. Iterations

With these results it's possible to see that for destination node 7, the route 1-2-7 prevails in the end, while the route 1-3-5-6-7 is discarded because it suffered much more losses at the beginning. Meanwhile for destination node 10 both routes are consistently selected, due to similarities of the failure and success probabilities along the simulation. Finally for destination node 14 the three routes at the end follow a constant behavior, were route 1-3-5-11-13-14 is almost discarded and the other two routes still have possibilities to get chosen.

In Table 14 is possible to see the number of feedbacks obtained per each route to reach each destination when all the simulations are performed.

To	Route #1	Nf	Route #2	Nf	Route #3	Nf
2	1-2	767923	1-4-2	189380		
3	1-3	808857	1-4-11-5-3	146752		
4	1-4	41763	1-2-4	915982		
5	1-3-5	251243	1-4-11-5	706090		
6	1-3-5-6	20	1-2-7-6	956118		
7	1-2-7	955860	1-3-5-6-7	5		
8	1-3-8	958219	1-2-7-10-9-8	82		
9	1-3-8-9	1392224	1-2-7-10-9	29778		
10	1-3-8-9-10	1670885	1-3-5-6-7-10	2232811		
11	1-4-11	957846	1-3-5-11	10		
12	1-4-11-12	440157	1-3-5-11-12	960486	1-2-7-10-12	960486
13	1-4-11-13	1118803	1-3-5-11-13	286849	1-2-7-10-9-13	286849
14	1-3-8-14	1386593	1-3-5-11-13-14	861	1-2-7-10-14	19689

Table 14. Final Results of the number of feedbacks per route to reach each destination

In this table is possible to see how some routes are practically discarded in comparison with the other possible routes to reach the same destination due to the constant failures in the beginning of the simulation. These results can be seen in a more graphical way in Figure 36, were the possible routes to get to each destination are described by Route #1, Route #2 and Route #3, following the same order than the previous table.

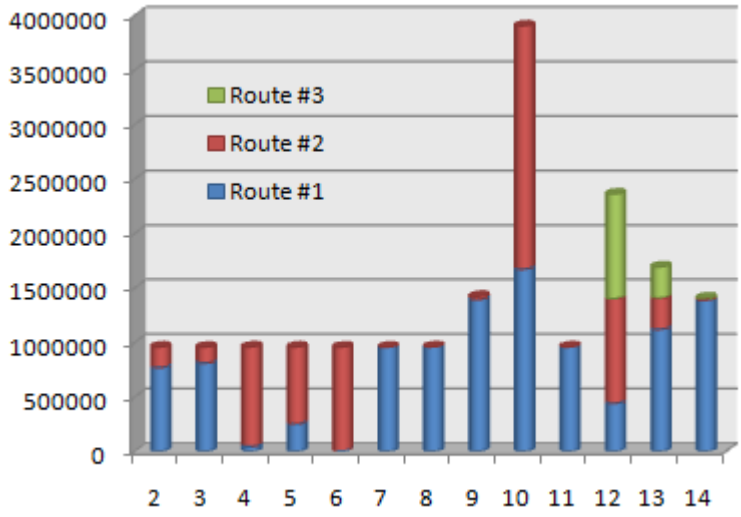

Figure 36. Number of Feedbacks per route vs. Possible Destination

In test #12 the models were evaluated in a network with link failures. After the execution, the models gave the results that can be seen in Table 12, were the amount of sent packets to the network, the finale received and the losses for the three models and the three CoS, are shown. Finally, in Figure 37 is possible to see the performance per CoS of RPBS model, in Figure 38 the performance of the second model based on feedback but without statistical routing and in Figure 39 is possible to see the typical JET performance.

Packets Status	CoS1	CoS2	CoS3
Send RPBS	3135969	4506824	9562788
Received RPBS	3000517	4363788	9345117
Lost RPBS	135452	143036	217671
Send WS	2924206	4404872	9625090
Received WS	2719422	4188369	9305098
Lost WS	204784	216503	319992
Send JET	3540442	4720591	9441181
Received JET	2179481	1906819	5538204
Lost JET	1360961	2813772	3902977

Table 15. Obtained Results in test #12

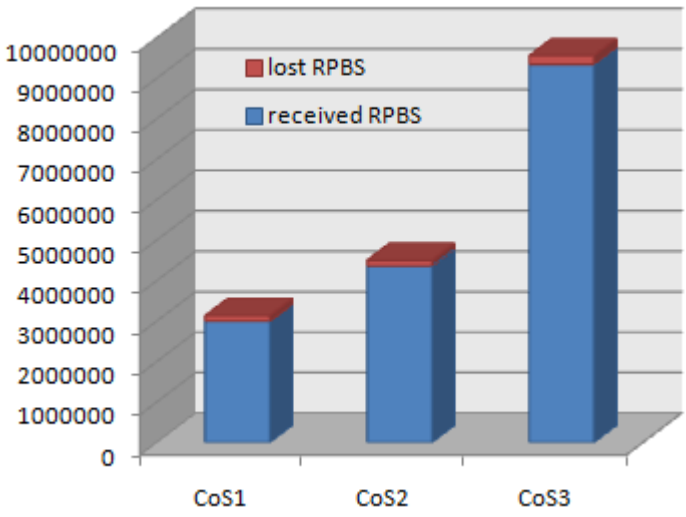

Figure 37. RPBS results with link failures: received & lost packets per each CoS.

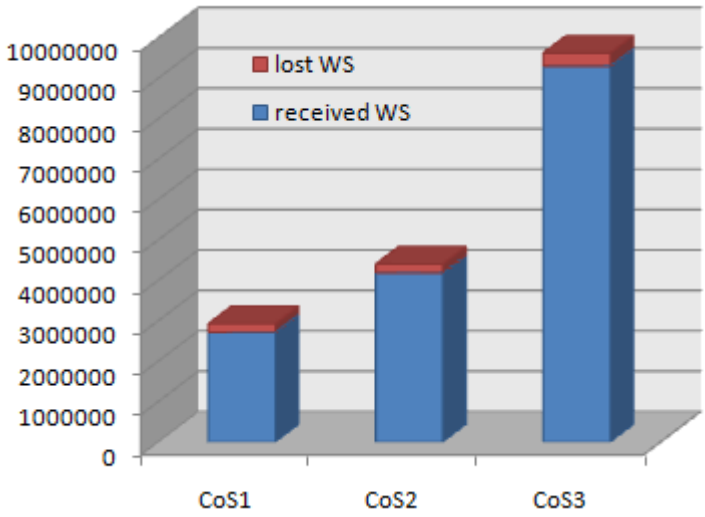

Figure 38. WS model results with link failures: received & lost packets per each CoS.

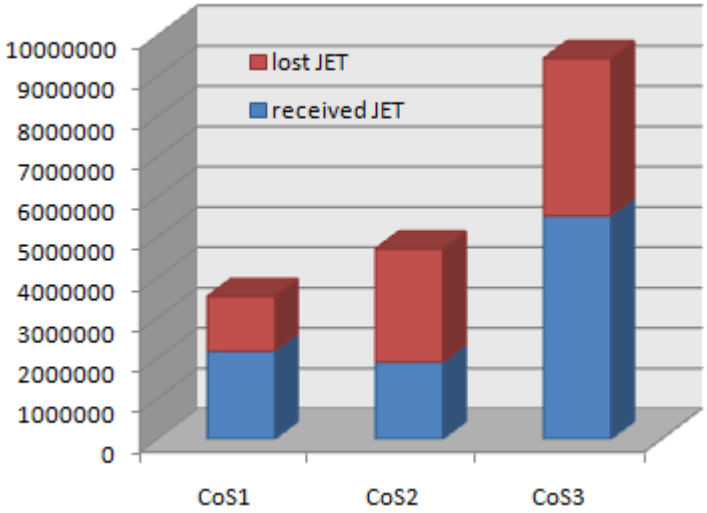

Figure 39. JET results with link failures: received & lost packets per each CoS.

With these results is possible to make a comparison between the models, as seen in Table 16; were the total amount of received, sent and losses per model is calculated as also the loss percentage each one. This can also be seen in Figure 40, were the general performance of the three models are shown, with its total amount of sent packets to the network, making a differentiation between the losses and the final received.

Model	Send	Received	Lost	Loss Probability
RPBS	17205581	16709422	496159	2,88E-02
WS	16954168	16212889	741279	4,37E-02
JET	17702214	9624504	8077710	4,56E-01

Table 16. General Performance in the Network with link failures

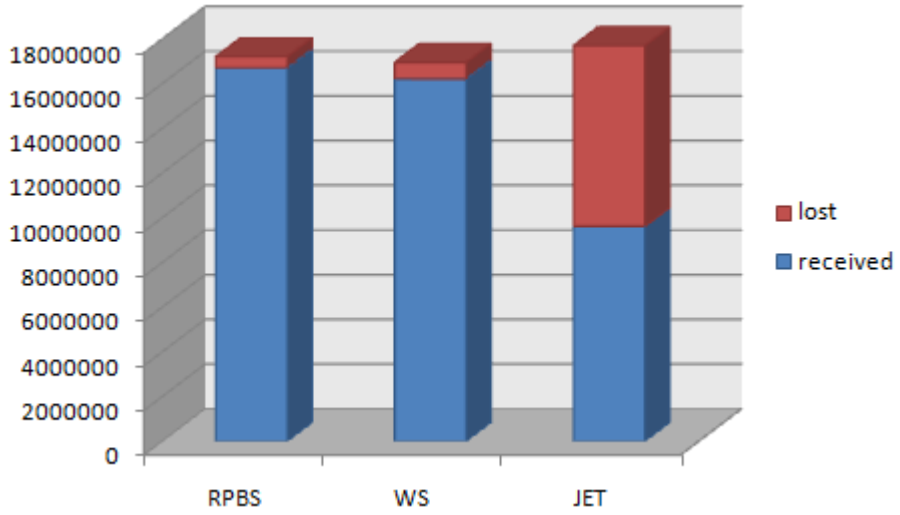

Figure 40. General Performance in the Network with link failures

With these results is possible to see that for RPBS models, the percentage of losses is still smaller than for the other two models, and also it proves that the feedback scheme performance of the typical JET algorithm improved in a big quantity as seen with model two and three.

CHAPTER V
Conclusions and Recommendations

Conclusions

Optical networks are currently the most studied and the most likely to be operated in the future to provide services on Internet based on OPS technology. But to achieve the migration to this network, which is currently very immature, OBS is being used. That's the reason why this area is been studied, in order to improve its functioning, which has big importance and it moves towards what awaits us in optical future. Such networks will allow users to have a higher transmission speed and IP service providers to have more confidence in their networks, having more subscribers to give a better quality of service, compared with the current systems.

Based on this approach, the general main of this research was based on the design of a new model for optical networks, based on OBS, which is able to improve the current quality of service conditions, which can serve as a basis for further studies and improvements in the optical networking systems.

