UNIVERSITY OF BEIRA INTERIOR

Department of Informatics

Resource Reservation Protocols for Optical Burst Switched Networks

José Manuel Boavida Gregório

Supervisor: Prof. Joel José Puga Coelho Rodrigues

MSc in Informatics Engineering

University of Beira Interior
August 24, 2009

Acknowledgments

In this lines, first I would like to thank my supervisor Prof. Joel Rodrigues for all the support and constant motivation. Without his help, this work would not be possible.

I am grateful to Dr. George N.Rouskas and Dr. Jing Teng, for the permission to use their OBS simulator, which supports this work. I thank University of Beira Interior, the Institute of Telecommunication - Covilhã Lab and the NetGNA Group, for the support and availability of their infrastructures. Finally, I must thank so much to my family, my girlfriend and her family for all dedication and encouragement.

Thank you all.

Resumo

Nesta dissertação é feito um estudo em redes com comutação óptica de agregados de pacotes (redes OBS). Assim, este estudo apresenta e descreve os conceitos mais importantes relacionados com as redes OBS.

Inicialmente é analisado o estado da arte, efectuando uma descrição detalhada da arquitectura de uma rede OBS.

Seguidamente analisam-se os protocolos de reserva unidireccional de recursos (JIT, JIT$^+$, JET, Jumpstart, Horizon and E-JIT). Para isso são utilizadas redes com topologias regulares (em anel e em malha) com um número variável de nós, e também irregulares (NSFNET e ARPANET). É também apresentado um novo protocolo de reserva unidireccional de recursos chamado E-JIT$^+$.

Este novo protocolo baseia-se no protocolo já existente JIT$^+$ e tenta optimizar o seu desempenho. Para melhor descrever o modo de operação deste protocolo proposto é apresentada a especificação formal do mesmo. Depois da apresentação deste protocolo analisa-se o seu desempenho. Para isso são utilizadas as topologias de rede referidas anteriormente e também um simulador de redes OBS adaptado de forma a suportar o novo protocolo.

O simulador utilizado, de nome OBSSimulator, devolve os valores de probabilidade de perda dos agregados de pacotes em cada salto (hop) na rede. Deste modo, foram considerados vários factores que podem influenciar o desempenho dos protocolos de reserva unidireccional de recursos, tais como o número de nós da topologia de rede utilizada, a quantidade de tráfego na rede, o ganho do grau nodal, o tempo de processamento das mensagens de setup e o tempo de configuração dos comutadores ópticos. No fim deste trabalho concluiu-se que o protocolo proposto, E-JIT$^+$, melhora o desempenho em relação aos outros protocolos de reserva unidireccional de recursos estudados, nos casos estudados.

Contents

List of Figures

Acronyms and Abreviations

ARPANET Advanced Research Projects Agency Network

DWDM Dense Wavelength Division Multiplexing

DnT Degree n topology

EON European Optical Network

E-JIT Enhanced Just-In-Time

E-JIT$^+$ Enhanced Just-In-Time Plus

EFSM Extended finite state machine

FCFS First-come, first-served

FDL Fiber Delay Lines

IP Internet Protocol

ISO International Organization for Standardization

ITU-T International Telecmmunication Union Telecommunication Standardization Sector

JIT Just-In-Time

JIT$^+$ Just-In-Time Plus

JITPAC Just-In-Time Protocol Acceleration Unit

OBS Optical Burst Switching

OSI Open Systems interconnection

OXC Optical Cross Connect

OPS Optical Packet Switching

QoS Quality of Service

SCU Switch Control Unit

TCP Transmission Control Protocol

WDM Wave Division Multiplexing

Chapter 1

Introduction

1.1 Focus

Optical networks are high-capacity telecommunication networks based on optical technologies. In the year 1980 a revolution in networks began when fiber-optic technology started to be applied in telecommunication networks. Since then, tremendous advances have been made and the quality of networks has improved significantly.

Networks that once provided basic telephone service through a local operator are now transmitting the equivalent of thousands of encyclopedias per second.Submarine cables with optical amplifiers operate at speeds up to 5 gigabits per second (Gbps). The biggest challenge remaining for fiber optics is economic. Today telephone and cable television companies can cost-justify installing fiber links to remote sites serving tens to a few hundred customers.[1]

One of the many reasons for the introduction of fiber optic technology is the speed of transmission. Fiber optic networks operate at speeds up to 10 Gbps, as opposed to 1.54 megabits per second (Mbps) for copper. One other reason is the immunity to electrical and radiofrequency interference. Fiber optic cables have a greater resistance to electromagnetic noise from items such as radios, motors, or other nearly cables. Because optical fibers carry beams of light, they are free of electrical noise and interference[1]

This dissertation focuses on the scope of Optical Burst Switched Networks. Burts are sets of packets, which are aggregated and transmitted over an optical network. In order to correctly switch these bursts a resource reservation protocol must be used. Hence, this study focuses in the resource reservation protocols describing their mode operation and analysing their performance.

1.2 Objectives

The main goal of this dissertation is to study all the concepts related with OBS networks and use the knowledge acquired to develop a new resource reservation protocol called E-JIT$^+$. Thus, all the underlying concepts related with OBS networks are analyzed. One of the most important issues in this approach is the OBS network architecture, because it introduces the most important concepts related with OBS networks, so this architecture is described in more detail. Another objective is to describe one-way resource reservation protocols developed until nowadays (JIT, JIT$^+$, JET, Jumpstart, Horizon and E-JIT).

Related with the proposal of the new resource reservation protocol, E-JIT$^+$, the following objectives where identified:

- Analyze the E-JIT$^+$protocol operation;
- Present the formal specification of E-JIT$^+$;
- Performance evaluation of E-JIT$^+$, compared with the performance of existent protocols.

1.3 Main Contributions

The main contribution in this study is the proposal of a new resource reservation protocol for Optical Burst Switched Networks (OBS Networks) called E-JIT$^+$. To support this proposal a formal specification of this new resource reservation protocol is presented.

The performance of the proposed protocol, E-JIT$^+$, is evaluated and compared with other resource reservation protocols (JIT$^+$ and E-JIT). In this study, different metrics based on the burst loss probability where used: the influence of the number of nodes, the nodal degree gain, the effects of the OXC configuration time and setup message processing time and the impact of the amount of traffic in the network (λ/μ).

Several network topologies both regular and irregular are considered in this performance evaluation. Examples of used regular topologies are rings, chordal rings and Mesh-torus and irregular are ARPANET, NSFNET and EON.

1.4 Organization

This dissertation is organized in six chapters. The first chapter is the introduction, on which the focus, objectives, main contributions and organization are presented. In the second chapter, is presented a concept approach of Optical Burst Switched Networks. Still in the second chapter the architecture of an OBS network, including edge nodes and core nodes is presented.

The third chapter is dedicated to the analysis and description of the existent resource reservation protocols for optical burst switched networks. Also in chapter three, a new resource reservation protocol is presented, called E-JIT$^+$. The formal specification of this new resource reservation protocol is also specified.

The fourth chapter presents a brief description of the existent OBS simulators and the OBS simulator used in this study to evaluate the performance of E-JIT$^+$.

Chapter four presents also the various network topologies, the traffic parameters of an OBS network model and the various performance metrics, used to evaluate the performance of the new resource reservation protocol, E-JIT$^+$.

Chapter number five, presents the performance assessment of the E-JIT$^+$ resource reservation protocol, analyzing the various parameters referred in the fourth chapter.

Finally, the last chapter is dedicated to present our conclusions obtained with this work, mainly the evaluation results obtained with the new resource reservation protocol E-JIT$^+$.

Chapter 2

Optical Burst Switched Networks

2.1 The concept approach

The burst switching concept appeared for the first time in 1980, proposed by Haselton[2] and Amstutz[3] in the scope of voice communications. Ten years later in 1990 Optical Burst Switching was proposed as a new switching paradigm for optical networks.

OBS is a complex concept that can't be completely defined in a simple paragraph. However a set of characteristics proposed by Dolzer et al.[4] can be enumerated.

The first characteristic, Granularity, is the transmission unit size, called "burst" which is between the Optical circuit switching and optical packet switching. Other characteristic is the separation between control and data, i.e. control messages are transmitted first in a dedicated channel or wavelength called control channel and bursts are transmitted after some time offset relatively to the corresponding control message through data channels.

Other issue is the resource allocation which is done without using explicit two-way reservation, so the source node starts transmitting without waiting for the acknowledge message from the destination node. Another characteristic is the variable burst length, according to a specified burst assembly policy.

Finally there is no need to have optical buffering, because OBS routers are previously configured and when the burst arrives is automatically routed to the output port.

Resuming, in optical burst switching a variable number of packets are aggregated into variable sized bursts and then transported trough a data channel shortly after a setup message as been sent on a separate control channel. When the router receives a setup message processes it and according to a given resource reservation protocol, if the channel can be reserved to the corresponding burst then it is routed to the next node. If the channel can't be reserved at that time then the burst is dropped.

2.2 OBS Network Architecture

The architecture of a network is the structure and behaviour definition of the system that composes the network. This architecture is very important because other interconnected systems need that information to communicate with that network[5].

The OBS network architecture is based on the Open Systems Interconnection (OSI) model and the Transmission Control Protocol/Internet Protocol (TCP/IP) architecture. It has five layers (from the lower to the top: OBS, Network, Transport, Application). The OBS layer provides mechanisms of signalling, transmission and reception of optical bursts, optical to electronic conversion of control packets, wavelength multiplexing and demultiplexing and resource reservation among others[6].

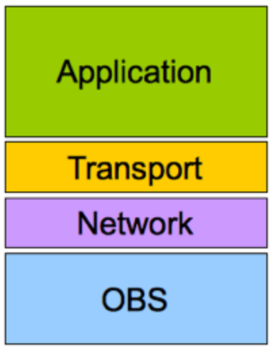


Figure 2.1: OBS network layers architecture.

An OBS network is constituted by a set of OBS nodes connected by DWDM links (figure 2.2). Each DWDM link can have hundreds of data channels and these channels are dynamically chosen to transmit a burst. Usually one channel per link (but possible more) is used to control the assignment of data channels to data bursts.

OBS nodes can be edge nodes or core nodes. Edge nodes are at the periphery of the OBS network and core nodes are inside the network.

All the process of burst assembly and buffering is made at the edge nodes, then bursts are sent trough the core network and are routed by the core nodes [7].

2.2.1 Edge Node

The role of an OBS edge node is to perform all the operations needed to interact with the external network having IP routers and carrying data through IP packets. Briefly, these operations are[8]:

- Assemblage of IP packets into data bursts;
- Generate and schedule the control packet for each data burst;

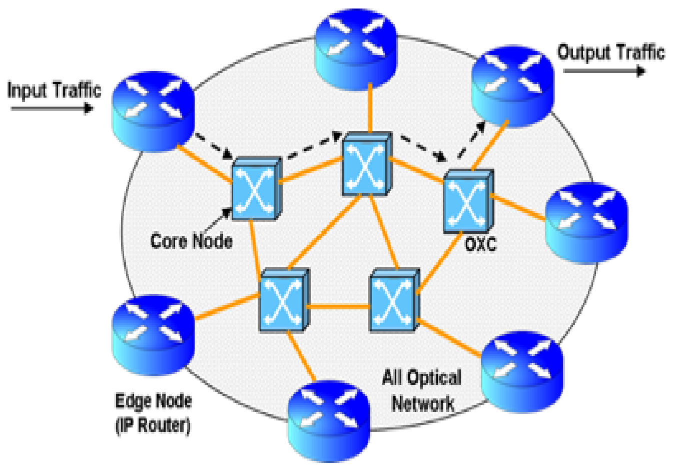


Figure 2.2: Representation of an OBS network inside an IP network.

- Convert the traffic from electronic signal into optical domain and multiplex it into WDM wavelength;
- Demultiplex incoming wavelength channels and converted the traffic received from optical to electronic signal;
- Disassemble and forward IP packets to the IP routers connected to it;

Given these operations the edge node can be divided into three modules. The Burst Assembly module, whose task is the assemblage of data bursts from the coming IP packets. The routing module, which receives the incoming traffic and chooses the appropriate output port for each packet and sends it to the corresponding burst assemble module. The third module is the scheduler, which in conjunction with the burst assembly module creates the burst and the corresponding control packet and forwards it to the output port.

2.2.2 Core Node

OBS Core nodes are the nodes inside the OBS network backbone. Each OBS core node needs to be able to perform the routing of data bursts in order that they reach their destination node.

Core nodes are sub-divided into two main modules[6, 9]: the Switch Control Unit (SCU) and the Optical Cross-Connect (OXC). The SCU converts the control packets from optical to electronic signal and then processes them in order to determine the right output port.

The control packet has information about the arrival time and the duration of the corresponding data bursts, so the SCU can determine if the output port is available when the data burst arrives and if positive then the data burst can be forward to that port, else according to the contention resolution implemented the burst could be dropped or the current burst occupying that port is dropped or segmented.