Following, each of the obtained conclusions in the development Final Master Project, are described:

The proposed model was based in the JET routing technique, because all previous studies show it as the leader technique for providing absolute QoS in OBS networks, which gives a better possibility to allow isolation among CoS and gives different treatments to these classes, using an adequate offset time calculation. Based on this model, a new OT calculation was studied and applied in the model, obtaining positive results.

Once it was possible to obtain differentiation among classes, the biggest problem was to attack the losses in the system within these kinds of networks. This is due to control packets, which are not able to reserve the link for the burst

transmission. This was based on that a research of recent works was done and some models were found and used as based for the new model creation.

These related works opened a door, which is based on the feedback scheme, but using this scheme it's known that the losses were reduced in big amounts; but delays were introduced, so these two aspects were the key to attack in the new model creation.

In order to add improvements to these previous works, the idea of using statistical knowledge was introduced to choose the route to reach the destination. The model will not just perform the feedback scheme to see if the packet needs retransmission, but also it will use the knowledge of failures and successes to adjust the routing tables, giving a priority to each route, letting the system have statistical results of used routed.

After the model was designed, based on these parameters, a simulation of eighteen (18) hours was performed, which introduced an improvement to the previous work, which was the normalization of routes priorities. With this idea the model will have a better performance for a longer period of time.

Finally, once it was tested that the proposed model (RPBS) was able to work as expected for a long period of time; two new models were represented based on related studies. The first model was the typical JET algorithm which didn't use the feedback scheme and a second model (model WS), which was the later found study, which used the feedback scheme to see if a packet needs retransmission, giving different conditions for each CoS.

These three models were tested in different traffic loads, and it could be seen that with RPBS the losses in the network are reduced when the ρ increases, in comparison with the previous two models, being the crucial point $\rho=0,4$.

After the performances of the three models were tested and the expected results were obtained, the models were tested with the same network conditions

and then some failures were introduced to the network to see the performance of the algorithms.

These results give the information that using the feedback scheme the losses can be reduced in a big amount. With the proposed model (RPBS), the improvement could be seen in comparison with the model based on feedback in both aspects. First the losses were reduced almost to half and the delay was also reduced, which choosing a better route let the system have less losses and then less retransmission. So in the same period of time, more incoming packets can be sent to the network.

Finally, it is possible to say, that with the proposed model, the main objective of this study was achieved and it was able to find a strategy to provide better QoS in IP/OBS networks.

Recommendations

Here are some recommendations that emerge from the previous conclusions:

The system works better in small environments; it can present increase in losses when the network grows. In this case, it would be necessary to define autonomous systems, keeping the delays small and not causing problems in the development of certain applications.

That's why is recommended a strategy, were can be add functionality to internal nodes, were a better route to send the packets by themselves may be found, by using the feedback scheme. This way the source node should not be responsible of finding the whole route to reach the final destination.

For future works some parameters should be taken into account in order to improve them and to get better results. The parameters are:

- ✓ The priorities of the routes can be seen in more detail, starting with the equation that calculates the evolution of these priorities. In this study

the formula that was used, it's used very frequently in OBS networks based on feedback. But other equations could be taken into account, which could give a better performance in the future, as the normalization scheme that was used.

- ✓ Add more intelligence to the core nodes. This means that the core nodes could be able to find the following node in the route with better probability of success, using the feedback scheme, that way the edge source node wouldn't have the complete responsibility to find the best route. In this work, intelligence was only added to the edge nodes.

- ✓ Also, a protocol should be considered for assigning suitable routes automatically to reach each destination. In this study all the routes were established manually.

Other possible recommendations is to combine the previous model with the reflection routing strategy in order to get more efficiency in the system, which could lead to less losses.

List of Acronyms

OBS: *Optical Burst Switching*

WDM: *Wavelength Division Multiplexing*

DWDM: *Dense WDM.*

ASON: *automatically-switched optical network.*

IP: Internet Protocol

Diffserv: *Differentiated services*

SONET: *Synchronous Optical Network.*

SDH: *Synchronous digital hierarchy.*

ETSI: *European Telecommunications Standards Institute.*

ANSI: *American National Standards Institution.*

OCS: *Optical Circuit Switching.*

OPS*: Optical Packet Switching.*

AON: *All Optical Networks.*

BCP: *burst control packet*

BD: *burst payload or burst data*

CoS: *Class of Services*

QoS: *Quality of Services*

MPLS: *Multiprotocol Label Switching*.

RSVP-TE: *Resource Reservation Protocol with Traffic Engineering*

CR-LDP: *Constraint-Based Label Distribution Protocol.*

FDL: *Fiber Delay Line.*

FFUC: *First Fit Unscheduled Channel.*

LAUC: *Latest Available Unscheduled Channel*.

LAUC-VF: *LAUC with Void Filling.*

ITU-T: *International Telecommunication Union Telecommunication Sector Standardization.*

ATM: *Asynchronous Transfer Mode.*

ABT: *ATM block transfer*.

ABT/DT: *ABT with Delayed Transmission*.

ABT/IT: *ABT with Immediate transmission.*

TAG: *Tell-and-Go.*

TAW: *Tell-and-Wait.*

JIT: *Just-in-Time.*

JET: *Just-Enough-time.*

GMPLS: *generalized MPLS*

OT: *Offset Time*

OTD: *Offset Time Decision*

IOT: *Insufficient Offset Time*

BLR: *burst lost rate*.

LOBS: *labeled OBD.*

LSR: *label switched router.*

LSP: *labels witched path*

CBR: *Constraint Based Routing*

ER: *Explicit Routing.*

LSP: *light switching path*

EDF: *Earliest Deadline First.*

Bibliography

[1.] **Zerbas, G, Nejabati, R and Simeonidou, D.** *Design and Implementation of a Hybrid IP-OBS Ingress Node for the Future Optical Networks.* University of Essex, UK : IEE Photonics Professional Network.

[2.] **Sun, Yongmei, et al.** *Design and Implementation of an Optical Burst-Switched Network Testbed.* Tokyo : IEEE Optical Communications, November 2005.

[3.] **So, Won-Ho, Lee, Hae-Chong and Kim, Young-Chong.** *QoS Supporting Algorithms for Optical Internet Based on Optical Burst Switching.* Korea : IEEE, September 2002.

[4.] **Pedrola, Oscar, et al.** *Flexible Simulators for OBS Network Architectures.* s.l. : IEEE, 2008.

[5.] **Hesselbach, Xavier, Huerta, Mónica and Alvizu, Rodolfo.** *Strategy for IP traffic support with QoS in OBS networks.* s.l. : IEEE.

[6.] **Guasch, Daniel, et al.** *Construcción de un Entorno de Simulación para Redes Ópticas de Conmutación de Ráfagas.* Barcelona : IEEE.

[7.] **Cereijo, Ibán.** Simulación y Comparativa de Mecanismos de Conmutación en Redes Ópticas. Vigo : Universidad de Vigo, 2005.

[8.] **ALCATEL.** Alcatel: Optical Networks Tutorial. [Online] February 6, 1998. http://www.webproforum.com/alcatel/full.html.

[9.] **V. Vokkrane, Q. Zhang, J.P. Jue, B. Chen.** *Generalized burst assembly and scheduling techniques for QoS.* . s.l. : GLOBECOM ´02, 2002.

[10.] Networks Simulation. *Optical Burst Switching.* [Online] [Cited: October 3, 2008.] http://www.networks-simulation.com/OpNets/OpticalBurstSwitching.aspx.

[11.] **Vokkarane, Vinod M.** SciencDirect. *Optical Switching and Networking.* [Online] February 2007. [Cited: Octuber 3, 2008.] http://www.sciencedirect.com.

[12.] **Xu, Lisong, Perros, Harry G. and Rouskas, George.** *Techniques for Optical Packet Switching and Optical Burst Switching.* North Carolina : IEEE Communications., January 2001.

[13.] **Yoo, Myunsgsik, Qiao, Chumming and Dixit, Sudhir.** *QoS performance of optical burst sitching in IP over WDM networks.* s.l. : IEEE Journal on Selected Areas in Communication, Octuber 2000.

[14.] *IEEE Communicacion Magazine.* s.l. : Vol 38, N° 3, March 2000.

[15.] **Institution of Electrical Engineers.** *Synchronous Digital Hierarchy (SDH).* UK : IEE Electronics & Communication Engineering Journal, June 1994.

[16.] **Battestilli, Tzvetelin and Perros, Harry.** *An Introduction to optical Burst Switching.* s.l. : IEEE Communications Magazine, Agust 2003. N°8 .

[17.] **Turner, J.S.** *Terabit burst switching and High Speed Networks.* 1998 : s.n.

[18.] **Chen, Y. and Turner, J. S.** *WDM burst switching for petabit capacity routers.* s.l. : MILCOMM, 1999.

[19.] **Callegati, F., et al.** *Design issues of optical IP touters for internet backbone applications.* s.l. : IEEE Communicacion Magazine, December 1999. vol 37, N° 12.

[20.] **Hudek, G. C. and Muder, D.J.** *Signalling Analysis for a Multi-Switch All-Optical Network.* s.l. : IEEE ICC, 1995.

[21.] **Baldine, et al.** *Jumpstart: A just-in-time signaling architecture for WDM burst-switched networks.* s.l. : IEEE Communications Magazine 40, 2002.

[22.] **Ramaswami, R. and Sivarajan, K. C.** *Optical Networks.* 2002. Second Edition.

[23.] **Xu, L., Perros, H. and Rouskas, G.** *Techniques for optical packet switching and optical burst switching.* s.l. : IEEE Communicationes Magazine, January 2001. vol 3, N°1.

[24.] **Xiong, Y., Vndenhoute, M. and Cankaya, H. C.** *Control architecture in optical burst-switched WDM networks.* s.l. : IEEE journal on Selected Areas in Communications, Octuber 2000. Vol 18, N° 10.

[25.] **Vokkarane, Vinod M. and Jue, Jason P.** *Prioritized Burst Segmantaion and Composite Burst-Assembly Techniques for QoS Support in Optical Burst-Switched Networks.* s.l. : IEEE Journal on Selected Areas in Communications, September 2003. N° 7.

[26.] **Yang, M., et al.** *A QoS supporting scheduling algorithm for optical burst switching.* San Antonio, Texas : GLOBECOM , November 2001.

[27.] **Dolzer, K., et al.** *Evaluation of reservation mechanisms in optical burst switching.* s.l. : AEU International Journal of Electronics and Communicacions, January 2001. vol. 5, N°1.

[28.] **Qiao, Chunming.** *Labeled Optical Burst Switching for IP over WDM Integration.* s.l. : IEEE Communications Magazine, September 2000.

[29.] **Detti, Andrea and Listanti, Marco.** *Impact of segments agregation on TCP Reno flows in optical burst switched networks.* s.l. : IEEE INFOCOM, January 2001. N° 1.

[30.] **Dolzer, Klaus, et al.** *Evaluation of Reservation Mechanisms for Optical Burst Switching.* s.l. : AEU Int. J. Electronic Communication, 2001.

[31.] **Dolzer, K. and Payer, W.** *On aggregation strategies for multimedia traffic.* s.l. : PGTS 2000, September 2000.

[32.] **Yoo, M. and Qiao, C.** *A new Optical Burst Switching Protocol for Supporting Quality of Service.* s.l. : All Optical Communicatio System, 1998.

[33.] **T., Hiroki, K, Ohmae and Choi, Young-Bok.** *An Effective BECN/CRN Typed Deflection Routing for QoS Guaranteed Optical Burst Switching.* s.l. : IEEE GLOBECOM , 2003.

[34.] **Lee, SeoungYoung, Hwang, In-Yong and Park, Hong-Shik.** *A New Paradigm of Optical Burst Switching System for Lossless Transmission* . s.l. : IEEE, 2006.

[35.] **Perelló, Jordi, et al.** *Controlled Loops: A New QoS Differentiation Framework for Delay-Sensitive OBS Networks.* Barcelona : IEEE, 2008.

[36.] **Lui, Hongbo and Mouftah., Hussein T.** *A New Absolute QoS Differentiation Scheme Supporting Best-Effort Class in OBS Networks.* Canada : IEEE, 2007.

[37.] **Yang, Li and Rouskas, George N.** *Generalized Wavelength Sharing Policies for Absolute QoS Guarantees in OBS Networks.* s.l. : IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, April 2007. Vol. 25, N°. 4.

[38.] **Garg, Amit Kumar and Kaler, R S.** *Enhancing Bandwidth Utilization and QoS in Optical Burst Switched High-Speed Network* . s.l. : Brazilian Society of Electromagnetism-SBMag, April 2008.

[39.] **Belbekkouche, Abdeltouab and Hafid, Abdelhakim.** *An Adaptive Reinforcement Learning-based Approach to Reduce Blocking Probability in Bufferless OBS Networks* . s.l. : IEEE Communications Society , 2007 .

[40.] **Wu, Bing, et al.** *Providing absolute QoS in an OBS/WR architecture* . Singapore : JOURNAL OF OPTICAL NETWORKING, September 2008. Vol. 7, No. 9.

[42.] **Yu Ben, Qian Ying Tang.** Optical Packet Switching. [Online] May 2, 2006. [Cited: November 17, 2008.] http://inst.eecs.berkeley.edu/~ee233/sp06/student_presentations/EE233_Optical_paket_switching.pdf..

Annexes A

Source Code

In this section will be presented the main program source code as the fundamental classes used in this study, being all the source code used for the entire simulation presented in the CD annexed to this memory work.

The source code of the main program is shown next, which shows the simulation of the RPBS model, which was programmed in Java

```
public class ModelwithNet {
  public static void main(String args[]){

 packetelec pe1= new packetelec();
 packetelec pe2= new packetelec();
 counters count = new counters();
 packets pbcp1 = new packets();
 packets pa1 = new packets();
 ruta ru2 = new ruta();
 nack na1 = new nack();
 contpaksinenvio c1 = new contpaksinenvio();
 bancorutas br1 = new bancorutas();
 resultado res1= new resultado();
 bcpout paq = new bcpout();
 colaWQ cowq = new colaWQ();
 bcpTQ val = new bcpTQ();
 colaor cola1 = new colaor();
 packets2 pbcpr1 = new packets2();
 packetsbcp pbcptot = new packetsbcp ();
 packetsbcp pbcpout = new packetsbcp();
 countNACK cou = new countNACK();
 caudallink cl = new caudallink();
 contlink cl2 = new contlink();
 int perd = 0;
 int retrans = 0;
 int gene = 0;
 int No1=0;
 int N1=0;
 int N2=0;
 int N21=0;
 int N22=0;
 int N23=0;
 int N24=0;
 int N31=0;
 int N32=0;
 int N33=0;
 int N34=0;
 int N35=0;
 int N36=0;
 int cos=0;
 int env1=0;
 int env2=0;
 int env3=0;
 int env4= 0;
 int envn1=0;
 int envn2=0;
 int envn3=0;
 int c=0;
for(c=0; c<1000000000; c++){
 int N = pbcp1.getN();
 int Nmax = pbcp1.getNmax();
 int resul = na1.getres();
 countingNACK con = new countingNACK();
 cou = con.contar (cou);
   if (resul==1| N > Nmax){
         System.out.println(" ");
     System.out.println("A new electric packet arrives   ");
     gene = gene+1;
 contadores cont = new contadores();
```

```
count = cont.contarse (count);
traficoelec trf = new traficoelec();
pe2 = trf.trafic (pe1, count);
double clase= pe2.getc0se();
if (clase==1){
    envn1=envn1+1;
} else {
}if (clase==2){
    envn2=envn2+1;
}else {
}if (clase==3){
    envn3=envn3+1;
}
phase1 p1 = new phase1();
pbcp1 = p1.metodo1 (pe2);
} else {
       System.out.println(" ");
       System.out.println("A packet that needs retx arrives  ");
   retrans = retrans + 1;
   No1 = pbcp1.getN();
   double clase1= pbcp1.getc0s();
   if (clase1==1){
       env1=env1+1;
      if (No1==1){
       N1=N1+1;
      }
   } else {
   }if (clase1==2){
       env2=env2+1;
       if (No1==1){
           N21=N21+1;
          } else{}
       if (No1==2){
           N22=N22+1;
           N21=N21-1;
          }else{}
       if (No1==3){
           N23=N23+1;
           N22=N22-1;
          }
  }else {
   }if (clase1==3){
       env3=env3+1;
       if (No1==1){
           N31=N31+1;
          } else{}
       if (No1==2){
           N32=N32+1;
           N31=N31-1;
          }else{}
       if (No1==3){
           N33=N33+1;
           N32=N32-1;
          }else{}
       if (No1==4){
           N34=N34+1;
           N33=N33-1;
          }else{}
       if (No1==5){
           N35=N35+1;
           N34=N34-1;
          }
   }
}
route0 C = new route0();
ru2 = C.calculoruta (pbcp1, br1);
phase2 R = new phase2();
pbcpr1 = R.routefinder (pbcp1,ru2);
String rou = pbcpr1.getroute();
phase2b OTTL = new phase2b();
pbcptot = OTTL.offsetcal (pbcpr1);

 //aca mando a la transmission queue
```

```
   push add = new push();   // primer push
   cola1 = add.agregar (cola1, pbcptot);
   push gu = new push();
   val = gu.guardando(pbcptot,val);
   cola ord3 = new cola();
   cola1 = ord3.bubble(cola1);
   pop sac = new pop ();
   pbcpout = sac.sacar(cola1,val);
   WQpush alma = new WQpush();
   cowq = alma.almacenando(pbcpout, cowq);
   bcpcreation crear = new bcpcreation();
  paq = crear.crearbcps(pbcpout, cl2);

  pop borro = new pop();
  cola1 = borro.borrar(cola1);
  networksim creo = new networksim();
  na1 = creo.na1(paq, cou, cl); //, count);

  resul = na1.getres();
       switch (resul){
       case 1:  //sirve la ruta

                WQpop result1 = new WQpop();
                res1= result1.calculo(na1, cowq);

                ajusterutas ajus2 = new ajusterutas();
                br1 = ajus2.ajuste(br1, res1);

                ajusteP calp = new ajusteP();
                br1 = calp.verp(br1);

        break;
       case 0:  //falla la ruta
            WQpop saco = new WQpop();
            pbcp1= saco.sacar(na1, cowq, c1);


            WQpop result = new WQpop();
                res1= result.calculo(na1, cowq);

                perd = perd+1;

                ajusterutas ajus1 = new ajusterutas();
                br1 = ajus1.ajuste(br1, res1);

                ajusteP calp1 = new ajusteP();
                br1 = calp1.verp(br1);

            break;
       }

        double tmed=0.000056;  // valor de 1/lambda
        double t=0.0;
        t -= tmed* Math.log((100-Math.random()*100)/100);
        double taj = t*1000000;
        c = c+(int)taj;
  }

    int c0 = c1.getc1();
    int c2 = c1.getc2();
    int c3 = c1.getc3();
    N1= N1-c0;
    N2= c0;
    N23= N23-c2;
    N24=c2;
    N36 = c3;
    N35 = N35 - N36;
    System.out.println(" ");

    System.out.println("The amount of new packets of CoS1 transmitted are
"+envn1);
```

```
    System.out.println("The amount of new packets of CoS2 transmitted are
"+envn2);
    System.out.println("The amount of new packets of CoS3 transmitted are
"+envn3);

    System.out.println("The amount of packets of CoS1 transmitted are  "+env1);
    System.out.println("The amount of packets of CoS1 with N=1, were "+N1+ " and
with N=2 " +N2);
    System.out.println("The amount of packets of CoS2 transmitted are  "+env2);
    System.out.println("The amount of packets of CoS2 with N=1, were "+N21+ " with
N=2 " +N22+ " with N=3 " +N23+ " with N=4 " +N24);
    System.out.println("The amount of packets of CoS3 transmitted are  "+env3);
    System.out.println("The amount of packets of CoS3 with N=1, were "+N31+ " with
N=2 " +N32+ " with N=3 " +N33+ " with N=4 " +N34+ " with N=5 " +N35+ " with N=6 "
+N36);
    env4= c0+c2+c3;

    System.out.println(" ");
    System.out.println("The amount of losses in the network are  "+env4);

    System.out.println(" ");
    System.out.println("The amount of lost packtes in the network are "+ perd+ "
and " +retrans+ " where retransmitted, and new generate packets are "+gene);
    System.out.println(" ");
    System.out.println("In the model where "+ c0+ " CoS 1 packets that couldn't be
transmitted");
    System.out.println("In the model where "+ c2+ " CoS 2 packets that couldn't be
transmitted");
    System.out.println("In the model where "+ c3+ " CoS 3 packets that couldn't be
transmitted");
    System.out.println(" ");

  }
}
```

The source code of the different classes is shown next.