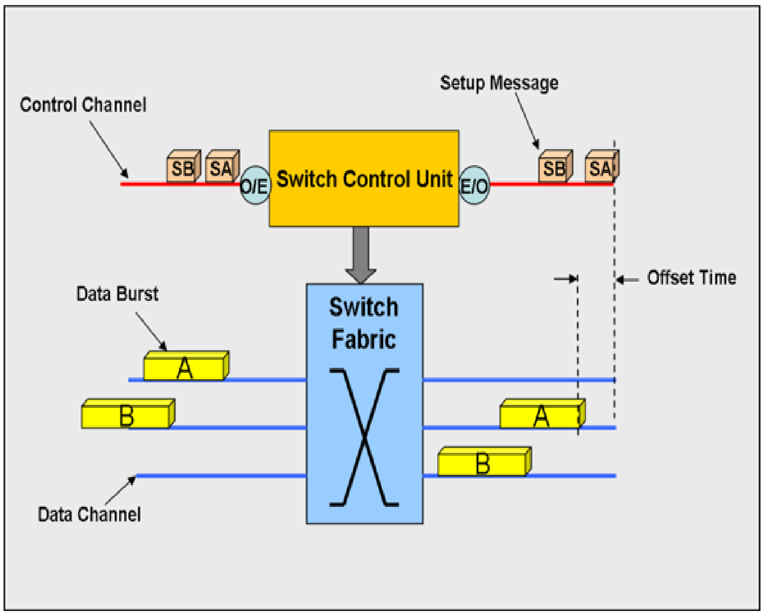


Figure 2.3: Schematic representation of an OBS core node.

The OXC is an optical switch with a fixed number of input fibers and the same number of output fibers, it has a mechanism called optic switch matrix to forward data bursts to the right next node. This process is performed without the need to convert the optical signal into electronic, because the OXC is previously configured by the SCU and when the data burst arrives it simple passes trough the OXC to the right output port.

This is a great advantage in OBS comparing to Optical Packet Switching (OPS) because only one setup message per burst is converted to electric signal while in OPS every IP packets have a header, which needs to be extracted, converted to electric signal and then processed.

After every channel reservation the core node needs to regenerate the control packet, convert it back to optical domain and send it to the next node in order to prepare the next hop.

Chapter 3

Resource Reservation Protocols

3.1 Introduction

As mentioned in the above section the SCU module of a core node needs to configure the OXC to make possible the transmission of the data burst trough the right port in order to be sent right to the next node [6]. But as above-mentioned, problems may occur if the data channel is already reserved to another burst or if the data burst comes first than the control packet.

By these and other reasons it's very important to have a set of rules so that switches can be correctly configured in order to minimize the number of lost bursts without compromise. For this reason, to implement this set of rules, resource reservation protocols are used.

Resource reservation protocols can be classified regarding the following characteristics [7]:

- one-way or two-way reservation;
- immediate or delayed reservation;
- implicit or explicit release;
- centralized or distributed;

This study focuses on the one-way reservation protocols, because is the class of the most important protocols present in the actual literature.

One-way reservation protocols can be subdivided into immediate reservation and delayed reservation. In immediate reservation a channel is reserved for the burst as soon as possible after the reception of the setup message and in delayed reservation the channel is reserved only before the arrival of the first bit of the burst.

Furthermore if a node needs to receive a release message from the previous node to release OXC resources, then is classified explicit release, otherwise the resources of the OXC are immediately free after all the burst have passed trough it. The next section focuses on the most relevant one-way reservation protocols as above-mentioned.

3.2 Existent One-way Resource Reservation Protocols

There are six most relevant resource reservation protocols. In order of appearance they are: HORIZON in 1997 [10], JET in 1997 [11], JIT in 2000 [12], JUMPSTART in 2002 [13], JIT^{+} in 2004 [14] and E-JIT in 2006 [6].

As above-mentioned, the main goal of this dissertation is to present and evaluate a new resource reservation protocol based on JIT^{+}, named E-JIT^{+}. For this reason in this section the above-referred protocols are described with particular emphasis on JIT^{+}. In the other hand JIT^{+} and E-JIT^{+} are both based on JIT, so this last protocol is the first to describe.

To describe the details of these different protocols, a set of important parameters and the respective notation are adopted in this study[6]:

- $T_{Setup}(X)$ -Is the amount of time needed to process the setup message; X is aresource reservation protocol. This time can be different for each protocol, but is the same in every node in the network because all of them use the same protocol.

- T_{OXC} - Is the amount of time needed by the OXC to configure his switch fabric in order to establish a connection between an input port and an output port.

- $T_{Offset}(X)$- Is the interval of time at the source node between the transmission of the first bit of the setup message and the transmission of the first bit of the correspondent data burst. This value should be defined so that the first bit of the data burst arrives at the egress node shortly after this node is configured and ready to accept the burst. This minimum time is given by the following expression, where K is the number of nodes in the path of a burst between the source and the destination node[14]:

$$T_{Offset}^{(min)}(X) = K \times T_{Setup}(X) + T_{OXC} \tag{3.1}$$

- T_{Idle}-During this period of time the OXC is ready to accept a burst, but is still waiting its arrival.

- T_{void}- Is the time during which an output channel is free and ready to be configured in order to switch a burst.

In the following section a brief description of the existent resource reservation protocols is presented using the above parameters.

3.2.1 JIT (Just-In-Time)

JIT or Just-in-Time resource reservation protocol was proposed in December 2000 by Wei and McFarland [12]. This is a very simplistic protocol with immediate resources reservation, i.e. as soon as a setup message arrives at the node, it processes this message and an output channel is reserved for the correspondent data burst. However, if none channel is available at that moment the reserve can not be made, the setup message is rejected and the corresponding data burst is lost.

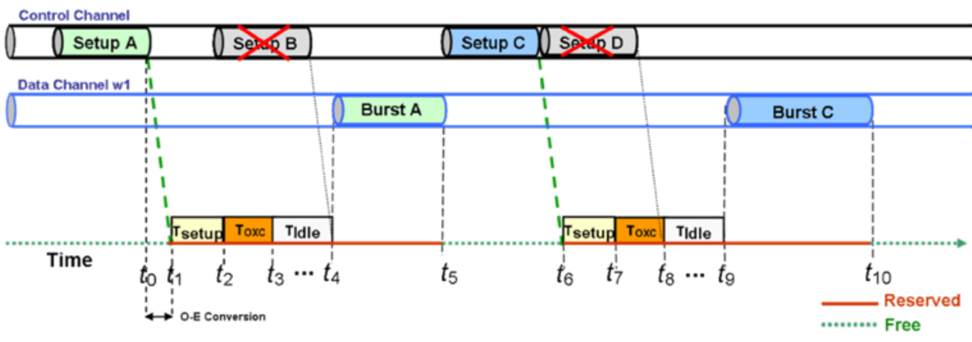


Figure 3.1: Operation of JIT resource reservation protocol

Figure 3.1 represents the activity of an OBS node. In this figure, a single output wavelength is represented. This wavelength, or channel, can be in two states: free or reserved.

As may be seen at the instant t_0 a setup message arrives at the node and at that moment the output channel is free, so the setup message is accepted. Between the instant t_0 and t_1 the setup message is converted from optical to electronic signal and between t_1 and t_2 is processed. After the instant t_2 until t_3 the OXC is configured to switch the data burst.

Between t_3 and t_4 the node is in an idle state waiting for the arrival of burst A, which occurs at time t_4. At time t_5 the last bit of burst A arrives and the channel becomes to free again ready to accept a new setup message. The length of the reserved period is equal to the burst length plus the corresponding offset time and the time of the free status is equal to the time until the arrival of the next setup message. Figure 3.1 also illustrates two burst rejections, at instant t_4 setup message B arrives but is rejected because the channel is already reserved to switch burst A. Then at instant t_8 setup message D is rejected because the channel is reserved to switch burst C. As described the JIT operation is very simple, it performs a first-come, first-served (FCFS) service because bursts are served in the order that their corresponding setup messages arrive at the node[12].

3.2.2 Horizon

Horizon is a resource reservation protocol proposed by Turner in 1997, which uses delayed reservation scheme[10]. This means that an output wavelength is reserved for a burst just before the arrival at the switch of the first bit of that burst.

On the other hand when a setup message arrives, if it is determined that when the burst arrives no wavelength can be reserved when required, then the setup message is rejected and the correspondent data burst is lost.

This protocol introduces an important concept called "time horizon" also used in JIT$^+$, JET and the new E-JIT$^+$. The time horizon is defined as "the earliest time to which there is no prevision to use the channel (wavelength)" [12]. Other definition is "the earliest time (deadline) in which a channel is ready to accept new bursts"[6].

The time horizon value is calculated using the burst length and the offset time, both provided by the setup message.

Unlike immediate reservation protocols, delayed reservation protocols allow that multiple setup messages are accepted and the respective wavelength reservation are scheduled, however the associated bursts can not overlap in time.

In horizon a channel is reserved for a burst if and only if that burst arrives after all the bursts scheduled for that channel have been sent[10].

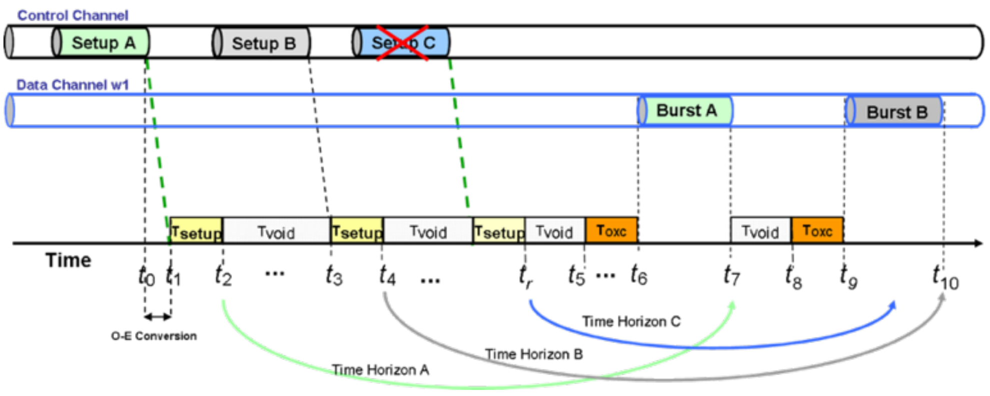


Figure 3.2: Operation of Horizon resource reservation protocol

Figure 3.2 shows an operation scheme of the horizon resource reservation protocol. As may be seen Burst A and Burst B are accepted and a third burst, Burst C is rejected. After processing the setup message A the OXC calculates the instant it needs to start the configuration of the switch fabric to switch burst A. This calculation is based in the burst duration ($t_7 - t_6$) and the TOXC ($t_6 - t_5$). In this case as may be seen in figure 3.2 the instant to start the configuration for Burst A is t5.

Burst B is accepted because its setup message arrives at time t_3, which is before t_7 (time horizon A) and after t_2 (T_{Setup} of message A), otherwise it will be rejected.

Figure 3.2 , shows also setup message C being rejected. As may be seen, setup message C is rejected because the time horizon of burst C is between t_9 and t_{10}, which is before the time horizon of burst B.

JET (Just- Enough-Time)

JET is a delayed resource reservation protocol proposed by Qiao in 1997[11], which performs void filling. Unlike horizon if the channel is in an idle state and the last bit of a burst arrives before the end of the idle interval then this burst can be accepted. Hence, in JET an output channel is reserved for a burst if one of the following situations occurs:

- The arrival of the burst is after the time horizon of that channel;
- The arrival of the burst happens during an idle state of the channel and the last bit of the burst arrives before the end of the idle state interval;

If none of the conditions are satisfied for any channel, then the setup message is rejected and the associated burst is dropped.

In order to evaluate if a reservation can be made the setup message as to carry specific information like the burst length, the burst offset and obviously the source and destination nodes.

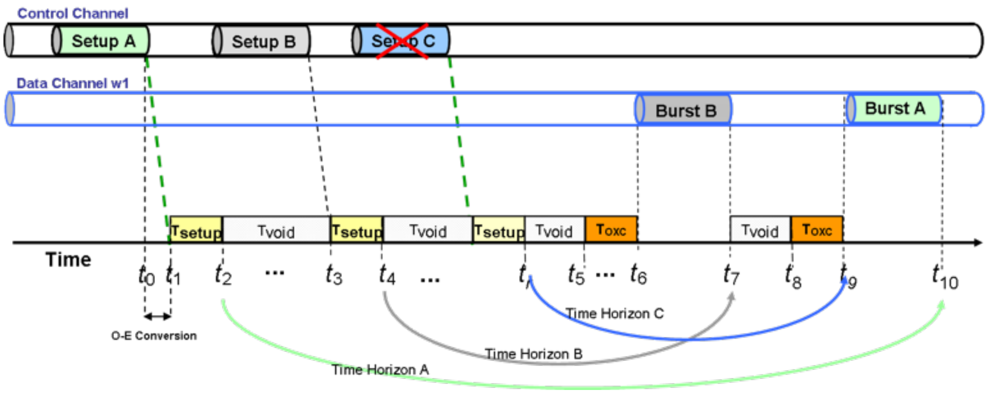


Figure 3.3: Operation of JET resource reservation protocol

Figure 3.3 represents the operation of JET reservation protocol. As may be seen burst A and burst B are successfully switched and burst B is rejected. Although the setup message B arrives after the setup message A ($t_1 < t_3$), burst B arrives before Burst A ($t_6 < t_9$).

Hence, JET performs void filling because burst B is accepted and the associated time horizon (t_7) is before the instant arrival of the first bit of burst A (t_9) and the channel is in an idle state waiting for burst A.