- Contadores Class:

```
public class contadores {


        public counters contarse (counters seq) {
               counters contadorseq = new counters();
               Double secuencia = seq.getseq();
               secuencia = secuencia + 1;
         contadorseq.setseq(secuencia);
         double destino = 2 + Math.random()*13;
         Integer destinoen = (int) destino;
         contadorseq.setDA(destinoen);
         Integer clase = seq.getc0s();
         if (secuencia%5==0){
                        clase=1;
                  }else{
                  }if ((secuencia%3==0)&&(secuencia%5!=0)){
                        clase=2;
                 }else {
                 }if ((secuencia%5!=0)&& (secuencia%3!=0)){
                        clase = 3;
        }
            contadorseq.setc0s(clase);
           return contadorseq;
    }
}
```

- Traficoelec Class:

```
public class traficoelec {
        // se genera el trafico de paquetes electronicos

       public packetelec trafic (packetelec pelec, counters secu) {
    packetelec pe1 = new packetelec();

    Integer contador1 = secu.getDA();
    Integer contadorSN = secu.getSA();
    Double contadorseq = secu.getseq();

        Integer b = secu.getc0s();

                    if (b==1) {
                      pe1.setDN(contador1);   // paquetes electronicos de CoS1
                pe1.setSN(contadorSN);
                       pe1.setc0se(new Integer(1));
                pe1.setdata0(new String("aaaa"));
                pe1.setNseq(contadorseq);
                  pe1.setdata1(" ");
                  pe1.setdata2(" ");
                  pe1.setdata3(" ");

                }
                    if (b==2) {
                for (int j=0; j<2; j++){
                  pe1.setDN(contador1);     // paquetes electronicos de CoS2
                  pe1.setSN(contadorSN);
                  pe1.setNseq(contadorseq);
                  pe1.setc0se(new Integer(2));
                   pe1.setdata0(new String("aaaa"));
                   pe1.setdata1(new String("bbbb"));
                   pe1.setdata2(" ");
                   pe1.setdata3(" ");
                }
              }
                    if (b==3) {
                       for (int j=0; j<3; j++){
                         pe1.setDN(contador1);     // paquetes electronicos de
CoS3
                          pe1.setc0se(new Integer(3));
                          pe1.setSN(contadorSN);
                          pe1.setNseq(contadorseq);
                          pe1.setdata0(new String("aaaa"));
                          pe1.setdata1(new String("bbbb"));
                          pe1.setdata2(new String("cccc"));
                          pe1.setdata3(new String("dddd"));
                       }
                   }; if (b < 3){
                        b=b+1;
                   }
                     else {
                        b=1;
                   }
                    double seq = pe1.getNseq();
                    Integer DA = pe1.getDN();
                    Integer SA = pe1.getSN();
                    String data0 = pe1.getdata0();
                    String data1 = pe1.getdata1();
                    String data2 = pe1.getdata2();
                    String data3 = pe1.getdata3();
                    seq = (int) seq;
                    System.out.println(" ");
                  System.out.println("The electric packet will be "+seq+"\t"+
DA+"\t"+ SA+"\t"+ data0+"\t"+ data1+"\t"+ data2+"\t"+ data3);
                  System.out.println(" ");
          return pe1;
      }
}
```

- Phase 1 Class:

```
public class phase1 {
 public packets metodo1 (packetelec pe1) {
   packets pbcp1 = new packets();
   Integer sourcebcp = pe1.getSN();
   Integer destinobcp = pe1.getDN();
   Integer claseser = pe1.getc0se();
   Double seq = pe1.getNseq();
   pbcp1.setDA(destinobcp);
   pbcp1.setSA(sourcebcp);
   pbcp1.setc0s(claseser);
   pbcp1.setNseq(seq);
  // System.out.println("The info to the bcp packet will be "+seq+"\t"+
destinobcp+"\t"+ sourcebcp+"\t"+ claseser);

   return pbcp1;
 }

 public burst1 metodo2 (burst1 bu1){
   packetelec pe1 = new packetelec();
   String dato0;
   Double seq = pe1.getNseq();
   dato0 = pe1.getdata0();
   bu1.setdata10(dato0);
   bu1.setNseq(seq);
System.out.println("The burst of class1 will be " +"\t"+dato0);
  return bu1;
  }

  public burst2 metodo3 (burst2 bu2){
   packetelec pe1 = new packetelec();
   String dato0, dato1;
   Double seq = pe1.getNseq();
   dato0 = pe1.getdata0();
   dato1 = pe1.getdata1();
   bu2.setdata20(dato0);
   bu2.setdata21(dato1);
   bu2.setNseq(seq);
System.out.println("The burst of class2 will be "+"\t"+dato0+"\t"+dato1);
  return bu2;
  }

 public burst3 metodo4 (burst3 bu3){
  packetelec pe1 = new packetelec();
  String dato0, dato1, dato2, dato3;
  dato0 = pe1.getdata0();
  dato1 = pe1.getdata1();
  dato2 = pe1.getdata2();
  dato3 = pe1.getdata3();
  Double seq = pe1.getNseq();
  bu3.setdata30(dato0);
  bu3.setdata31(dato1);
  bu3.setdata32(dato2);
  bu3.setdata33(dato3);
  bu3.setNseq(seq);
  Integer DA = pe1.getDN();
  Integer SA = pe1.getSN();
  Integer CoS = pe1.getc0se();
System.out.println("The electic packet will be "+seq+"\t"+  DA+"\t"+ SA+"\t"+
CoS+"\t"+dato0+"\t"+dato1+"\t"+dato2+"\t"+dato3);
 return bu3;
  }
       }
```