Figure 3.3 illustrates also a burst rejection. As may be seen, setup message C arrives after setup message B and the time horizon C is t_9, but at this instant the channel is reserved to switch burst A. In this situation if none other channel is free, then setup message C is rejected and the corresponding data burst is lost.

Many studies and simulations show that JET has good results reducing the burst lost probability thanks to void filling.

3.2.3 JIT$^+$

JIT$^+$ is a resource reservation protocol proposed by Teng and Rouskas, which is published in [14]. It is an immediate resource reservation protocol, which uses the time horizon concept combined with the simplicity of the JIT protocol. JIT$^+$ has the ability to perform limited burst scheduling (maximum of two bursts per channel).

In JIT$^+$ an output channel is reserved for a burst if one of the following situations occurs:

- The burst arrives after the time horizon for that channel;
- The data channel has at most one reservation scheduled;

The authors consider that JIT$^+$ maintains all the simplicity of JIT and the possibility to schedule two burst reservations contributes to reduce the burst loss probability[14].

Figure 3.4 represents the operation scheme of JIT$^+$ using the data channel $w1$ for a given node. As may be seen, burst A and burst B are accepted but burst C is rejected.

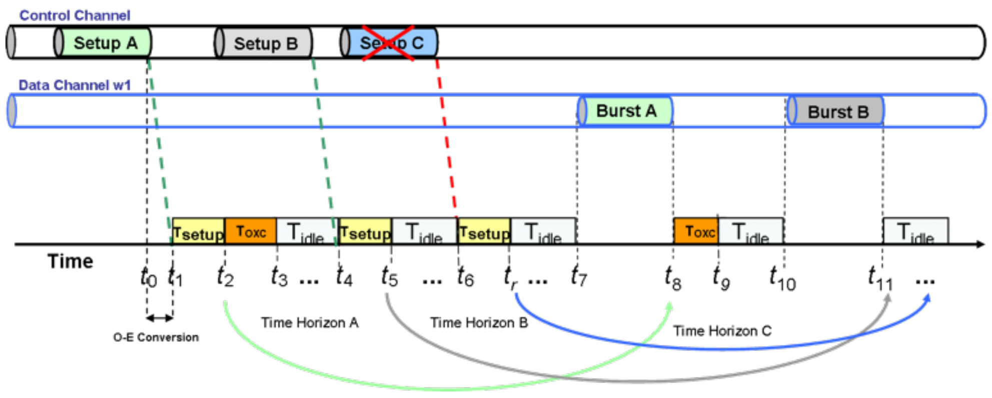


Figure 3.4: Operation of JIT$^+$ resource reservation protocol

When the setup message A arrives the channel $w1$ is free, so this message is accepted and the switch is configured immediately. After some time at instant $t_4$setup message B arrives, as the correspondent time horizon is the instant t_{11} and the channel have only one reservation (Burst A) then this message is accepted and the reservation is scheduled.

At time t_6 another setup message arrives and the correspondent time horizon is greater than the time horizon of the channel (t_{11}), but there are already two burst reservations scheduled, so this setup message is rejected. Using Horizon or JET burst C would be accepted because they have no limit in the number of scheduled burst reservations permitted.

Like the JIT protocol, JIT$^+$ performs FCFS (First-Come-First-Served) service, because it does not use void filling, i.e. resources are reserved for bursts in the same order that they arrive at the node[14].

3.2.4 E-JIT (Enhanced JIT)

E-JIT is the most recent resource reservation protocol proposed by Joel Rodrigues, presented in [6]. To develop this resource reservation protocol the author has made several comparative studies in order to evaluate the performance assessment of JIT, JumpStart, JIT$^+$, JET and Horizon. He has concluded that the difference between performances of the existent protocols is very small, so he tried to improve and optimize the JIT protocol because is the simplest to implement. This improves are made with the intention to keep the advantages of its simplicity in terms of implementation.

The main goal of E-JIT is to reduce the period of time during which the data channel remains in "reserved" status and take advantage of it to reduce the burst loss probability.

In E-JIT a data channel is reserved for a burst if one of two following situations occurs[6]:

- The data channel is free;
- If the data channel is reserved, the end time of the last switched burst is before than the end time needed to process the actual incoming setup message ($\leq T_{Setup}$);

If none of the above situations is verified then the setup message is rejected and the corresponding data burst is lost.

Like the JIT protocol E-JIT is an immediate resource reservation protocol, which means that once a setup message is accepted and processed an output channel is immediately reserved for the corresponding burst.

In order to release the switch fabric resources, E-JIT uses estimate (or implicit) release which means that when a burst leaves an OXC the previously reserved resources for that burst are automatically updated to free without the need to wait a release message.

Figure 3.5 demonstrates E-JIT operation at one node, using the output channel $w1$. In this figure two bursts (Burst A and Burst B) are accepted.

In the following paragraphs the main procedures adopted by E-JIT are described.

As may be seen in figure 3.5 when the setup message A arrives at the node the channel $w1$ is free, so this message is accepted and the channel state changes to reserved. The OXC is immediately configured to switch burst A, which arrives at time t_4, and the last bit of this burst arrives at time t_6. Before the end of burst A, a new setup message arrives at time t_5.

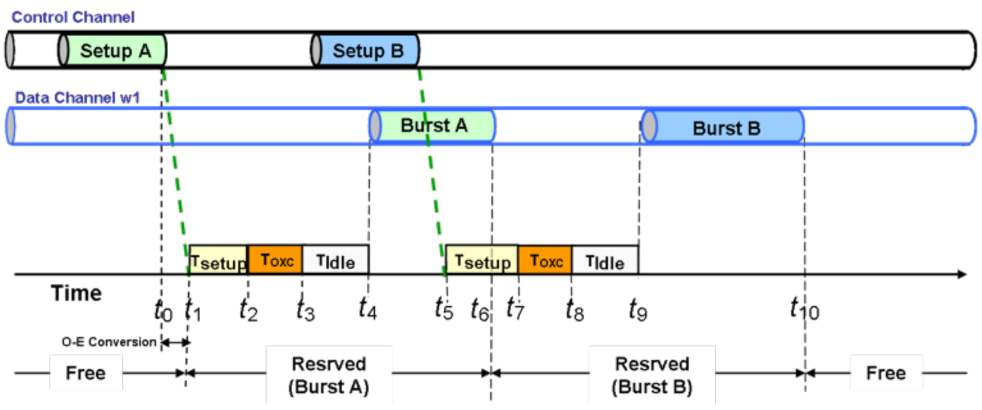


Figure 3.5: Operation of E-JIT resource reservation protocol.

As the end time of burst A occurs at time t_6 and the time needed to process this message (T_{Setup} B) is greater than the end of burst A ($t_7 > t_6$), setup message B is processed and burst B is accepted. In this situation the channel remains in reserved state because when burst A ends it is immediately reserved to switch burst B.

In the JIT protocol setup message B would be automatically rejected because at instant t_5 the channel $w1$ is reserved for burst A. In figure 3.6 is represented a burst rejection using the E-JIT protocol.

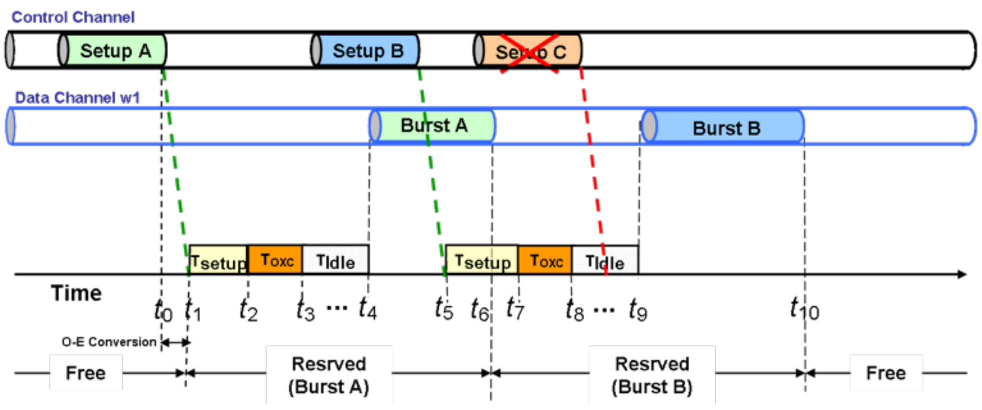


Figure 3.6: Operation of E-JIT protocol (rejecting a burst).

This figure illustrates a situation similar to previous, adding a new setup message, setup message C, which arrives between the time t_8 and t_9. As may be seen in this time interval $[t_8, t_9]$, the channel $w1$ is reserved to burst B, which arrives at time t_9. Moreover the end time of burst B (t_{10}) is greater than the end time needed to process setup message C, so for E-JIT none of the conditions needed to accept the burst is satisfied, which means that setup message C is rejected and the corresponding burst (Burst B) is lost.

Like the JIT resource reservation protocol E-JIT uses a FCFS (First-Come-First-Served) service, because bursts are switched in the order that their setup messages arrive at the node. Moreover, E-JIT chooses the first available data channel to switch a burst.

3.3 A New Resource Reservation Protocol: E-JIT$^+$ (Enhanced JIT$^+$)

In this section a new resource reservation protocol for Optical Burst Switched Networks is presented, called E-JIT$^+$. This new protocol is based on the existent resource reservation protocol JIT$^+$, explained in section 3.2.3.

As above-mentioned, the most recent resource reservation protocol E-JIT is based on JIT and as demonstrated by the author it performs better then JIT[6]. Therefore, these facts encourage us to develop a new resource reservation protocol based in the existent protocol JIT$^+$.

The JIT$^+$ protocol was developed based on JIT, with the goal of reducing burst lost probability keeping its simplicity in terms of implementation. Therefore, several comparative studies demonstrate that JIT$^+$ optimizes JIT reducing its burst lost probability[6, 14].

As mentioned in section 3.2.3, JIT$^+$ performs limited burst scheduling because it allows scheduling a maximum of two bursts per channel.

However, the possibility of an unlimited burst scheduling, has not been explored until nowadays. Thus, in this study the effects of adding unlimited burst scheduling to JIT$^+$ is analyzed, proposing a new resource reservation protocol called E-JIT$^+$.

E-JIT$^+$ is an one-way resource reservation protocol, which assumes an out of band signalling and the signalling channel is best-effort link by link.

Like JIT$^+$, E-JIT$^+$ can support both short and long bursts.

Like other one-way resource reservation protocols such as E-JIT[6, 15], E-JIT$^+$ implements a set o functions. These functions are:

- Session Declaration - Announces a new connection to the network;
- Path Setup - Configures the multiple resources required to establish an all optic path from the source node to destination node;
- Data Transmission - Informs intermediate nodes of burst arrival time and the respective length;
- State Maintenance - Refresh the state information which keeps the connection alive;
- Path Release - Releases the resources previously reserved to transmit the burst;

Sessiondeclaration,*Pathsetup*, and *Datatransmission* are functions performed by the setup message informing intermediate modes for a burst arrival.

The $StateMaintenance$ function is used to maintain a connection active especially for long bursts preventing the switch state from timing out. To perform this function the source node sends KEEP_ALIVE messages for the intermediate nodes along the path[15].

A setup message contains information about the burst offset and burst length allowing the calculation of the time horizon.

In order to set free the switch fabric resources, E-JIT$^+$ uses estimated release, so the source node does not need to send a release message when a burst arrives at the destination. This means that a data channel $w1$ is automatically updated to free when the last bit of the burst arrives at the switch. Hence, the representation of an implicit "release message" can be ignored.

However if a network element detects a "setup" failure it sends a release message to all the network elements along the path in order to update the status to "free" allowing to switch a new burst.

In figure 3.7, a successful burst transmission across a path in an OBS network is represented. As may be seen, the burst is transmitted from the Edge Node A to Edge Node B, passing through an ingress core node, two intermediate nodes and an egress core node.

The burst is transmitted with an initial offset time calculated using the same equation as the E-JIT resource reservation protocol, which is presented in equation (3.1) [6].

In order to switch the burst each switch fabric has to be configured as soon as possible before the burst arrival. Otherwise the burst would be lost.

In this scheme the representation of the release messages is ignored, because E-JIT$^+$ uses estimated (or implicit) release and in the situation represented by figure 3.7 all the resources are reserved with success, so it does not occur any setup failure.

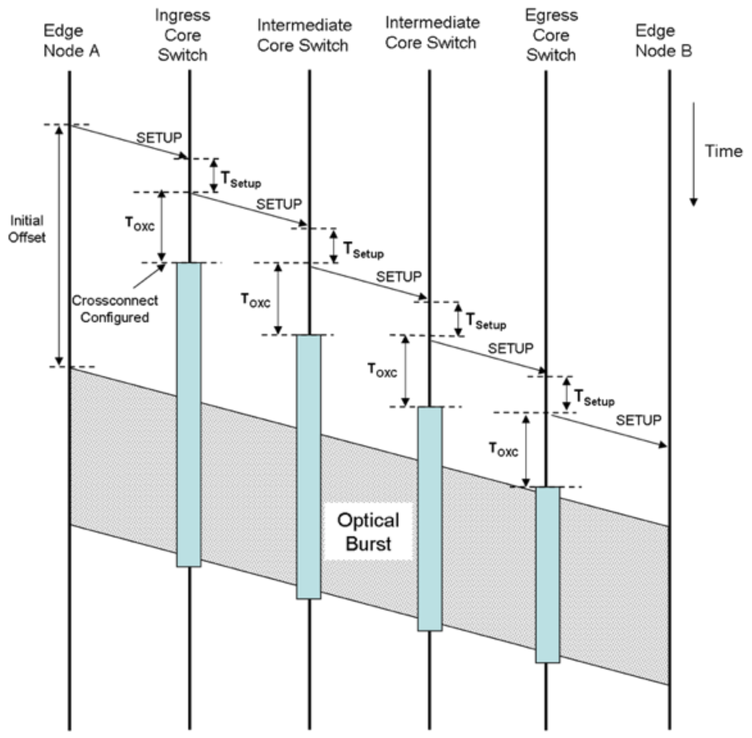


Figure 3.7: Burst transmission across an OBS network path.

3.3.1 E-JIT$^+$ System Architecture

To describe E-JIT$^+$, the description provided for E-JIT specified in [6, 15] is followed. E-JIT$^+$ architecture has two different layers. The upper layer called Transport Layer and the base layer called JIT Layer.

Figure 3.8 illustrates the traffic flow and the relationships between the different protocol entities.

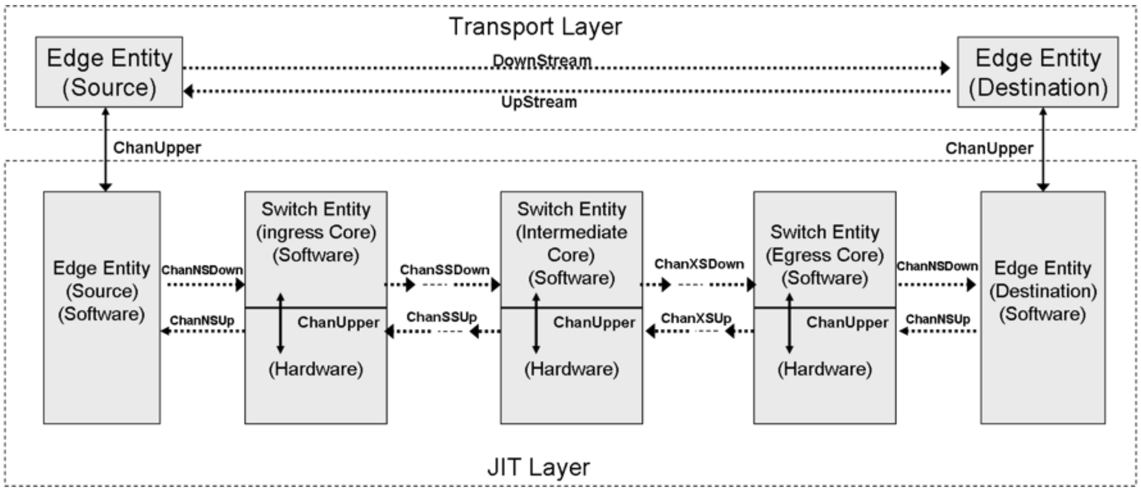

Transport Layer
Edge Entity (Source)
DownStream
UpStream
Edge Entity (Destination)
ChanUpper
ChanUpper
Edge Entity (Source) (Software)
ChanNSDown
ChanNSUp
Switch Entity (ingress Core) (Software)
ChanUpper
(Hardware)
ChanSSDown
ChanSSUp
Switch Entity (Intermediate Core) (Software)
ChanUpper
(Hardware)
ChanXSDown
ChanXSUp
Switch Entity (Egress Core) (Software)
ChanUpper
(Hardware)
ChanNSDown
ChanNSUp
Edge Entity (Destination) (Software)
JIT Layer

Figure 3.8: Representation of protocol stack architecture.

In the Transport Layer, *DownStream* represents the traffic flow from the source edge node to the destination edge node and *UpStream* represents the traffic flow from the destination edge node to the source edge node.

At the Transport Layer the source edge node starts a transition sending an $OPEN$ message through a channel called *ChanUpper*. As soon as the source edge node at the JIT Layer receives the $OPEN$ message, it generates a $SETUP$ message and transmits it through the $ChanNsDown$. $ChanNSUp$ is used to transmit messages from the ingress node and the source node.

Messages from the ingress node to an intermediate node are transmitted through the channel $ChanSSDown$ and messages from an intermediate node to the ingress node are transmitted through $ChanSSUp$. An Intermediate node sends messages to the egress node through $ChanXSDown$ and the inverse through the channel $ChanXSUp$.

Finally, the destination node uses $ChanNSUp$ to communicate with the egress node and $ChanNSDown$ is used by the egress node to communicate with the destination node[6, 15].

In the next section, the E-JIT$^+$ protocol operation is presented using illustrative schemes.

3.3.2 E-JIT$^+$ Protocol Operation

In this section the E-JIT$^+$ protocol operation mode is presented. E-JIT$^+$ intends to be an OBS one-way resource reservation protocol, which uses immediate reservation. This new protocol is based in the existent JIT$^+$ protocol. Moreover E-JIT$^+$ intends to maintain JIT$^+$ simplicity and consequently maintain JIT simplicity in terms of implementation[15].

Under the E-JIT$^+$ protocol an output data channel is reserved for a burst if the arrival time of the burst is later then the time horizon of that data channel.

If this condition is not satisfied then the setup message is rejected and the correspondent data burst is lost. To understand the E-JIT$^+$ operation, the time horizon concept has to be remembered. As mentioned in section 3.2.2, the time horizon is "the earliest time in which a data channel is ready to accept new burst". It is possible to calculate the time horizon value because, like JIT$^+$, under E-JIT$^+$ setup messages contain both the offset time and the burst length values.

In the following paragraphs, some figures to represent the E-JIT$^+$ operation are used. These figures are relative to an OBS node assuming that only a single data channel $w1$ is present[6]. The following parameters used in the previously schemes are used:

- T_{Setup} - The time during which a setup message is processed;
- T_{OXC} - Is the time that OXC needs to configure its switch fabric in order to correctly switch a burst;
- T_{idle} - Period of time during which the OXC is already configured but its still waiting the arrival of the burst.
- Time Horizon - Is the earliest time in which a channel is ready to accept new bursts.

Figure 3.9 shows a scheme in which three successive bursts arrive at a node. When the first setup message (Setup A) arrive at the instant t_0 the output data channel does not have any scheduled reservation, so obviously the time horizon of burst A (t_8) is greater than the time horizon of the channel. Hence this message is automatically accepted and the OXC configures its switch fabric between the instant t_2 and t_3. After t_3 the node is in an idle state waiting for another setup message.

As may be seen a second setup message (setup B) arrives at the node at the instant t_4. At this instant the channel has one scheduled burst reservation (for burst A), and the time horizon for burst B is t_{12}. As the time horizon of this setup message (t_{12}) is greater than the time horizon of the channel (t_8), this setup message is accepted and the burst reservation is scheduled.

At the instant t_6 burst A arrives at the node and the last bit of this burst arrives at the instant t_8. As may be seen, another setup message (setup C) arrives at the instant t_7 and the time horizon of burst C (t_{15}) is greater than the

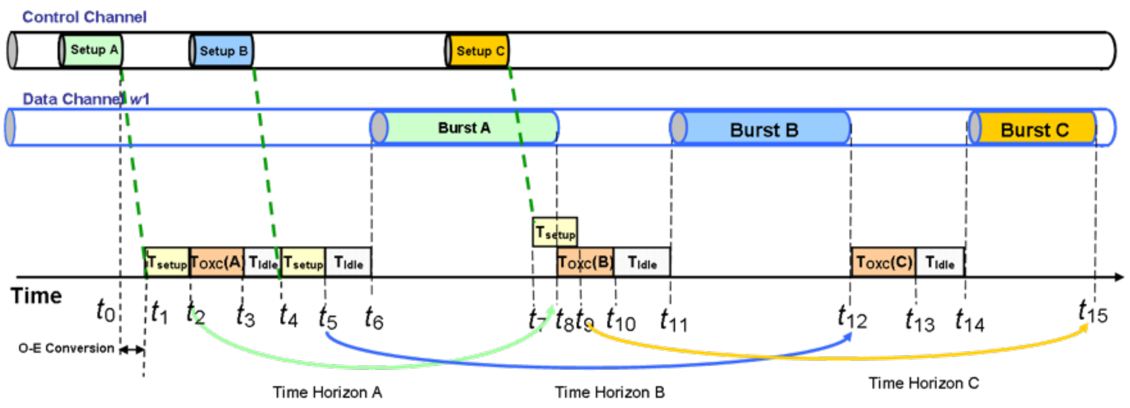


Figure 3.9: Operation of E-JIT$^+$ protocol.

time horizon of the channel (t_{12}), so the setup message C is accepted and the channel reservation is scheduled.

Using the JIT$^+$ protocol this setup message (Setup C) would be rejected because the channel has already two burst reservations i.e. is reserved for burst A which is being switched at that moment and has a scheduled reservation for burst B.

As may be seen, after t_8 the switch fabric is immediately configured to switch burst B, because E-JIT$^+$ is an OBS resource reservation protocol with immediate resource reservation.

The same situation occurs at the instant t_{12}, i.e. the switch fabric is configured to switch burst C as soon as possible and after the instant t_{13} the switch stays in an idle state waiting for the arrival of burst C.

Figure 3.10 represents another instance of last figure with the addition of a new setup message (Setup D), which arrives at the instant t_{11}. At this instant the channel has one pending burst reservation for burst C and is switching Burst B. However, as the time horizon of burst D is t_{16}, which is greater than the time horizon of the channel (t_{15}), then this setup message can be accepted and the burst reservation is scheduled.

Another instance of the E-JIT$^+$ protocol operation showing a burst rejection is represented in figure 3.11.

In this figure setup messages A and B arrive at the node in the same circumstances than in the previous images, so they are accepted by the same reasons mentioned above. At the instant t_7 a new setup message (Setup C) arrives at the node. The time horizon of burst C (t_{12}) is less than the time horizon of the channel (t_{13}), so because of that, setup message C is rejected and the correspondent data burst is lost.

Like JIT$^+$ resource reservation protocol, E-JIT$^+$ uses FCFS service because bursts are scheduled in the order that their setup messages arrive at the node. Moreover, like JIT$^+$, the E-JIT$^+$ protocol use the first available data channel to switch an incoming burst.

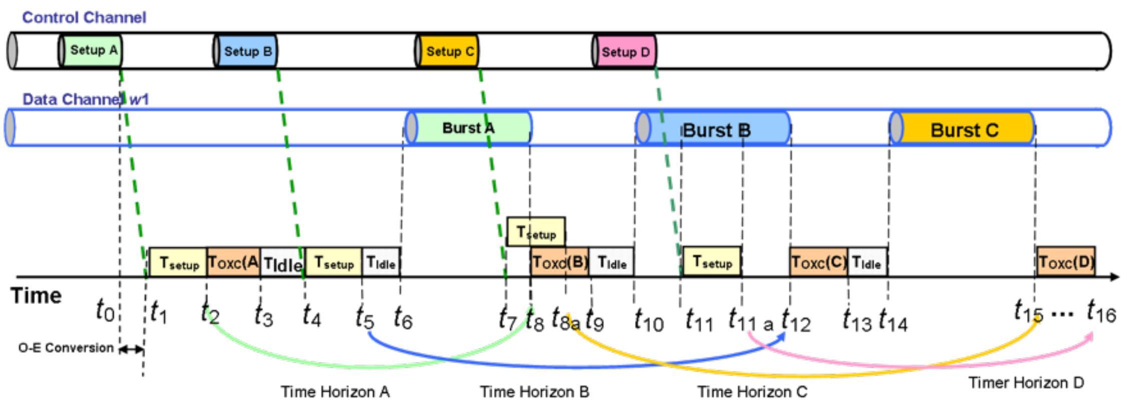


Figure 3.10: Operation of E-JIT$^+$ protocol.

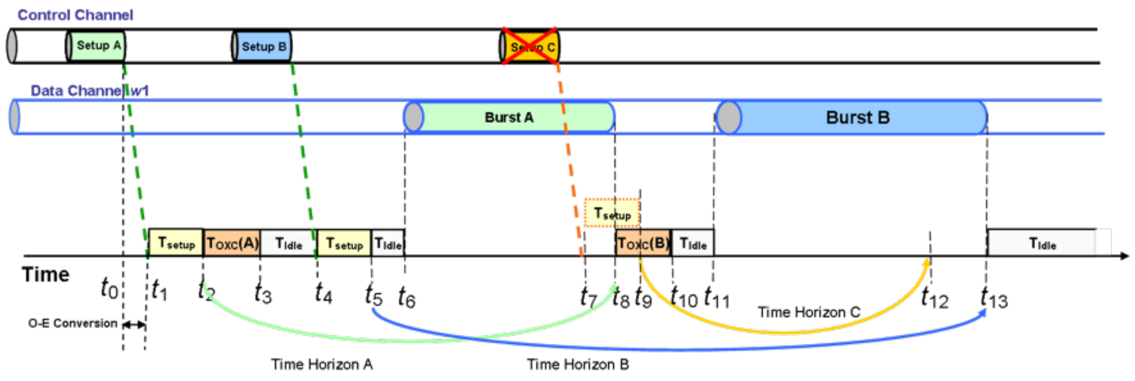


Figure 3.11: Operation of E-JIT$^+$ protocol (rejecting a burst).