- Route0 Class:

```
public class route0 {
       public ruta calculoruta (packets p1, bancorutas rutas1) {

                 ruta ru1 = new ruta();

                 Integer destinobcp;
```

```
                    String route =  " ";
                    destinobcp = p1.getDA();
                    Integer DN = destinobcp;

                    double rou1;
                    double prio1;
                    double Nf1x;
                    double h1x;
                    double rou2;
                    double prio2;
                    double Nf2x;
                    double h2x;
                    double rou3;
                    double prio3;
                    double Nf3x;
                    double h3x;

                    switch (DN){
                    case 2:
                            rou1= rutas1.getroute31();
                            prio1 = rutas1.getP31();
                            Nf1x=  rutas1.getNf31();
                            h1x=  rutas1.geth31();

                            rou2= rutas1.getroute32();
                            prio2 = rutas1.getP32();
                            Nf2x=  rutas1.getNf32();
                            h2x=  rutas1.geth32();
                            double[][] matriz={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                                    System.out.println("The possible routes to
get to the destination node " +DN+ " from the source node 0, are:");
                                    System.out.print("Route"+"\t"+"\t"+"
P"+"\t"+" Nf"+"\t"+" H"+"\t");
                                    System.out.println("");
                        if (matriz[0][0]==31){
                            route="1-2";
                                System.out.print(route+"\t");
                            for (int j3=1; j3 < matriz[0].length; j3++) {
                                     System.out.print(matriz[0][j3]+"\t");
                             }
                                    System.out.println("");
                        } else {
                        }; if (matriz[1][0]==32){
                            route="1-4-2";
                                System.out.print(route+"\t");
                            for (int j3=1; j3 < matriz[1].length; j3++) {
                                      System.out.print(matriz[1][j3]+"\t");
                             }
                                System.out.println("");
                        }

                        Double route1=matriz[matriz.length-2][0];// ojo con esta
inicializacion!!!
                        Double P=matriz[matriz.length-2][1];  // estar pendiente que
si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                        Double Nf=matriz[matriz.length-2][2];
                        Double h=matriz[matriz.length-2][3];

                        int i1, j=1;
                            for(i1=0; i1<(matriz.length-1); i1++){
                                  if (matriz[i1][j] > matriz[i1+1][j]){
                                 route1 = matriz[i1][j-1];
                                   P = matriz[i1][j];
                                 Nf = matriz[i1][j+1];
                                 h = matriz[i1][j+2];
                                }
                             }

                            if (route1==31){
                                    route="1-2";
                                    System.out.print("The selected route will be: ");
```

```
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P+"\t"+ Nf+"\t"+
h);
                                    System.out.println("");
                                } else {
                                }; if (route1==32){
                                    route="1-4-2";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P+"\t"+ Nf+"\t"+
h);
                                    System.out.println("");
                                }
                                   ru1.setrouteas(route);  // asigno los valores a la
ruta
                                          ru1.setPas(P);
                                          ru1.setNfas(Nf);
                                          ru1.sethas(h);
                                          ru1.setruta(route1);
                                break;
                        case 3:
                                rou1= rutas1.getroute1();
                                prio1 = rutas1.getP1();
                                Nf1x=  rutas1.getNf1();
                                h1x=  rutas1.geth1();

                                rou2= rutas1.getroute2();
                                prio2 = rutas1.getP2();
                                Nf2x=  rutas1.getNf2();
                                h2x=  rutas1.geth2();
                                double[][] matriz3={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                            System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                             System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                                   System.out.println("");
                            if (matriz3[0][0]==1){
                                route="1-3";
                                 System.out.print(route+"\t");
                                for (int j3=1; j3 < matriz3[0].length; j3++) {
                                  System.out.print(matriz3[0][j3]+"\t");
                                 }
                                System.out.println("");
                            } else {
                            }; if (matriz3[1][0]==2){
                                route="1-4-11-5-3";
                                System.out.print(route+"\t");
                                for (int j3=1; j3 < matriz3[1].length; j3++) {
                                  System.out.print(matriz3[1][j3]+"\t");
                                 }
                               System.out.println("");
                            }

                            Double route3=matriz3[matriz3.length-2][0];// ojo con esta
inicializacion!!!
                            Double P3=matriz3[matriz3.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                            Double Nf3=matriz3[matriz3.length-2][2];
                            Double h3=matriz3[matriz3.length-2][3];

                            int i3, j2=1;
                                for(i3=0; i3<(matriz3.length-1); i3++){
                                       if (matriz3[i3][j2] > matriz3[i3+1][j2]){
                                     route3 = matriz3[i3][j2-1];
                                       P3 = matriz3[i3][j2];
                                     Nf3 = matriz3[i3][j2+1];
                                     h3 = matriz3[i3][j2+2];
                                    }
                                 }

                                if (route3==1){
```

```
                                    route="1-3";
                                      System.out.print("The selected route will be:
");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P3+"\t"+
Nf3+"\t"+ h3);
                                    System.out.println("");
                                } else {
                                }; if (route3==2){
                                    route="1-4-11-5-3";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P3+"\t"+
Nf3+"\t"+ h3);
                                    System.out.println("");
                                }
                                   ru1.setrouteas(route);  // asigno los valores a la
ruta
                                         ru1.setPas(P3);
                                         ru1.setNfas(Nf3);
                                         ru1.sethas(h3);
                                        ru1.setruta(route3);
                                break;
                        case 4:
                                rou1= rutas1.getroute3();
                                prio1 = rutas1.getP3();
                                Nf1x=  rutas1.getNf3();
                                h1x=  rutas1.geth3();

                                rou2= rutas1.getroute4();
                                prio2 = rutas1.getP4();
                                Nf2x=  rutas1.getNf4();
                                h2x=  rutas1.geth4();
                                double[][] matriz1={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                            System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                            System.out.println("");
                            if (matriz1[0][0]==3){
                                route="1-4";
                                  System.out.print(route+"\t");
                                for (int j3=1; j3 < matriz1[0].length; j3++) {
                                     System.out.print(matriz1[0][j3]+"\t");
                                 }
                                System.out.println("");
                            } else {
                            }; if (matriz1[1][0]==4){
                                route="1-2-4";
                                System.out.print(route+"\t");
                                for (int j3=1; j3 < matriz1[1].length; j3++) {
                                     System.out.print(matriz1[1][j3]+"\t");
                                 }
                                System.out.println("");
                            }

                            Double route2=matriz1[matriz1.length-2][0]; // ojo con esta
inicializacion!!!
                            Double P2=matriz1[matriz1.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                            Double Nf2=matriz1[matriz1.length-2][2];
                            Double h2=matriz1[matriz1.length-2][3];

                            int i2, j4=1;
                                for(i2=0; i2<(matriz1.length-1); i2++){
                                       if (matriz1[i2][j4]>matriz1[i2+1][j4]){
                                     route2 = matriz1[i2][j4-1];
                                        P2 = matriz1[i2][j4];
                                     Nf2 = matriz1[i2][j4+1];
                                     h2 = matriz1[i2][j4+2];
                                    }
```

```
                        }

                       if (route2==3){
                               route="1-4";
                               System.out.print("The selected route will be: ");
                               System.out.println("");
                               System.out.println(route+"\t"+   P2+"\t"+
Nf2+"\t"+ h2);
                               System.out.println("");
                           } else {
                           }; if (route2==4){
                               route="1-2-4";
                               System.out.print("The selected route will be: ");
                               System.out.println("");
                               System.out.println(route+"\t"+   P2+"\t"+
Nf2+"\t"+ h2);
                               System.out.println("");
                           }
                               ru1.setrouteas(route);  // asigno los valores a
la ruta
                                         ru1.setPas(P2);
                                         ru1.setNfas(Nf2);
                                         ru1.sethas(h2);
                                         ru1.setruta(route2);
                            break;

                   case 5:
                       rou1= rutas1.getroute5();
                           prio1 = rutas1.getP5();
                           Nf1x=  rutas1.getNf5();
                           h1x=  rutas1.geth5();

                           rou2= rutas1.getroute6();
                           prio2 = rutas1.getP6();
                           Nf2x=  rutas1.getNf6();
                           h2x=  rutas1.geth6();
                           double[][] matriz2={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};
                          System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                           System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                           System.out.println("");
                          if (matriz2[0][0]==5){
                              route="1-3-5";
                                System.out.print(route+"\t");
                              for (int j3=1; j3 < matriz2[0].length; j3++) {
                                   System.out.print(matriz2[0][j3]+"\t");
                               }
                              System.out.println("");
                          } else {
                          }; if (matriz2[1][0]==6){
                              route="1-4-11-5";
                              System.out.print(route+"\t");
                              for (int j3=1; j3 < matriz2[1].length; j3++) {
                                   System.out.print(matriz2[1][j3]+"\t");
                               }
                              System.out.println("");
                          }

                          Double route4=matriz2[matriz2.length-2][0]; // ojo con esta
inicializacion!!!
                          Double P4=matriz2[matriz2.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                          Double Nf4=matriz2[matriz2.length-2][2];
                          Double h4=matriz2[matriz2.length-2][3];

                          int i4, j3=1;
                              for(i4=0; i4<(matriz2.length-1); i4++){
                                     if (matriz2[i4][j3] > matriz2[i4+1][j3]){
                                   route4 = matriz2[i4][j3-1];
                                     P4 = matriz2[i4][j3];
                                   Nf4 = matriz2[i4][j3+1];
```

```
                                h4 = matriz2[i4][j3+2];
                               }
                             }

                            if (route4==5){
                                route="1-3-5";
                                System.out.println("The selected route will be:
");

                                System.out.println("");
                                System.out.println(route+"\t"+   P4+"\t"+
Nf4+"\t"+ h4);

                                System.out.println("");
                            } else {
                            }; if (route4==6){
                                route="1-4-11-5";
                                System.out.print("The selected route will be: ");
                                System.out.println("");
                                System.out.println(route+"\t"+   P4+"\t"+
Nf4+"\t"+ h4);

                                System.out.println("");
                            }
                                ru1.setrouteas(route);  // asigno los valores a
la ruta

                                          ru1.setPas(P4);
                                          ru1.setNfas(Nf4);
                                          ru1.sethas(h4);
                                          ru1.setruta(route4);
                             break;
                    case 6:

                        rou1= rutas1.getroute7();
                            prio1 = rutas1.getP7();
                            Nf1x=  rutas1.getNf7();
                            h1x=  rutas1.geth7();

                            rou2= rutas1.getroute8();
                            prio2 = rutas1.getP8();
                            Nf2x=  rutas1.getNf8();
                            h2x=  rutas1.geth8();
                            double[][] matriz6={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                           System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                           System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                           System.out.println("");
                          if (matriz6[0][0]==7){
                              route="1-3-5-6";
                                System.out.print(route+"\t");
                              for (int j6=1; j6 < matriz6[0].length; j6++) {
                                   System.out.print(matriz6[0][j6]+"\t");
                               }
                              System.out.println("");
                          } else {
                          }; if (matriz6[1][0]==8){
                              route="1-2-7-6";
                              System.out.print(route+"\t");
                              for (int j6=1; j6 < matriz6[1].length; j6++) {
                                        System.out.print(matriz6[1][j6]+"\t");
                               }
                              System.out.println("");
                          }

                          Double route6=matriz6[matriz6.length-2][0]; // ojo con esta
inicializacion!!!
                          Double P6=matriz6[matriz6.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                          Double Nf6=matriz6[matriz6.length-2][2];
                          Double h6=matriz6[matriz6.length-2][3];

                          int i6, j6=1;
                              for(i6=0; i6<(matriz6.length-1); i6++){
```

```
                                if (matriz6[i6][j6] > matriz6[i6+1][j6]){
                               route6 = matriz6[i6][j6-1];
                                  P6 = matriz6[i6][j6];
                               Nf6 = matriz6[i6][j6+1];
                               h6 = matriz6[i6][j6+2];
                              }
                             }

                            if (route6==7){
                                    route="1-3-5-6";
                                    System.out.println("The selected route will be:
");

                                    System.out.println("");
                                    System.out.println(route+"\t"+   P6+"\t"+
Nf6+"\t"+ h6);

                                    System.out.println("");
                                } else {
                                }; if (route6==8){
                                    route="1-2-7-6";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P6+"\t"+
Nf6+"\t"+ h6);

                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta
                                              ru1.setPas(P6);
                                              ru1.setNfas(Nf6);
                                              ru1.sethas(h6);
                                              ru1.setruta(route6);
                                 break;
                        case 7:
                            rou1= rutas1.getroute9();
                                prio1 = rutas1.getP9();
                                Nf1x=  rutas1.getNf9();
                                h1x=  rutas1.geth9();

                                rou2= rutas1.getroute10();
                                prio2 = rutas1.getP10();
                                Nf2x=  rutas1.getNf10();
                                h2x=  rutas1.geth10();
                                double[][] matriz7={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                               System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                                System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                               System.out.println("");
                              if (matriz7[0][0]==9){
                                  route="1-2-7";
                                    System.out.print(route+"\t");
                                  for (int j7=1; j7 < matriz7[0].length; j7++) {
                                       System.out.print(matriz7[0][j7]+"\t");
                                   }
                                  System.out.println("");
                              } else {
                              }; if (matriz7[1][0]==10){
                                  route="1-3-5-6-7";
                                  System.out.print(route+"\t");
                                  for (int j7=1; j7 < matriz7[1].length; j7++) {
                                       System.out.print(matriz7[1][j7]+"\t");
                                   }
                                  System.out.println("");
                              }

                              Double route7=matriz7[matriz7.length-2][0]; // ojo con esta
inicializacion!!!
                              Double P7=matriz7[matriz7.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                              Double Nf7=matriz7[matriz7.length-2][2];
                              Double h7=matriz7[matriz7.length-2][3];
```

```
                        int i7, j7=1;
                            for(i7=0; i7<(matriz7.length-1); i7++){
                                  if (matriz7[i7][j7] > matriz7[i7+1][j7]){
                                   route7 = matriz7[i7][j7-1];
                                     P7 = matriz7[i7][j7];
                                   Nf7 = matriz7[i7][j7+1];
                                   h7 = matriz7[i7][j7+2];
                                  }
                             }

                            if (route7==9){
                                    route="1-2-7";
                                    System.out.println("The selected route will be:
");

                                    System.out.println("");
                                    System.out.println(route+"\t"+   P7+"\t"+
Nf7+"\t"+ h7);
                                    System.out.println("");
                                } else {
                                }; if (route7==10){
                                    route="1-3-5-6-7";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P7+"\t"+
Nf7+"\t"+ h7);
                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta
                                              ru1.setPas(P7);
                                              ru1.setNfas(Nf7);
                                              ru1.sethas(h7);
                                              ru1.setruta(route7);
                                     break;
                    case 8:
                        rou1= rutas1.getroute11();
                            prio1 = rutas1.getP11();
                            Nf1x=  rutas1.getNf11();
                            h1x=  rutas1.geth11();

                            rou2= rutas1.getroute12();
                            prio2 = rutas1.getP12();
                            Nf2x=  rutas1.getNf12();
                            h2x=  rutas1.geth12();
                            double[][] matriz8={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                         System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");

                            System.out.println("");
                           if (matriz8[0][0]==11){
                               route="1-3-8";
                                 System.out.print(route+"\t");
                               for (int j8=1; j8 < matriz8[0].length; j8++) {
                                    System.out.print(matriz8[0][j8]+"\t");
                                }
                               System.out.println("");
                           } else {
                           }; if (matriz8[1][0]==12){
                               route="1-2-7-10-9-8";
                               System.out.print(route+"\t");
                               for (int j8=1; j8 < matriz8[1].length; j8++) {
                                       System.out.print(matriz8[1][j8]+"\t");
                                }
                               System.out.println("");
                           }

                           Double route8=matriz8[matriz8.length-2][0]; // ojo con esta
inicializacion!!!
```

```
                        Double P8=matriz8[matriz8.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                        Double Nf8=matriz8[matriz8.length-2][2];
                        Double h8=matriz8[matriz8.length-2][3];

                        int i8, j8=1;
                            for(i8=0; i8<(matriz8.length-1); i8++){
                                   if (matriz8[i8][j8] > matriz8[i8+1][j8]){
                                    route8 = matriz8[i8][j8-1];
                                       P8 = matriz8[i8][j8];
                                    Nf8 = matriz8[i8][j8+1];
                                    h8 = matriz8[i8][j8+2];
                                   }
                                }

                            if (route8==11){
                                    route="1-3-8";
                                    System.out.println("The selected route will be:
");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P8+"\t"+
Nf8+"\t"+ h8);
                                    System.out.println("");
                                } else {
                                }; if (route8==12){
                                    route="1-2-7-10-9-8";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P8+"\t"+
Nf8+"\t"+ h8);
                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta
                                              ru1.setPas(P8);
                                              ru1.setNfas(Nf8);
                                              ru1.sethas(h8);
                                              ru1.setruta(route8);
                                     break;
                    case 9:
                        rou1= rutas1.getroute13();
                            prio1 = rutas1.getP13();
                            Nf1x=  rutas1.getNf13();
                            h1x=  rutas1.geth13();

                            rou2= rutas1.getroute14();
                            prio2 = rutas1.getP14();
                            Nf2x=  rutas1.getNf14();
                            h2x=  rutas1.geth14();
                            double[][] matriz9={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                             System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                            System.out.println("");
                           if (matriz9[0][0]==13){
                               route="1-3-8-9";
                                 System.out.print(route+"\t");
                               for (int j9=1; j9 < matriz9[0].length; j9++) {
                                  System.out.print(matriz9[0][j9]+"\t");
                                }
                               System.out.println("");
                           } else {
                           }; if (matriz9[1][0]==14){
                               route="1-2-7-10-9";
                               System.out.print(route+"\t");
                               for (int j9=1; j9 < matriz9[1].length; j9++) {
                                     System.out.print(matriz9[1][j9]+"\t");
                                }
                                   System.out.println("");
                           }
```

```
                    Double route9=matriz9[matriz9.length-2][0]; // ojo con esta
inicializacion!!!
                    Double P9=matriz9[matriz9.length-2][1];  // estar pendiente
que si pongo alguna ruta mas este tiene que ser el valor de la ultima ruta
                    Double Nf9=matriz9[matriz9.length-2][2];
                    Double h9=matriz9[matriz9.length-2][3];

                    int i9, j9=1;
                        for(i9=0; i9<(matriz9.length-1); i9++){
                               if (matriz9[i9][j9] > matriz9[i9+1][j9]){
                             route9 = matriz9[i9][j9-1];
                                P9 = matriz9[i9][j9];
                             Nf9 = matriz9[i9][j9+1];
                             h9 = matriz9[i9][j9+2];
                            }
                         }

                        if (route9==13){
                                route="1-3-8-9";
                                System.out.println("The selected route will be:
");

                                System.out.println("");
                                System.out.println(route+"\t"+   P9+"\t"+
Nf9+"\t"+ h9);

                                System.out.println("");
                            } else {
                            }; if (route9==14){
                                route="1-2-7-10-9";
                                System.out.print("The selected route will be: ");
                                System.out.println("");
                                System.out.println(route+"\t"+   P9+"\t"+
Nf9+"\t"+ h9);

                                System.out.println("");
                            }
                                ru1.setrouteas(route);  // asigno los valores a
la ruta

                                          ru1.setPas(P9);
                                          ru1.setNfas(Nf9);
                                          ru1.sethas(h9);
                                          ru1.setruta(route9);
                             break;
                    case 10:
                        rou1= rutas1.getroute15();
                            prio1 = rutas1.getP15();
                            Nf1x=  rutas1.getNf15();
                            h1x=  rutas1.geth15();

                            rou2= rutas1.getroute16();
                            prio2 = rutas1.getP16();
                            Nf2x=  rutas1.getNf16();
                            h2x=  rutas1.geth16();
                            double[][] matriz10={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};


                         System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                         System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");

                         System.out.println("");
                        if (matriz10[0][0]==15){
                            route="1-3-8-9-10";
                              System.out.print(route+"\t");
                            for (int j10=1; j10 < matriz10[0].length; j10++) {
                                 System.out.print(matriz10[0][j10]+"\t");
                             }
                            System.out.println("");
                        } else {
                        }; if (matriz10[1][0]==16){
                            route="1-3-5-6-7-10";
                            System.out.print(route+"\t");
                            for (int j10=1; j10 < matriz10[1].length; j10++) {
```

```
                                System.out.print(matriz10[1][j10]+"\t");
                            }
                                System.out.println("");
                        }

                        Double route10=matriz10[matriz10.length-2][0]; // ojo con
esta inicializacion!!!
                        Double P10=matriz10[matriz10.length-2][1];  // estar
pendiente que si pongo alguna ruta mas este tiene que ser el valor de la ultima
ruta
                        Double Nf10=matriz10[matriz10.length-2][2];
                        Double h10=matriz10[matriz10.length-2][3];

                        int i10, j10=1;
                            for(i10=0; i10<(matriz10.length-1); i10++){
                                  if (matriz10[i10][j10] > matriz10[i10+1][j10]){
                                route10 = matriz10[i10][j10-1];
                                   P10 = matriz10[i10][j10];
                                Nf10 = matriz10[i10][j10+1];
                                h10 = matriz10[i10][j10+2];
                               }
                            }

                            if (route10==15){
                                    route="1-3-8-9-10";
                                    System.out.println("The selected route will be:
");

                                    System.out.println("");
                                    System.out.println(route+"\t"+   P10+"\t"+
Nf10+"\t"+ h10);

                                    System.out.println("");
                                } else {
                                }; if (route10==16){
                                    route="1-3-5-6-7-10";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P10+"\t"+
Nf10+"\t"+ h10);

                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta

                                             ru1.setPas(P10);
                                             ru1.setNfas(Nf10);
                                             ru1.sethas(h10);
                                             ru1.setruta(route10);
                                        break;
                    case 11:
                        rou1= rutas1.getroute17();
                            prio1 = rutas1.getP17();
                            Nf1x=  rutas1.getNf17();
                            h1x=  rutas1.geth17();

                            rou2= rutas1.getroute18();
                            prio2 = rutas1.getP18();
                            Nf2x=  rutas1.getNf18();
                            h2x=  rutas1.geth18();
                            double[][] matriz11={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{0, 1.10, 0, 0}};

                         System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                         System.out.println("");
                        if (matriz11[0][0]==17){
                            route="1-4-11";
                              System.out.print(route+"\t");
                            for (int j11=1; j11 < matriz11[0].length; j11++) {
                                 System.out.print(matriz11[0][j11]+"\t");
                            }
                            System.out.println("");
                        } else {
```

```
                        }; if (matriz11[1][0]==18){
                            route="1-3-5-11";
                            System.out.print(route+"\t");
                            for (int j11=1; j11 < matriz11[1].length; j11++) {
                                System.out.print(matriz11[1][j11]+"\t");
                             }
                            System.out.println("");
                        }

                        Double route11=matriz11[matriz11.length-2][0]; // ojo con
esta inicializacion!!!
                        Double P11=matriz11[matriz11.length-2][1];  // estar
pendiente que si pongo alguna ruta mas este tiene que ser el valor de la ultima
ruta
                        Double Nf11=matriz11[matriz11.length-2][2];
                        Double h11=matriz11[matriz11.length-2][3];

                        int i11, j11=1;
                            for(i11=0; i11<(matriz11.length-1); i11++){
                                  if (matriz11[i11][j11] > matriz11[i11+1][j11]){
                                 route11 = matriz11[i11][j11-1];
                                    P11 = matriz11[i11][j11];
                                 Nf11 = matriz11[i11][j11+1];
                                 h11 = matriz11[i11][j11+2];
                                }
                             }

                            if (route11==17){
                                    route="1-4-11";
                                    System.out.println("The selected route will be:
");

                                    System.out.println("");
                                    System.out.println(route+"\t"+   P11+"\t"+
Nf11+"\t"+ h11);

                                    System.out.println("");
                                } else {
                                }; if (route11==18){
                                    route="1-3-5-11";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P11+"\t"+
Nf11+"\t"+ h11);

                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta

                                            ru1.setPas(P11);
                                            ru1.setNfas(Nf11);
                                            ru1.sethas(h11);
                                            ru1.setruta(route11);
                                 break;
                    case 12:
                        rou1= rutas1.getroute19();
                            prio1 = rutas1.getP19();
                            Nf1x=  rutas1.getNf19();
                            h1x=  rutas1.geth19();

                            rou2= rutas1.getroute20();
                            prio2 = rutas1.getP20();
                            Nf2x=  rutas1.getNf20();
                            h2x=  rutas1.geth20();

                            rou3 = rutas1.getroute27();
                            prio3 = rutas1.getP27();
                            Nf3x=  rutas1.getNf27();
                            h3x=  rutas1.geth27();
                            double[][] matriz12={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{rou3,prio3,Nf3x, h3x},{0, 1.10, 0, 0}};


                         System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
```

```
                        System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                        System.out.println("");
                       if (matriz12[0][0]==19){
                           route="1-4-11-12";
                             System.out.print(route+"\t");
                           for (int j12=1; j12 < matriz12[0].length; j12++) {
                                System.out.print(matriz12[0][j12]+"\t");
                            }
                           System.out.println("");
                       } else {
                       }; if (matriz12[1][0]==20){
                           route="1-3-5-11-12";
                           System.out.print(route+"\t");
                           for (int j12=1; j12 < matriz12[1].length; j12++) {
                                System.out.print(matriz12[1][j12]+"\t");
                            }
                           System.out.println("");
                       } else {
                       }; if (matriz12[2][0]==27){
                           route="1-2-7-10-12";
                           System.out.print(route+"\t");
                           for (int j12=1; j12 < matriz12[1].length; j12++) {
                                System.out.print(matriz12[1][j12]+"\t");
                            }
                           System.out.println("");
                       }
                       Double route12=matriz12[matriz12.length-2][0]; // ojo con
esta inicializacion!!!
                       Double P12=matriz12[matriz12.length-2][1];  // estar
pendiente que si pongo alguna ruta mas este tiene que ser el valor de la ultima
ruta
                       Double Nf12=matriz12[matriz12.length-2][2];
                       Double h12=matriz12[matriz12.length-2][3];

                       int i12, j12=1;
                           for(i12=0; i12<(matriz12.length-1); i12++){
                                 if (matriz12[i12][j12] > matriz12[i12+1][j12]){
                                route12 = matriz12[i12][j12-1];
                                   P12 = matriz12[i12][j12];
                                Nf12 = matriz12[i12][j12+1];
                                h12 = matriz12[i12][j12+2];
                               }
                            }

                           if (route12==19){
                                   route="1-4-11-12";
                                   System.out.println("The selected route will be:
");
                                   System.out.println("");
                                   System.out.println(route+"\t"+   P12+"\t"+
Nf12+"\t"+ h12);
                                   System.out.println("");
                               } else {
                               }; if (route12==20){
                                   route="1-3-5-11-12";
                                   System.out.print("The selected route will be: ");
                                   System.out.println("");
                                   System.out.println(route+"\t"+   P12+"\t"+
Nf12+"\t"+ h12);
                                   System.out.println("");
                               } else{
                                }; if (route12==27){
                                       route="1-2-7-10-12";
                                   System.out.print("The selected route will be: ");
                                   System.out.println("");
                                   System.out.println(route+"\t"+   P12+"\t"+
Nf12+"\t"+ h12);
                                   System.out.println("");
                                      }
                                   ru1.setrouteas(route);  // asigno los valores a
la ruta
                                           ru1.setPas(P12);
```

```
                                        ru1.setNfas(Nf12);
                                        ru1.sethas(h12);
                                        ru1.setruta(route12);
                                break;
                    case 13:
                        rou1= rutas1.getroute21();
                            prio1 = rutas1.getP21();
                            Nf1x=  rutas1.getNf21();
                            h1x=  rutas1.geth21();

                            rou2= rutas1.getroute22();
                            prio2 = rutas1.getP22();
                            Nf2x=  rutas1.getNf22();
                            h2x=  rutas1.geth22();

                            rou3 = rutas1.getroute25();
                            prio3 = rutas1.getP25();
                            Nf3x=  rutas1.getNf25();
                            h3x=  rutas1.geth25();
                            double[][] matriz13={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{rou3,prio3, Nf3x, h3x},{0, 1.10, 0, 0}};

                         System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                            System.out.println("");
                           if (matriz13[0][0]==21){
                               route="1-4-11-13";
                                 System.out.print(route+"\t");
                               for (int j13=1; j13 < matriz13[0].length; j13++) {
                                    System.out.print(matriz13[0][j13]+"\t");
                                }
                               System.out.println("");
                           } else {
                           }; if (matriz13[1][0]==22){
                               route="1-3-5-11-13";
                               System.out.print(route+"\t");
                               for (int j13=1; j13 < matriz13[1].length; j13++) {
                                    System.out.print(matriz13[1][j13]+"\t");
                                }
                                  System.out.println("");

                           }else {
                           }; if (matriz13[2][0]==25){
                               route="1-2-7-10-9-13";
                               System.out.print(route+"\t");
                               for (int j13=1; j13 < matriz13[1].length; j13++) {
                                    System.out.print(matriz13[1][j13]+"\t");
                                }
                                  System.out.println("");

                           }

                           Double route13=matriz13[matriz13.length-2][0]; // ojo con
esta inicializacion!!!
                           Double P13=matriz13[matriz13.length-2][1];  // estar
pendiente que si pongo alguna ruta mas este tiene que ser el valor de la ultima
ruta
                           Double Nf13=matriz13[matriz13.length-2][2];
                           Double h13=matriz13[matriz13.length-2][3];

                           int i13, j13=1;
                               for(i13=0; i13<(matriz13.length-1); i13++){
                                      if (matriz13[i13][j13] > matriz13[i13+1][j13]){
                                    route13 = matriz13[i13][j13-1];
                                      P13 = matriz13[i13][j13];
                                    Nf13 = matriz13[i13][j13+1];
                                    h13 = matriz13[i13][j13+2];
                                   }
                                }

                               if (route13==21){
```

```
                                    route="1-4-11-13";
                                    System.out.println("The selected route will be:
");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P13+"\t"+
Nf13+"\t"+ h13);
                                    System.out.println("");
                                } else {
                                }; if (route13==22){
                                    route="1-3-5-11-13";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P13+"\t"+
Nf13+"\t"+ h13);
                                    System.out.println("");
                                }else {
                                }; if (route13==25){
                                    route="1-2-7-10-9-13";
                                    System.out.print("The selected route will be: ");
                                    System.out.println("");
                                    System.out.println(route+"\t"+   P13+"\t"+
Nf13+"\t"+ h13);
                                    System.out.println("");
                                }
                                    ru1.setrouteas(route);  // asigno los valores a
la ruta
                                             ru1.setPas(P13);
                                             ru1.setNfas(Nf13);
                                             ru1.sethas(h13);
                                             ru1.setruta(route13);
                                        break;
                    case 14:
                        rou1= rutas1.getroute23();
                            prio1 = rutas1.getP23();
                            Nf1x=  rutas1.getNf23();
                            h1x=  rutas1.geth23();

                            rou2= rutas1.getroute24();
                            prio2 = rutas1.getP24();
                            Nf2x=  rutas1.getNf24();
                            h2x=  rutas1.geth24();

                            rou3= rutas1.getroute26();
                            prio3 = rutas1.getP26();
                            Nf3x=  rutas1.getNf26();
                            h3x=  rutas1.geth26();
                            double[][] matriz14={{rou1,prio1, Nf1x,
h1x},{rou2,prio2,Nf2x, h2x},{rou3,prio3, Nf3x, h3x},{0, 1.10, 0, 0}};

                           System.out.println("The possible routes to get to the
destination node " +DN+ " from the source node 0, are:");
                            System.out.print("Route"+"\t"+"\t"+"  P"+"\t"+" Nf"+"\t"+"
H"+"\t");
                           System.out.println("");
                           if (matriz14[0][0]==23){
                               route="1-3-8-14";
                                 System.out.print(route+"\t");
                               for (int j14=1; j14 < matriz14[0].length; j14++) {
                                    System.out.print(matriz14[0][j14]+"\t");
                                }
                               System.out.println("");
                           } else {
                           } if (matriz14[1][0]==24){
                               route="1-3-5-11-13-14";
                               System.out.print(route+"\t");
                               for (int j14=1; j14 < matriz14[1].length; j14++) {
                                    System.out.print(matriz14[1][j14]+"\t");
                                }
                               System.out.println("");
                           }
                           else {
                           } if (matriz14[2][0]==26){
                               route="1-2-7-10-14";
```

```
                        System.out.print(route+"\t");
                        for (int j14=1; j14 < matriz14[2].length; j14++) {
                             System.out.print(matriz14[2][j14]+"\t");
                         }
                        System.out.println("");
                    }

                    Double route14=matriz14[matriz14.length-2][0]; // ojo con
esta inicializacion!!!
                    Double P14=matriz14[matriz14.length-2][1];  // estar
pendiente que si pongo alguna ruta mas este tiene que ser el valor de la ultima
ruta
                    Double Nf14=matriz14[matriz14.length-2][2];
                    Double h14=matriz14[matriz14.length-2][3];

                    int i14, j14=1;
                        for(i14=0; i14<(matriz14.length-1); i14++){
                              if (matriz14[i14][j14] > matriz14[i14+1][j14]){
                             route14 = matriz14[i14][j14-1];
                               P14 = matriz14[i14][j14];
                             Nf14 = matriz14[i14][j14+1];
                             h14 = matriz14[i14][j14+2];
                            }
                         }

                        if (route14==23){
                                route="1-3-8-14";
                                System.out.println("The selected route will be:
");

                                System.out.println("");
                                System.out.println(route+"\t"+   P14+"\t"+
Nf14+"\t"+ h14);

                                System.out.println("");
                            } else {
                            } if (route14==24){
                                route="1-3-5-11-13-14";
                                System.out.print("The selected route will be: ");
                                System.out.println("");
                                System.out.println(route+"\t"+   P14+"\t"+
Nf14+"\t"+ h14);

                                System.out.println("");
                            } else {
                            } if (route14==26){
                                route="1-2-7-10-14";
                                System.out.print("The selected route will be: ");
                                System.out.println("");
                                System.out.println(route+"\t"+   P14+"\t"+
Nf14+"\t"+ h14);

                                System.out.println("");
                            }
                                ru1.setrouteas(route);  // asigno los valores a
la ruta
                                          ru1.setPas(P14);
                                          ru1.setNfas(Nf14);
                                          ru1.sethas(h14);
                                          ru1.setruta(route14);
                                          System.out.println("la ruta es
"+route14);
                                break;

                }

                return ru1;
   }
}
```