3.3.3 E-JIT$^+$ Formal Specification

To present a formal specification of the E-JIT$^+$ resource reservation protocol, the formal specification proposed in [6] for E-JIT is followed.

This formal specification uses the extended finite state machine ($EFSM$) because this is a good means to describe a communication protocol [6, 15].

An EFSM is formally represented using an eight-tuple $(\sum, S, s, V, E, T, A, \delta)$, where:

- $\sum$ - is the set of messages that can be sent or received;
- S - is the set o states;
- s -is the initial state;
- V- set of variables;
- E- set of predicates that operates on variables;
- T- set of timers;
- A - set of actions that can operate on variables;
- δ - set of state transition functions.

Each state transition function (δ) is represented by the following expression:

$$\sum * E(V) * T \rightarrow \sum * A(V) * S$$

There are two types of state transition functions:

1. *spontaneous* - Does not have a an input event on its condition part.
2. "*when*" - Have an input event satisfying the T condition.

To represent a transition from state S_1 to state S_2 where S_1 is the previous state and S_2 is the following state, the below expression is used:

$$S_1 \rightarrow S_2$$

A transition is executed when an input event is available and a condition is true. Each transition can be represented as a fraction in which the nominator is a condition having an input event and a Boolean expression. In the denominator is the action part that can be an output event or a statement operating on variables.

In a transition $?chan.m$ denotes an input message from a given channel carrying the message m and $!chan.m$ denotes an output message to the specified channel carrying the message m. Finally $Settimer(T, C)$ represents an action that sets the timer T to the value expressed by C.

In the following expressions $T1$ represents a spontaneous transition and $T2$ represents a "when" transition:

$$T1 : \frac{}{\begin{array}{c} var1 := FALSE \\ Settimer(T1, Const) \end{array}} \qquad T2 : \frac{?chanA1.Continue}{\begin{array}{c} Settimer(T1, Const) \\ !chanD1.Stop \end{array}}$$

In the next paragraphs, according to the $EFSM$ model, the state diagrams for the source edge node, the destination edge node and intermediate core node are defined.

Source Edge Node State Machine

The set of messages is the first to define [6, 15]:

$$\sum = \{Open, Setup, Failure, Timeout, Connection\_Failure, Close, Release, \\ Clear\_to\_Send, Transmission\_Complete, Keepalive\}$$

- Open - Generated by the Transport Layer to notify the JIT Layer of an incoming burst;
- Failure - Used by any node to notify an error;
- Timeout - Used for each timer defined within the Timeout message;
- Connection_Failure - Used to notify upper layers of an occurred error during the connection phase;
- Close - Used to release the connection (at the Software Layer);
- Release - Message used to set free the switch resources along the path (at the Hardware Layer);
- Clear_to_Send - Message used to notify the Transport Layer that the setup phase is complete and the corresponding burst can be sent;
- Transmission_Complete - Message used to notify the Upper Layer that a transmission of a burst has been completed successfully;
- Keepalive - Used to maintain the connection when long bursts are transmitted.

Below, the set of states is defined:

$$S = \{IDLE, SETUP\_PROCEEDING, DATA\_TRANSMISSION\}$$

The state machine starts by waiting in the *IDLE* (Initial state) state for an *Open* message notifying an incoming of a burst. After the arrival of this message the machine goes to the state *SETUP_PROCEEDING* and stays on it during the time specified by *Burst_Delay*.

When the *Setup_Timer* times out indicating the beginning of a data burst, the machine goes to the *DATA_TRANSMISSION* state. When the *Burst_Timer* times out the data burst has ended, the connection is closed and the state machine return to *IDLE* state, waiting a new request. The set of variables is:
$V = \{Burst\_Delay, Burst\_Time, KA\_Time\}$

- Burst_Delay - Delay at the source node before sending the burst (initial offset time);
- Burst_Time - The duration of the burst (burst length);
- KA_Time - The keepalive timer set up to send Keepalive messages.

Below the set of timers and the set of actions:

$$T = \{Setup\_Timer, Burst\_Timer, KA\_Timer\}$$

- *Setup_Timer*- The timer that controls the time to process a setup message (T_{Setup});
- *Burst_Timer*- Timer that controls the *Burst_Time*;
- KA_Timer- Timer that controls the *KA_Time*.

$$A = \{Settimer, Update\}$$

Figure 3.12 represents the state diagram for the source edge node.

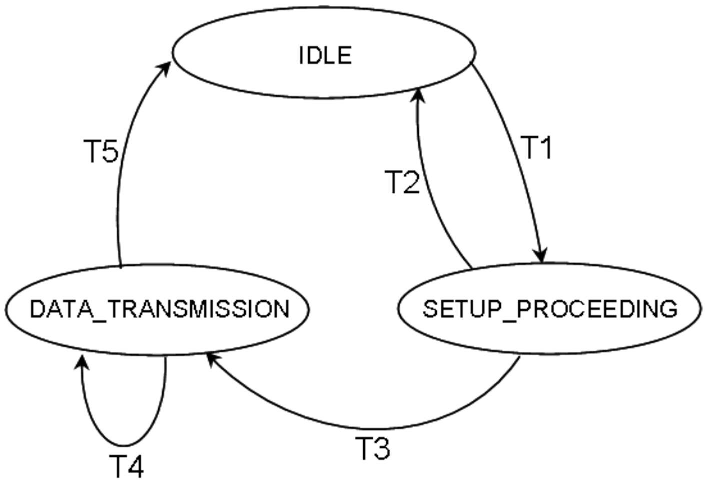


Figure 3.12: State diagram of source edge node.

As may be seen in this diagram the state machine waits in the $IDLE$ state until it receives an *Open* message from the Upper Layer. When this message arrives the source edge node generates a Setup message containing the *Burst_Delay* and *Burst_Time* variables, which are used to set the *Burst_Timer*. The function *Update* is used to update the *Burst_Delay* at each hope subtracting the processing time from the *Burst_Delay*.

After receive the *Open* message the state changes to $SETUP\_PROCEEDING$ (T_1) and the *Setup_Timer* is set. If a *Close* message is received from the Upper Layer or a *Failure* message from the ingress core node, the state changes to $IDLE$ again (T_2). Otherwise it waits until *Setup_Timer* times out and changes to $DATA\_TRANSMISSION$ state (T_3).

At the $DATA\_TRANSMISSION$ state the machine can goes to $IDLE$ if it receives a *Close* message from the Upper Layer or a *Failure* message from the Ingress Core Node. When *KA_Time*r times out the machine stays in the same state, resets the *KA_Timer* and sends *Keepalive* messages to maintain the connection.

Finally when the *Burst_Timer* times out the connection is closed and the machine goes to the $IDLE$ state waiting the arrival of a new *Open* message. Below, the expressions of each transition are defined:

$$IDLE \rightarrow SETUP\_PROCEEDING(T1)$$

$$T1 : \frac{?ChanUpper.Open}{\begin{array}{c}!ChanNsDown.Setup(Burst\_Time, Burst\_Delay)\\ Settimer(Setup\_Timer, Burst\_Delay)\end{array}}$$

$$SETUP\_PROCEEDING \rightarrow IDLE(T2)$$

$$T2 : \frac{?ChanUpper.Close}{!ChanNsDown.Release}$$

$$T2 : \frac{?ChanUpper.Failure}{!ChanUpper.Connection\_Failure}$$

$$SETUP\_PROCEEDING \rightarrow DATA\_TRANSMISSION(T3)$$

$$T3\ :\ \frac{?ChanT1.Timeout(Setup\_Timer)}{\begin{array}{l} if(Burst\_Time = Specified) \\ \quad \{Settimer(Burst\_Timer, Burst\_Time)\} \\ Settimer(KA\_Timer, KA\_Time) \\ !ChanUpper.Clear\_To\_Send \end{array}}$$

$$DATA\_TRANSMISSION \rightarrow DATA\_TRANSMISSION(T4)$$

$$T4\ :\ \frac{?ChanT1.Timeout(KA\_Timer)}{\begin{array}{c} Settimer(KA\_Timer, KA\_Time) \\ !ChanNSDown.Keepalive \end{array}}$$

$$DATA\_TRANSMISSION \rightarrow IDLE(T5)$$

$$T5\ :\ \frac{?ChanUpper.Close}{!ChanNSDown.Release}$$

$$T5\ :\ \frac{?ChanNSUp.Failure}{!ChanUpper.Connection\_Failure}$$

$$T5\ :\ \frac{?ChanT1.Timeout(Burst\_Timer)}{!ChanUpper.Transmission\_Complete}$$

Completed the definition of the state machine for the source edge node, next the state machine related to the destination edge node is defined.

Destination Edge Node State Machine

The set of messages for the Destination Edge Node is[6]:

$$\sum = \{Open, Setup, Failure, Timeout, Setup\_Complete, \\ Close, Release, Transmission\_Complete, Keepalive\}$$

Almost all this messages are the same used by the source edge node with the exception of *Setup_Complete* which is used by the destination edge node to notify the Upper Layer that the transmission of a burst as been completed with success. The set of states for destination edge node is:

$$S = \{IDLE, SETUP\_PROCEEDING, DATA\_TRANSMISSION\}$$

All this states used in the destination edge node state machine are the same used in the source edge node state machine and where explained above.The initial state is *IDLE*. The set of variables is the following:

$$V = \{Burst\_Delay, Burst\_Time, KA\_Time\}$$

These variables where also explained in the source edge node state machine. Finally, the set of actions is:

$$A = \{Settimer, Update\}$$

In figure 3.13 is represented the Destination Edge Node state machine.

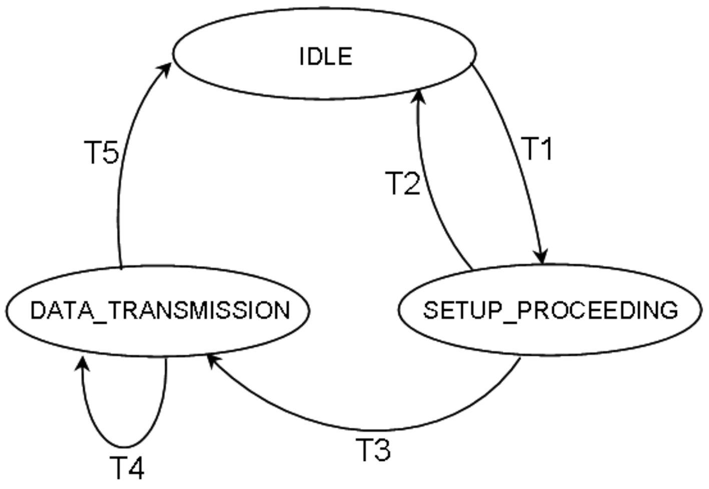


Figure 3.13: State diagram of destination edge node.

The Destination Edge Node State Machine starts in the *IDLE* state (initial state) and stays there until a *Setup* message arrives. When this *Setup* message arrives the state is changed to *SETUP_PROCEEDING* and the JIT Layer sends an *Open* message to the Upper Layer.

If the Upper Layer answers with a *Close* message, the Destination Edge Node sends a *Failure* message to the source node. Otherwise, it updates the *Burst_Delay*, sets the *Burst_Timer* and *KA_Timer* and goes to the *DATA_TRANSMISSION* state. In this state *Keepalive* messages can be received resetting the *KA_Timer* to maintain the connection active.

In the *DATA_TRANSMISSION* state if *Burst_Timer* or*KA_Timer* times out or if the edge node receives a *Release* message, or if it receives a *Close* message from the Upper Layer, then the state machine goes to the *IDLE* state. If the Destination Edge Node receives a *Close* message it sends a *Failure* message forcing the connection teardown. The expressions of each transition are:

$$IDLE \rightarrow SETUP\_PROCEEDING(T1)$$

$$T1\ :\ \frac{?ChanNSDown.Setup(Burst\_Time, Burst\_Delay)}{!ChanUpper.Open}$$

$$SETUP\_PROCEEDING \rightarrow IDLE(T2)$$

$$T2\ :\ \frac{?ChanUpper.Close}{!ChanNSUp.Failure}$$

$$SETUP\_PROCEEDING \rightarrow DATA\_TRANSMISSION(T3)$$

$$T3\ :\ \frac{?ChanUpper.Setup\_Complete}{\begin{array}{l} if(Burst\_Time = Specified) \\ \quad \{Settimer(Burst\_Timer, Burst\_Time + Burst\_Delay)\} \\ Settimer(KA\_Timer, KA\_Time) \end{array}}$$

$$DATA\_TRANSMISSION \rightarrow DATA\_TRANSMISSION(T4)$$

$$T4\ :\ \frac{?ChanNSDown.Keepalive}{Settimer(KA\_Timer, KA\_Time)}$$

$$DATA\_TRANSMISSION \rightarrow IDLE(T5)$$

$$T5\ :\ \frac{?ChanT1.Timeout(Burst\_Timer)}{!ChanUpper.Transmission\_Complete}$$

$$T5\ :\ \frac{?ChanT1.Timeout(KA\_Timer)}{!ChanUpper.Failure}$$

$$T5\,:\,\frac{?ChanNSDown.Failure}{!ChanUpper.Failure}$$

$$T5\,:\,\frac{?ChanUpper.Close}{!ChanNSUp.Release}$$

The Destination Edge Node state machine is complete. The next step is to define the state machine for the core node.

Core Node State Machine

A core node can receive setup messages from the source edge node, or from other core nodes[6].

When a setup message is received, it chooses the output channel to transmit the burst and configures the switch fabric as soon as possible. After this process it sends the Setup message to the next switch. The set of messages for the Core Node state machine is defined below:

$$\sum = \{Open, Setup, Failure, Timeout, Setup\_Complete,\\ Close, Release, Keepalive\}$$

The set of states is defined bellow, the $IDLE$ is the initial state:

$$S = \{IDLE, SETUP\_PROCEEDING\_ERR\_CHECK,\\ DATA\_TRANSMISSION\}$$

As may be seen, a new state appears in this set called $SETUP\_PROCEEDING\_ERR\_CHECK$. It has the same functionalities the same functionalities than the $SETUP\_PROCEEDING$ plus running the $ErrorCheck$ function that verifies if there are errors. Below is the set of variables:

$$V = \{Burst\_Delay, Burst\_Time, KA\_Time, ErrorCode\}$$

The set of timers is:

$$T = \{Burst\_Timer, KA\_Timer\}$$

The set of actions is:

$$A = \{ErrorCheck, Settimer, Update\}$$

Each intermediate core node runs a function called $ErrorCheck$ to verify if there exists some kind of errors. This function returns an error number expressed by the variable $ErrorCode$. The possible errors and respective values are:

- 0:*no_error* - Check returns without errors;
- 1:*crc_error* - Cyclic Redundancy Code;
- 2:*ime_bu_overflow* - Ingress core switch message buffer overflow;
- 3:*gme_buf_overflow* - Intermediate core switch message buffer overflow;
- 4:*signmess_state* - State machine error ;
- 5:*sigmess_oxc* - Optical Cross Connect Error;
- 6:*label_lut* - Label look-up table error.

If some error occurs then the state machine goes to the *IDLE* state, otherwise if the error code is NULL the state machine goes to *DATA_TRANSMISSION* state. To explain in more detail the core node state machine, figure 3.14 represents the respective state diagram. As may be seen the state machine waits in the *IDLE* state (initial state) until the arrival of a *Setup* message.

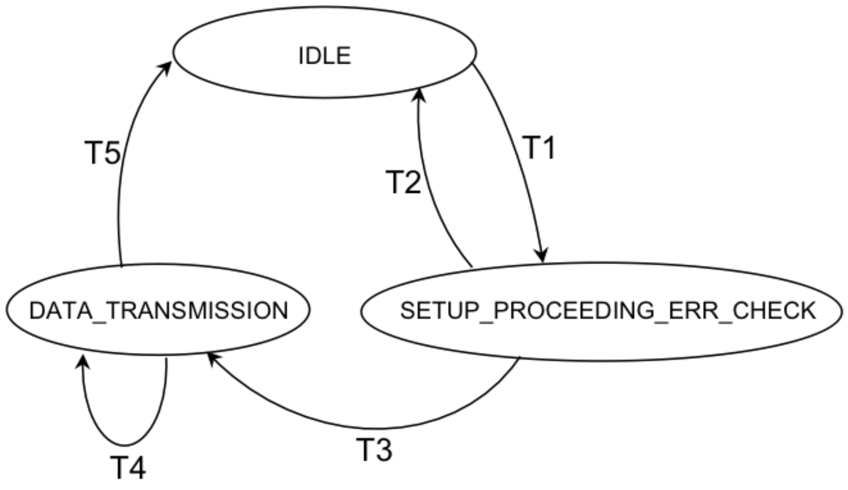


Figure 3.14: State diagram of a core node.

After receiving the setup message the state is changed to *SETUP_PROCEEDING_ERR_CHECK* and the JIT Layer sends as *Open* message to the Upper Layer and runs some checks to verify if some errors occurred. If an error is detected it sends a *Close* message to Upper Layer and a *Failure* message to the next switch.

If after receiving a setup message, the core node receives a *Release* message it sends a *Close* message to the Upper Layer and transmits the *Release* message to the next node. If no errors occur it updates the *Bust_Delay* and sets the *Burst_Timer*. After it sends the setup message to the next switch, sets the *KA_Timer* and changes to the *DATA_TRANSMISSION* state.

Once in $DATA\_TRANSMISSION$ state if the $Burst\_Timer$ times out the connection is closed and the state goes to $IDLE$ again. It can receive also $Keepalive$ messages to reset the KA_Timer to switch long bursts.

If a release message is received it sends this message to the next node and a $Close$ message to the Upper Layer returning to the $IDLE$ state. It can also receive a $Failure$ message and in this situation sends a $Close$ message to the Upper Layer and sends the $Failure$ message to the previous node. The expressions associated with each transition are:

$$IDLE \rightarrow SETUP\_PROCEEDING\_ERR\_CHECK(T1)$$

$$T1 : \frac{?ChanNSDown.Setup(Burst\_Time, Burst\_Delay)}{\begin{array}{c}ErrorCheck(ErrorCode)\\ !ChanUpper.Open\end{array}}$$

$$SETUP\_PROCEEDING\_ERR\_CHECK \rightarrow IDLE(T2)$$

$$T2 : \frac{ErrorCode}{\begin{array}{c}!ChanNSUp.Failure\\ !ChanUpper.Close\end{array}}$$

$$T2 : \frac{ChanNSDown.Release}{\begin{array}{c}!ChanNSDown.Release\\ !ChanUpper.Close\end{array}}$$

$$SETUP\_PROCEEDING\_ERR\_CHECK \rightarrow DATA\_TRANSMISSION(T3)$$

$$T3 : \frac{NoErrorCode}{\begin{array}{l}!ChanNSDown.Setup(Burst\_Time, Burst\_Delay)\\ if(Burst\_Time = Specified)\\ \quad \{Settimer(Burst\_Timer, Burst\_Time + Burst\_Delay)\}\\ Settimer(KA\_Timer, KA\_Time)\\ OXC\_Config()\end{array}}$$

$$DATA\_TRANSMISSION \rightarrow DATA\_TRANSMISSION(T4)$$

$$T4\ :\ \frac{?ChanNSDown.Keepalive}{\begin{array}{c}!ChanNSDown.keepalive\\ Settimer(KA\_Timer, KA\_Time)\end{array}}$$

$$DATA\_TRANSMISSION \rightarrow IDLE(T5)$$

$$T5\ :\ \frac{?ChanT1.Timeout(Burst\_Timer)}{!ChanUpper.Close}$$

$$T5\ :\ \frac{?ChanNSUp.Failure)}{\begin{array}{c}!ChanUpper.Failure\\ !ChanUpper.Close\end{array}}$$

$$T5\ :\ \frac{?ChanNSDown.Failure}{\begin{array}{c}!ChanNSDown.Failure\\ !ChanUpper.Close\end{array}}$$

Chapter 4

OBS Network Modeling

This chapter describes the OBS simulator and the different network topologies used, the underlying concepts related with OBS networks and the performance metrics used to evaluate the new resource reservation protocol E-JIT$^+$.

In this analysis the performance evaluation is based on the burst loss probability obtained using the mentioned simulator[6]. This study focuses in bufferless OBS networks, i.e. networks that fiber delay lines are not used. Fiber delay lines are used as buffers to keep bursts that cannot be switched immediately when they arrive, because no data channel is available. In bufferless OBS networks, if no data channel is available at the burst arrival time, then the data burst is dropped. Although, this is an important factor in terms of performance, the use of fiber delay lines is too expansive, so other ways to reduce burst loss probability are explored.

4.1 Existent OBS Simulators

To evaluate the performance of the existent OBS resource reservation protocols an OBS network simulator must be used. Simulators are pieces of software that can simulate an entire network of multiple computers and routers interconnected[6]. They are faster and cheaper than making a real network just for test experiences [16].

Actually there are several network simulators but only a few can simulate OBS networks. Some common used OBS simulators are *OBSim* developed in Java by Joel Rodrigues and presented in [6] , $OBS - ns$ simulator developed in C++ by DAWN Networking Research Lab from University of Maryland [16], and *OBSSimulator* developed in C++ by Teng and Rouskas from North Caroline State University [6].

4.1.1 OBSSimulator

In this study, the OBSSimulator developed by Teng and Rouskas is used, because the authors gently gave us access to its source code. *OBSSimulator* is an object-oriented simulator developed using C++ language. It is an event driven, stochastic and symbolic simulator. It is event driven, because data flows to the pace of events of some type [16]. Is stochastic because it uses random objects (usually random variables of numerical value) to simulate randomness of the real-life events and is symbolic because types of symbols to copy the behaviour of real elements are used.

The *OBSSimulator* support the most common OBS resource reservation protocols: JIT, JIT$^+$, Horizon, JET, JumpStart and E-JIT. Considering that E-JIT$^+$ is based in the JIT$^+$ resource reservation protocol, modifying the OBSSimulator to support E-JIT$^+$ becomes easier.

4.2 Network Topologies Under Study

In this study, to evaluate the performance of E-JIT$^+$, some network topologies are used. Rings, chordal rings, NSFNET, ARPANET, Mesh-torus and EON are examples of the used network topologies and are describes in the next sections.

4.2.1 Rings and Chordal ring topologies

Under a ring topology as the name suggests, nodes are arranged in a ring form. Each node is connected directly to two other nodes, one on either side of it. A ring topology has as even number of nodes given by N [6].

As an example, figure 4.1 represents a ring topology with 20 nodes ($N = 20$).

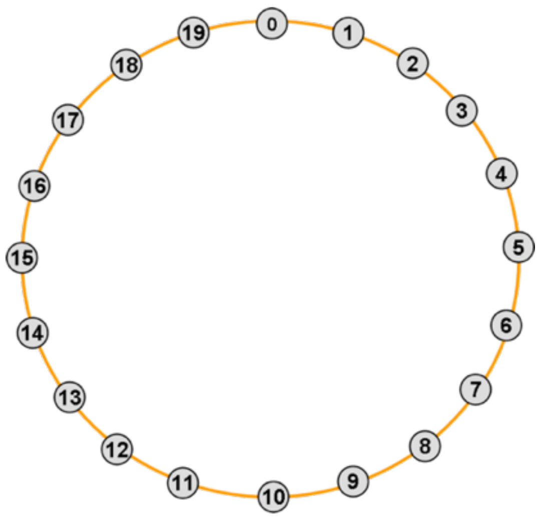
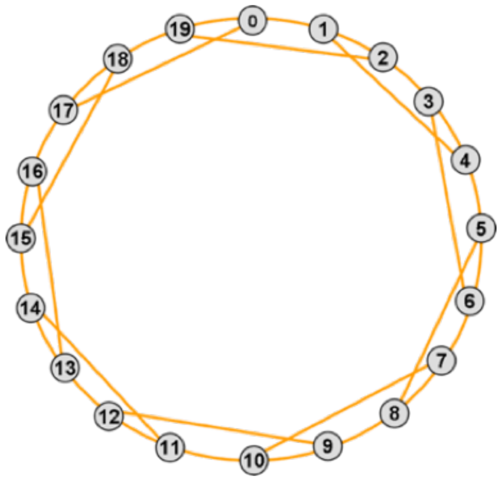


Figure 4.1: Ring network topology with $N = 20$

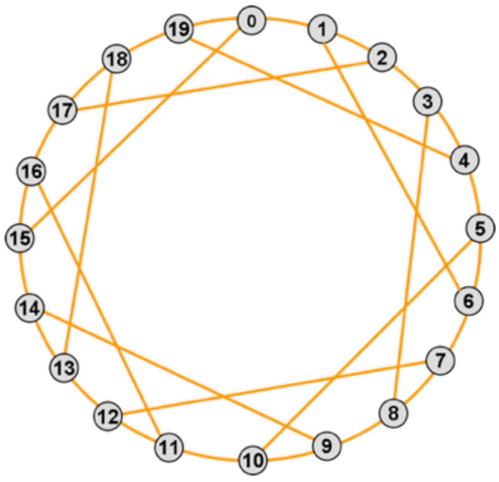
A chordal ring is a bi-directional ring network, in which each node has an additional bi-directional link called a chord.

In a chord ring topology each odd-numbered node i, $(i = 1, 3, \cdots, N - 1)$ is connected to a node $(i + w)\, mod\, N$, where w represents the chord length and is a positive odd number. Each network has an associated network diameter.

The network diameter is "the largest among all of the shortest path lengths between all pairs of nodes, the maximum length of a path being determined by the number of hope" [6]. In an N number of nodes topology there is an optimal chord length that leads to the smallest network diameter. Figure 4.2 represents two chordal ring topologies with 20 nodes ($N = 20$).



a) D3T(1,19,3)



b) D3T(1,19,5)

Figure 4.2: Chordal ring network topologies.

In this study, the notation used in [6] and proposed in [18] to represent ring and chordal ring topologies is adopted. In this notation it is assumed that the link to the previous node and the link to the next node are replaced by chords.