- Phase2 Class:

```
public class phase2 {
```

```
  public packets2 routefinder (packets bcp1, ruta route1) {   //consigo ruta
asigno, route/P/Nf/h
    //necesitos los packets provenientes de phase 1 y la ruta

  packets2 pbcpr1 = new packets2();


  Integer destinobcp = bcp1.getDA();   //valores de bcp phase1
  Integer claseser = bcp1.getc0s();
  Integer source = bcp1.getSA();
  Double seq = bcp1.getNseq();
  Integer c1 = bcp1.getc1();
  Integer c2 = bcp1.getc2();
  Integer c3 = bcp1.getc3();
  Integer N = bcp1.getN();
  String ruta = route1.getrouteas();            //valores dados por la ruta a usar
  Double prioridad = route1.getPas();
  Double numerofed = route1.getNfas();
  Double saltos = route1.gethas();
  pbcpr1.setDA(destinobcp);    // quiero pasarle el SA, DA, c0s definidos phase1
  pbcpr1.setSA(source);
  pbcpr1.setc0s(claseser);
  pbcpr1.setroute(ruta);       //valores de la ruta nuevos
  pbcpr1.setP(prioridad);
  pbcpr1.setNf(numerofed);
  pbcpr1.seth(saltos);
  pbcpr1.setNseq(seq);
  pbcpr1.setN(N);
  pbcpr1.setc1(c1);
  pbcpr1.setc2(c2);
  pbcpr1.setc3(c3);


  return pbcpr1; // tengo que devolver el packets2
   }
 }
```

- Phase2b Class:

```
public class phase2b {

public packetsbcp offsetcal (packets2 bcppr2){

  packetsbcp pbcptot = new packetsbcp();


  double tb1= (32*(double)8)/10000000; //le quite 3 ceros al num y den
  double tb2= (64*(double)8)/10000000;
  double tb3= (128*(double)8)/10000000;
  double tbcp=0.000032;
  double tp=0.00001;
  double tc=0.0000025;


  Integer destinobcptot = bcppr2.getDA();
  Integer origen = bcppr2.getSA();
  Integer clasesertot = bcppr2.getc0s();
  String rutatot = bcppr2.getroute();
  Double prioridadtot = bcppr2.getP();
  Double numerofedtot = bcppr2.getNf();
  Double saltostot = bcppr2.geth();
  Double seq = bcppr2.getNseq();
  Integer N = bcppr2.getN();
  Integer c1 = bcppr2.getc1();
  Integer c2 = bcppr2.getc2();
  Integer c3 = bcppr2.getc3();

  pbcptot.setDA(destinobcptot);
  pbcptot.setSA(origen);
  pbcptot.setc0s(clasesertot);
```

```
  pbcptot.setroute(rutatot);
  pbcptot.setP(prioridadtot);
  pbcptot.setNf(numerofedtot);
  pbcptot.seth(saltostot);
  pbcptot.setN(N);
  pbcptot.setNseq(seq);
  pbcptot.setc1(c1);
  pbcptot.setc2(c2);
  pbcptot.setc3(c3);


    switch (clasesertot) {
    case 1:
    double eot= (tb2+tb3);
    double ot=tbcp+tp+tc+eot;
    double ttl=saltostot*(ot+tb1);

    pbcptot.setot(ot);
    pbcptot.setttl(ttl);
    break;

       case 2:
       double eot1=tb1+tb3;
       double ot1=tbcp+tp+tc+eot1;
    double ttl1=saltostot*(ot1+tb2);

    pbcptot.setot(ot1);
    pbcptot.setttl(ttl1);
    break;

       case 3:
       double eot2=tb1+tb2;
       double ot2=tbcp+tp+tc+eot2;
    double ttl2=saltostot*(ot2+tb3);

    pbcptot.setot(ot2);
    pbcptot.setttl(ttl2);
    break;
   }
    return pbcptot;
}
}
```