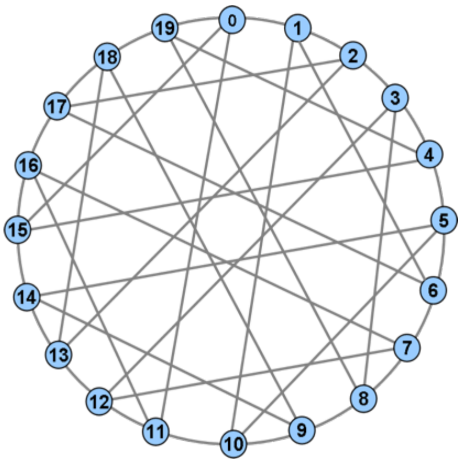
The degree of a network is the number of links at each node. Hence, for example in degree three topology each node i has three chords, one for $i - 1$ node, one for the $i + 1$ node and another for the $(i + w) \, mod \, N$ where w is the chord length. In this notation a chordal ring with chord length $w3$ is simply represented by $D3T(1, N - 1, w3)$ where N is the number of nodes.

This notation can be generalized for a degree n topology as follows[18]:

$$DnT(w1, w2, \cdots, wn)$$

As an example, figure 4.3 represents two degree four topologies.

a)D4T(1,19,5,9)



b) D4T(1,3,5,9)

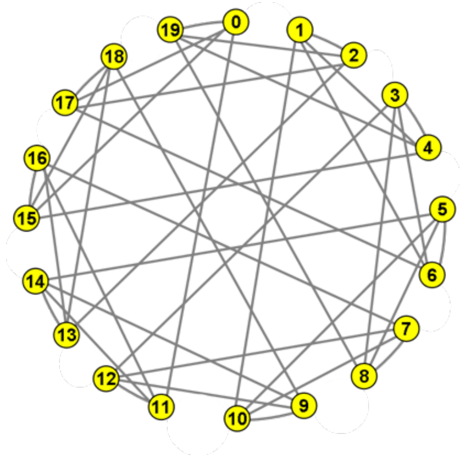
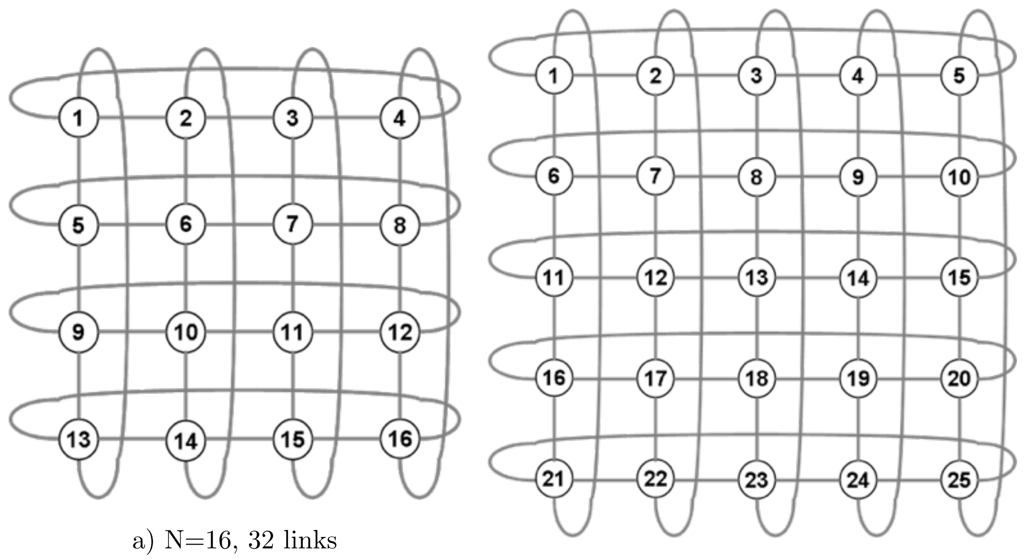


Figure 4.3: Representation of two degree four topologies.

4.2.2 Mesh-torus

Mesh-torus is another regular network topology used in this study. In this network topology each node is interconnected with 4 nodes, so the nodal degree is four. In this performance analysis, mesh-torus topology with 16 nodes and 32 links (4×4) and with 25 nodes and 50 links (5×5)[6] are considered. These networks topologies are illustrated in figure 4.4.



a) N=16, 32 links

b) N=25, 50 links

Figure 4.4: Mesh-torus topologies.

4.2.3 ARPANET

ARPANET is an irregular network with 20 nodes and 32 links having a nodal degree equal to 3.2 [6] . This network was established in 1969 and served as a test-bed linking many universities and research centers. ARPANET was based on a wide área network create by the United States Defense Advanced Research Project Agency (ARPA). Figure 4.5 represents the ARPANET topology.

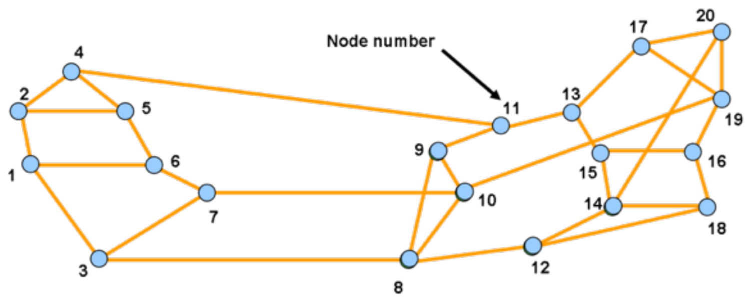
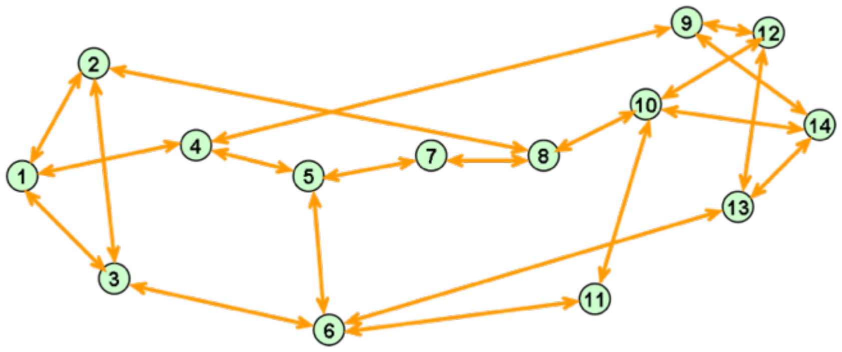


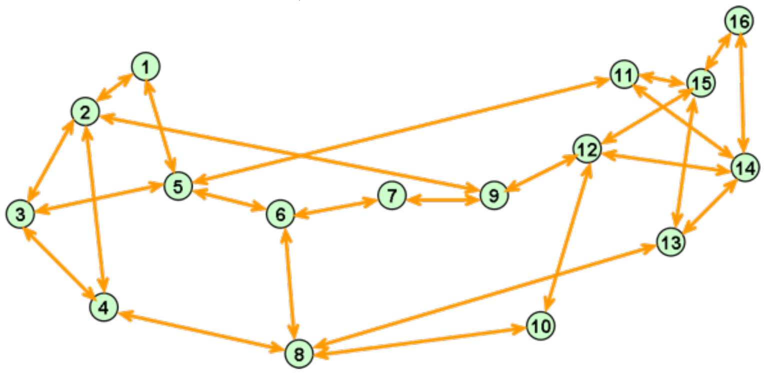
Figure 4.5: Representation of ARPANET topology.

4.2.4 NSFNET

NSFNET is a network topology developed by the USA National Science Foundation (NSF). There are two versions of NSFNET, the most known have 14 nodes and 21 links and a nodal degree equal to 3. Another version have 16 nodes and 25 links and a nodal degree equal to 3.125 [6]. Each of these topologies is represented in figure 4.6.



a) N=14, 21 links



b) N=16, 25 links

Figure 4.6: Representations of NSFNET network topologies.

4.2.5 European Optical Network

The European Optical Network (EON) was proposed in [19] with the objective to interconnect the major European capitals taking into account the traffic distribution according to the population around each node. The EON topology has 19 nodes and 37 links and the nodal degree is equal to 3.89 [6, 19]. Figure 4.7 represents the EON network topology.

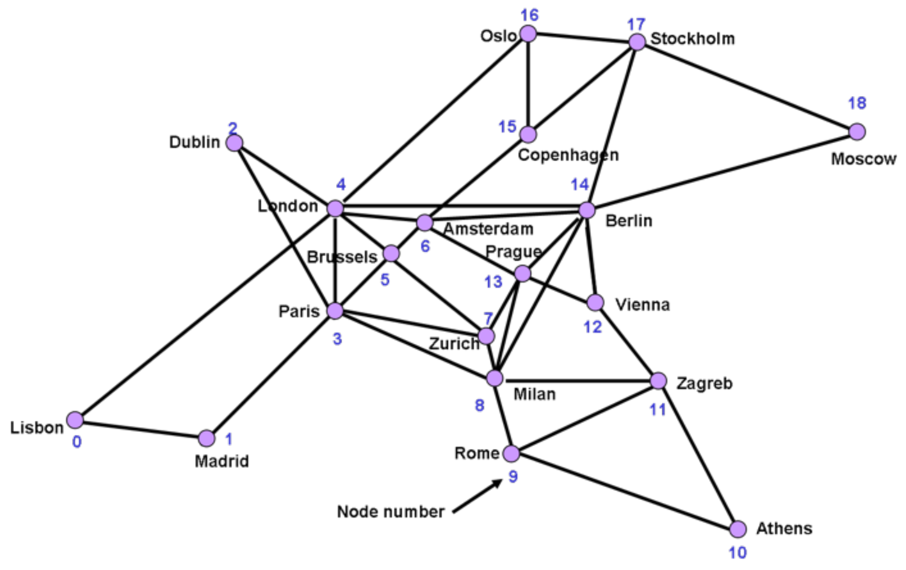


Figure 4.7: Representation of EON network topology.

4.3 Traffic parameters of the OBS Network Model

Some parameters considered in this study where already been mentioned on the previously sections. However, in this section all the relevant parameters to take into account in the performance assessment process are described.

These parameters are[6]:

1. $T_{Setup}(X)$ - Is the time needed to process the setup message using the resource reservation protocol X, where X can be JIT, JIT$^+$, JumpStart, JET, Horizon, E-JIT and E-JIT$^+$;

2. T_{OXC} - Is the time needed to configure the OXC switch fabric in order to establish a connection from an input port to an output port;

3. $T_{Offset}(X)$ - Is the burst offset time for the resource reservation protocol where X can be one of the protocols mentioned in 1. The value of the offset time depends on which protocol is used and must be such that the first bit of the burst arrives at the egress node immediately after the node has been configured;

4. F - Is the number of data channels available per link. Hence, the number of channels connecting two nodes is $F + 1$ where the other channel is the signaling channel. In this study F can be 16, 32, 64 or 128 ($F = 2^n; 4 \leq n \leq 7$).

Another important characteristic is the burst interarrival time. Setup messages arrival process follows a Poisson point process with rate λ and the mean duration of the burst interarrival time is given by $1/\lambda$ [6]. In terms of burst length, it follows an exponential distribution which average is given by $1/\mu$. Hence, as the average burst length distribution is $1/\mu$ and the setup message arrival rate is λ, then the burst generation ratio is denoted as λ/μ.

4.4 Performance Metrics

In this study, to measure the performance of existent resource reservation protocols and the new protocol E-JIT$^+$, the following metrics are used:

- Burst loss probability;
- Nodal degree gain

The first mentioned metric, burst loss probability, is the probability of a given burst does not arrive at his destination[6]. This is an important metric because bursts can contain lots of data and if the burst does not arrive at the destination this data is lost, and the communication quality is compromised. In an OBS network a burst is lost if resources cannot be reserved in time. This situation can occur due to traffic congestion or if another failure happens.

The nodal degree gain its a new metric presented in [6] and is used to quantify the benefits due to the increase of the nodal degree. This metric is represented as follows:

$$G_{n,k}(i,j) = \tfrac{P_i(n)}{P_j(k)}$$

In this expression $P_i(n)$ represents the burst loss probability at the $i-th$ hop of a degree n topology and $P_j(k)$ is the burst lost probability at the $j-th$ hop of a degree k topology. This metric can be used only if the given topologies have the same network conditions.

Chapter 5

Performance Assessment of E-JIT$^+$

In this section, an analysis of the performance assessment of E-JIT$^+$ compared with other one-way resource reservation protocols is presented. This analysis is made using the OBS simulator described in section 4.1.1, for OBS networks with ring and chordal ring topologies. As mentioned in section 3.3, E-JIT$^+$ is a new resource reservation protocol for OBS networks. It is based on the existent resource reservation protocol JIT$^+$. Joel Rodrigues in [6] as made a comparative study of the existent one-way resource reservation protocols: JIT, JumpStart, JIT$^+$, JET and Horizon. This comparative study was made taking into account mesh topologies with 16 and 20 nodes, analysing:

- The number of nodes impact,
- The influence of nodal degree on the nodal degree gain,
- The influence of the message setup processing time
- The influence of the OXC configuration time

As an example, figure 5.1 illustrates a graph plotting values of burst lost probability as a function of the number of nodes using the ring $D2T(1, N-1)$ and chordal ring $D3T(1, N-1, 5)$ and $D3T(1, N-1, 7)$ topologies for the existent OBS resource reservation protocols, JIT, JIT$^+$, JET, Jumpstart, Horizon, E-JIT and E-JIT$^+$.