- Bcpcreation Class:

```
public class bcpcreation {
       public bcpout crearbcps (packetsbcp p3, contlink cl2) {

              bcpout pbcp = new bcpout();

               Integer SA=p3.getSA();
               Integer DA=p3.getDA();
               Integer c0s=p3.getc0s();
               Double Nseq=p3.getNseq();
               String route=p3.getroute();
               Double h=p3.geth();
               Double ot = p3.getot();
               Integer N=p3.getN();
               int link1=cl2.getlink1();
               int link2=cl2.getlink2();
               int link3=cl2.getlink3();
               int link4=cl2.getlink4();
               int link5=cl2.getlink5();
               int link6=cl2.getlink6();
               int link7=cl2.getlink7();
               int link8=cl2.getlink8();
               int link9=cl2.getlink9();
               int link10=cl2.getlink10();
               int link11=cl2.getlink11();
               int link12=cl2.getlink12();
               int link13=cl2.getlink13();
```

```
                int link14=cl2.getlink14();
                int link15=cl2.getlink15();
                int link16=cl2.getlink16();
                int link17=cl2.getlink17();
                int link18=cl2.getlink18();
                int link19=cl2.getlink19();
                int link20=cl2.getlink20();
                int link21=cl2.getlink21();


                pbcp.setDN0(1);
                double c = h;
                //rutas que salen por el nodo 2
                if (route=="1-2" | route=="1-2-4" | route=="1-2-7" | route=="1-2-
7-6" | route=="1-2-7-10-9" | route=="1-2-7-10-9-8" | route=="1-2-7-10-9-13" |
route=="1-2-7-10-14"| route=="1-2-7-10-12"){
                pbcp.setDNF0(2);
                pbcp.setDN2(2);
                cl2.setlink1(link1+1);
                }
                if (route=="1-2-7" | route=="1-2-7-6" | route=="1-2-7-10-9" |
route=="1-2-7-10-9-8" | route=="1-2-7-10-9-13"| route=="1-2-7-10-14"| route=="1-2-
7-10-12"){
            pbcp.setDNF2(7);
                pbcp.setDN3(7);
                cl2.setlink5(link5+1);
                }
                if (route=="1-2-4"){
                        pbcp.setDNF2(4);
                        pbcp.setDN3(4);
                        cl2.setlink1(link4+1);
                }
                if (route=="1-2-7-10-9-13"){
                        pbcp.setDNF5(13);
                        pbcp.setDN6(13);
                        cl2.setlink16(link16+1);
                }
                if (route=="1-2-7-6"){
                        pbcp.setDNF3(6);
                        pbcp.setDN4(6);
                        cl2.setlink11(link11+1);
                }
                if (route=="1-2-7-10-14"){
                        pbcp.setDNF3(10);
                        pbcp.setDN4(10);
                        cl2.setlink12(link12+1);
                        pbcp.setDNF4(14);
                    pbcp.setDN5(14);
                    cl2.setlink17(link17+1);

                }
                if (route=="1-2-7-10-12"){
                        pbcp.setDNF3(10);
                        pbcp.setDN4(10);
                        cl2.setlink12(link12+1);
                        pbcp.setDNF4(12);
                    pbcp.setDN5(12);
                    cl2.setlink18(link18+1);

                }
                if (route=="1-2-7-10-9" | route=="1-2-7-10-9-8" | route=="1-2-7-
10-9-13"){
                        pbcp.setDNF3(10);
                        pbcp.setDN4(10);
                        cl2.setlink12(link12+1);
                        pbcp.setDNF4(9);
                    pbcp.setDN5(9);
                    cl2.setlink15(link15+1);
                }
                if (route=="1-2-7-10-9-8"){
                        pbcp.setDNF5(8);
                        pbcp.setDN6(8);
                        cl2.setlink13(link13+1);
```

```
                }
                //rutas que salen por el nodo 4
                if (route=="1-4" | route=="1-4-2" | route=="1-4-11" | route=="1-4-
11-5" | route=="1-4-11-5-3" | route=="1-4-11-12" | route=="1-4-11-13"){
                        pbcp.setDNF0(4);
                        pbcp.setDN2(4);
                        cl2.setlink3(link3+1);
                }
                if (route=="1-4-2"){
                        pbcp.setDNF2(2);
                        pbcp.setDN3(2);
                        cl2.setlink4(link4+1);
                }
                if (route=="1-4-11" | route=="1-4-11-5" | route=="1-4-11-5-3" |
route=="1-4-11-12" | route=="1-4-11-13"){
                        pbcp.setDNF2(11);
                        pbcp.setDN3(11);
                        cl2.setlink8(link8+1);
                }
                if (route=="1-4-11-5" | route=="1-4-11-5-3"){
                        pbcp.setDNF3(5);
                        pbcp.setDN4(5);
                        cl2.setlink10(link10+1);
                }
                if (route=="1-4-11-5-3"){
                        pbcp.setDNF4(3);
                        pbcp.setDN5(3);
                        cl2.setlink7(link7+1);
                }
                if (route=="1-4-11-12"){
                        pbcp.setDNF3(12);
                        pbcp.setDN4(12);
                        cl2.setlink19(link19+1);
                }
                if (route=="1-4-11-13"){
                        pbcp.setDNF3(13);
                        pbcp.setDN4(13);
                        cl2.setlink20(link20+1);
                }
                //rutas que salen por el nodo 3
                if (route=="1-3" | route=="1-3-5" | route=="1-3-5-6" | route=="1-
3-5-6-7" | route=="1-3-8" | route=="1-3-8-9" | route=="1-3-8-9-10" | route=="1-3-
5-6-7-10" | route=="1-3-5-11" | route=="1-3-5-11-12" | route=="1-3-5-11-12" |
route=="1-3-5-11-13" | route=="1-3-8-14" | route=="1-3-5-11-13-14"){
                        pbcp.setDNF0(3);
                        pbcp.setDN2(3);
                        cl2.setlink2(link2+1);
                }
                if (route=="1-3-5" | route=="1-3-5-6" | route=="1-3-5-6-7"|
route=="1-3-5-6-7-10" | route=="1-3-5-11" | route=="1-3-5-11-12" | route=="1-3-5-
11-12" | route=="1-3-5-11-13" | route=="1-3-5-11-13-14"){
                        pbcp.setDNF2(5);
                        pbcp.setDN3(5);
                        cl2.setlink7(link7+1);
                }
                if (route=="1-3-5-6" | route=="1-3-5-6-7" | route=="1-3-5-6-7-
10" ){
                        pbcp.setDNF3(6);
                        pbcp.setDN4(6);
                        cl2.setlink9(link9+1);
                }
                if (route=="1-3-5-6-7"| route=="1-3-5-6-7-10" ){
                        pbcp.setDNF4(7);
                        pbcp.setDN5(7);
                        cl2.setlink11(link11+1);
                }
                if (route=="1-3-5-6-7-10" ){
                        pbcp.setDNF5(10);
                        pbcp.setDN6(10);
                        cl2.setlink12(link12+1);
                }
                if (route=="1-3-5-11" | route=="1-3-5-11-12" | route=="1-3-5-11-
13" | route=="1-3-5-11-13-14" ){
```

```
                        pbcp.setDNF3(11);
                        pbcp.setDN4(11);
                        cl2.setlink10(link10+1);
                }
                if (route=="1-3-5-6-12" ){
                        pbcp.setDNF4(12);
                        pbcp.setDN5(12);
                        cl2.setlink18(link18+1);
                }
                if (route=="1-3-5-11-13" | route=="1-3-5-11-13-14"){
                        pbcp.setDNF4(13);
                        pbcp.setDN5(13);
                        cl2.setlink20(link20+1);
                }
                if (route=="1-3-5-11-13-14"){
                        pbcp.setDNF5(14);
                        pbcp.setDN6(14);
                        cl2.setlink21(link21+1);
                }

                if (route=="1-3-8" | route=="1-3-8-9" | route=="1-3-8-9-10" |
route=="1-3-8-14" ){
                        pbcp.setDNF2(8);
                        pbcp.setDN3(8);
                        cl2.setlink6(link6+1);
                }
                if (route=="1-3-8-9" | route=="1-3-8-9-10" ){
                        pbcp.setDNF3(9);
                        pbcp.setDN4(9);
                        cl2.setlink13(link13+1);
                }
                if (route=="1-3-8-9-10" ){
                        pbcp.setDNF4(10);
                        pbcp.setDN5(10);
                        cl2.setlink15(link15+1);
                }
                if (route=="1-3-8-14" ){
                        pbcp.setDNF3(14);
                        pbcp.setDN4(14);
                        cl2.setlink14(link14+1);
                }
                Integer DN = pbcp.getDN0();
                Integer DNF = pbcp.getDNF0();
                pbcp.setSA0(SA);
                pbcp.setDA0(DA);
                pbcp.setc0s0(c0s);
                pbcp.setNseq0(Nseq);
                pbcp.setroute0(route);
                pbcp.setot0(ot);
                pbcp.setN0(N);

                //      System.out.println("the first bcp packet to send will be "
+Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN+" " +DNF+ " "+ot+" " +N);

               c= c-1;
               if (c>0){
                       pbcp.setSA2(SA);
                       pbcp.setDA2(DA);
                       pbcp.setc0s2(c0s);
                       pbcp.setNseq2(Nseq);
                       pbcp.setroute2(route);
                       pbcp.setot2(ot*2);
                       pbcp.setN2(N);

                       Double ot2 = pbcp.getot2();
                        Integer DN2 = pbcp.getDN2();
                        Integer DNF2 = pbcp.getDNF2();
                        //          System.out.println("the second bcp packet to
send will be " +Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN2+" " +DNF2+ "
"+ot2+" " +N);
               } c= c-1;

               if (c>0){
```

```
                        pbcp.setSA3(SA);
                        pbcp.setDA3(DA);
                        pbcp.setc0s3(c0s);
                        pbcp.setNseq3(Nseq);
                        pbcp.setroute3(route);
                        pbcp.setot3(ot*3);
                        pbcp.setN3(N);

                        Double ot3 = pbcp.getot3();
                         Integer DN3 = pbcp.getDN3();
                         Integer DNF3 = pbcp.getDNF3();
                         //                System.out.println("the third bcp
packet to send will be " +Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN3+" "
+DNF3+ " "+ot3+" " +N);

                } c= c-1;
                if (c>0){
                        pbcp.setSA4(SA);
                        pbcp.setDA4(DA);
                        pbcp.setc0s4(c0s);
                        pbcp.setNseq4(Nseq);
                        pbcp.setroute4(route);
                        pbcp.setot4(ot*4);
                        pbcp.setN4(N);

                        Double ot4 = pbcp.getot4();
                         Integer DN4 = pbcp.getDN4();
                         Integer DNF4 = pbcp.getDNF4();
                         //          System.out.println("the forth bcp packet to
send will be " +Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN4+" " +DNF4+ "
"+ot4+" " +N);

                } c= c-1;
                if (c>0){
                        pbcp.setSA5(SA);
                        pbcp.setDA5(DA);
                        pbcp.setc0s5(c0s);
                        pbcp.setNseq5(Nseq);
                        pbcp.setroute5(route);
                        pbcp.setot5(ot*5);
                        pbcp.setN5(N);

                        Double ot5 = pbcp.getot5();
                         Integer DN5 = pbcp.getDN5();
                         Integer DNF5 = pbcp.getDNF5();
                         //    System.out.println("the fifth bcp packet to send will
be " +Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN5+" " +DNF5+ " "+ot5+" " +N);
                } c= c-1;
                if (c>0){
                        pbcp.setSA6(SA);
                        pbcp.setDA6(DA);
                        pbcp.setc0s6(c0s);
                        pbcp.setNseq6(Nseq);
                        pbcp.setroute6(route);
                        pbcp.setot6(ot*6);
                        pbcp.setN6(N);

                        Double ot6 = pbcp.getot6();
                         Integer DN6 = pbcp.getDN6();
                         Integer DNF6 = pbcp.getDNF6();
                         //          System.out.println("the fifth bcp packet to
send will be " +Nseq+" "+DA+" "+SA +" " +c0s+" " +route+" " +DN6+" " +DNF6+ "
"+ot6+" " +N);
                }
        return pbcp;
        }
}
```

Annexes B

Reference Results Paper