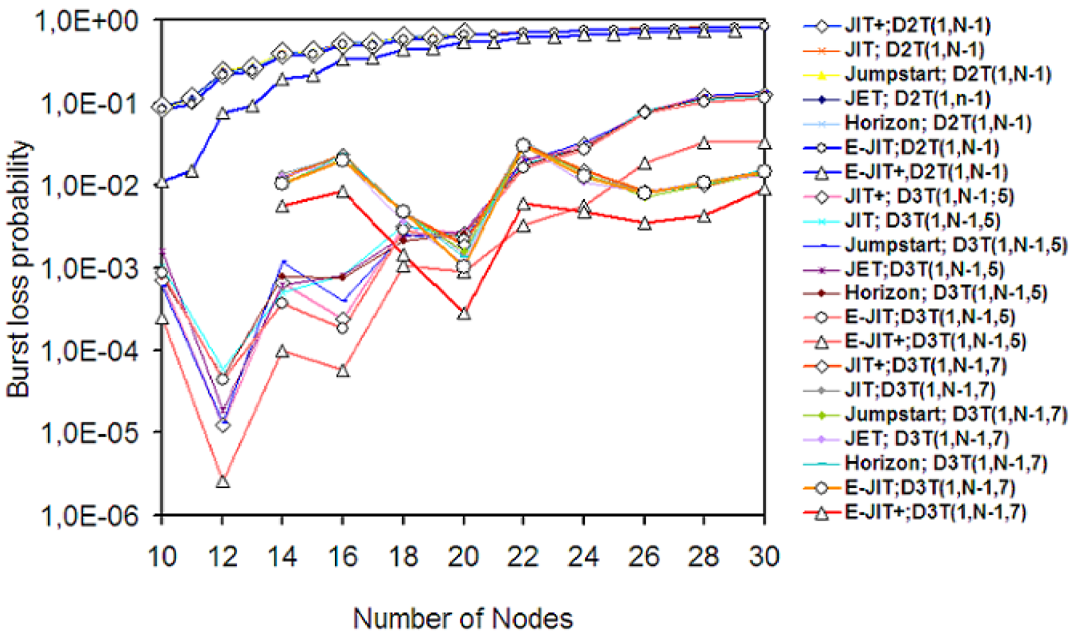


Figure 5.1: Burst loss probability, as a function of the number of nodes in the last hop. T_{Setup}(JIT)=T_{Setup}(JIT$^+$)=T_{Setup}(Jumpstart)=T_{Setup}(JET)=T_{Setup}(Horizon)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, λ/μ=32, F=64.

As may be seen the existent protocols JIT, JIT$^+$, JET, Jumpstart and Horizon have a similar performance in all the network topologies considered. The most recent protocol E-JIT performs better than the protocols referred above. It is possible to see also, that the new protocol proposed in this study, E-JIT$^+$, have a better performance in all the considered network topologies.

An important observation is that, due the fact of plotting the performance of the seven mentioned OBS resource reservation protocols the figure becomes difficult to understand because probability lines overlaps on each other. Thus, taking these facts into account, the same strategy to evaluate the E-JIT$^+$ performance used in [6] to evaluate E-JIT is followed. Therefore, in this section the performance between E-JIT, JIT$^+$ and E-JIT$^+$ protocols is compared, because E-JIT$^+$ is based on JIT$^+$ and E-JIT has the best performance.

To evaluate the E-JIT$^+$ performance, this study uses rings, degree three and degree four chordal rings, FCCN-NET,NSFNET with 14 and 16 nodes, Mesh-torus with 16 and 25 nodes, ARPANET and EON topologies.

5.1 Impact of the Number of Data Channels

This section focuses on the performance assessment of E-JIT$^+$ compared with JIT$^+$ and E-JIT for networks having 16 and 20 nodes. Networks with the smallest diameter are used, because they have best performance as demonstrated in [6]. With 16 nodes (N=16), D3T(1,15,5) and D4T(1,15,5,13) have the best performance for degree three and degree four network topologies, respectively. With 20 nodes (N=20), D3T(1,19,7) and D4T(1,19,3,9) have the best performance for degree three and degree four topologies, respectively. Following the first parameter to analyze is the performance regarding the number of data channels per link, and then the burst loss probability on each hop, maintaining the number of channels.

Using the OBs simulator presented in section 4.1.1, several simulations where performed in order to analyze the effects of the variation of the number of data channels per link. The simulator returns burst loss probabilities in each hop as output.

Figure 5.2 plots the burst loss probability in the last hop for E-JIT$^+$, JIT$^+$ and E-JIT in the ring D2T(1,15), in chordal rings D3T(1,15,5) and D4T(1,15,5,13), in NSFNET and Mesh-torus networks with 16 nodes ($N = 16$) and $\lambda/\mu = 32$.

As may be seen when a higher number of data channels is available, generally the burst lost probability is smaller. However the highest degree network D4T(1,15,5,13) have the best performance because its burst loss probability is the smallest in every number of plotted data channels. Moreover, the D2T(1,15) network have the worst performance. Therefore according to the graph plotted in figure 5.2 , network topologies can be sorted according their performance from the best to the worst as: D4T(1,15,5,3), D3T(1,15,5), Mesh-torus, NSFNET and D2T(1,15). Figure 5.2 also demonstrates that E-JIT$^+$ performs better than JIT$^+$ and E-JIT, especially when the number of data channels is higher.

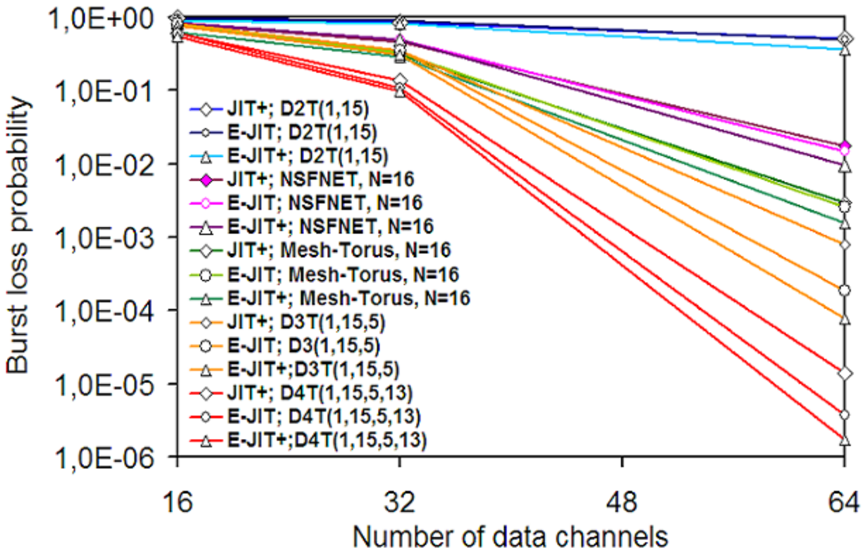


Figure 5.2: Burst loss probability, as a function of the number of data channels. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, λ/μ=32 , N=16.

Figure 5.3 confirms the previously observations made for figure 5.2, as may be seen this figure represents the burst loss probability as a function of number of hops. The network topologies and the value of λ/μ are the same considered previously, the number of data channels (F) is 64.

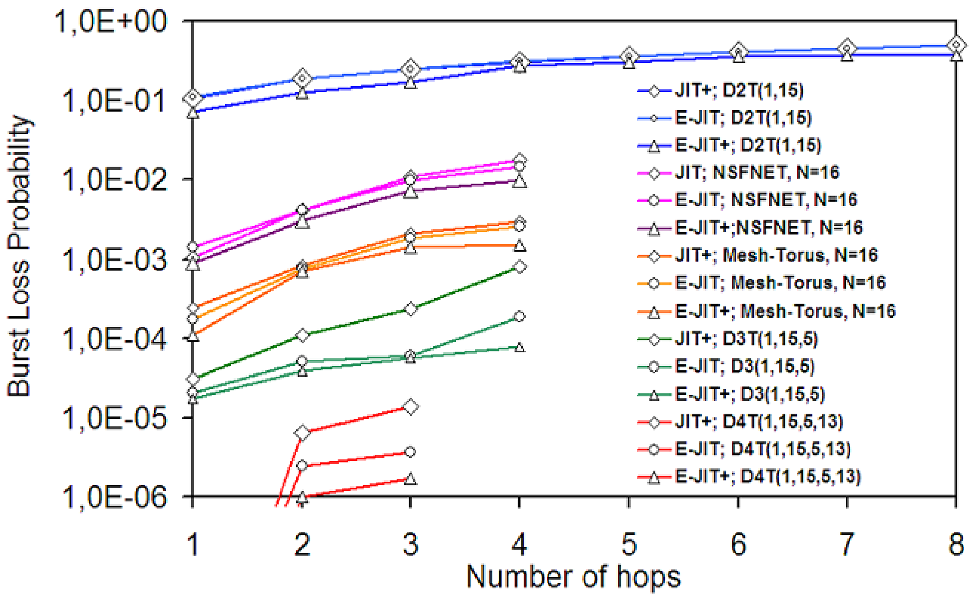


Figure 5.3: Burst loss probability, as a function of the number of hops. T_{Setup}(E-JIT$^+$)=T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=12.5 μs, λ/μ=32 ,
N=16, F=64.

As may be seen, the best performance belongs to the degree four topology D4T(1,15,5,13) which have the smallest diameter. In this topology the burst lost probability in the first hop tens to zero.

According to their performances, topologies can be sorted from the best to the worst as: D4T(1,15,5,13), D3T(1,5,5),Mesh-torus, NSFNET and D2T(1,15). The network with the highest diameter is D2T(1,15) and consequently the burst lost probability is very high. Moreover, as may be seen the E-JIT$^+$ protocols performs better than JIT$^+$ and E-JIT for all the specified topologies, however the improvement of performance is greater on the degree four topology D4T(1,15,5,13).

Once analyzed the performance assessment of E-JIT$^+$ relative to JIT$^+$ and E-JIT for networks with 16 nodes, following the performance of E-JIT$^+$ for networks around 20 nodes is present. Figure 5.4 represents the burst loss probability in function of the number of data channels for D2T(1,19), D3T(1,19,7), D4T(1,19,3,9), ARPANET and EON topologies with λ/μ =32.

As may be seen in figure 5.4, like networks with 16 nodes, for networks with 20 nodes, the burst lost probability decreases when more data channels are available. Networks with higher degree have also the best performance and the lowest degree have the worst performance.

Thus, these network topologies can be sorted according to their performance,

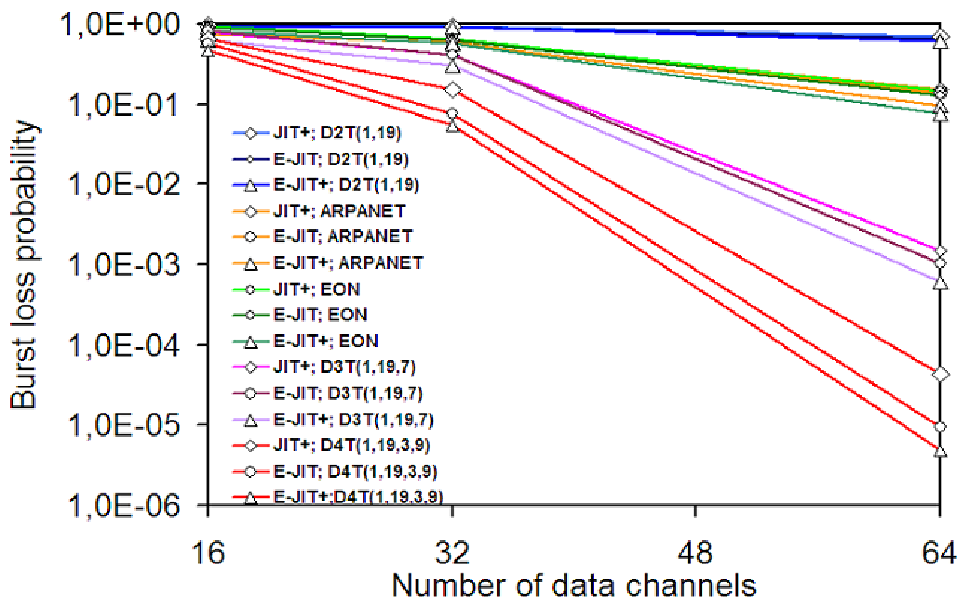


Figure 5.4: Burst loss probability, as a function of the number of data channels. T_{Setup}(E-JIT)=T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT$^+$)=12.5 μs, λ/μ=32.

from the best to the worst as: D4T(1,19,3,9), D3T(1,19,7), EON, ARPANET and D2T(1,19). As may bee seen, the performance of the E-JIT$^+$ protocol performs better than JIT$^+$ and E-JIT especially for lowest values of burst loss probability.

Another observation is the fact that the EON and ARPANET topologies have performances very similar, but as may bee seen E-JIT$^+$ performs slightly better for EON than for ARPANET.

Figure 5.5 represents the burst loss probability as a function of the number of hops for D2T(1,19), D3T(1,19,7), D4T(1,19,3,9), ARPANET and EON topologies with λ/μ =32. As may bee seen the results observed in the previous figure are confirmed.

Figure 5.5 shows also that network topologies with higher degree perform better than topologies with lowest degree. Moreover topologies with lower diameter, performs better than topologies with higher diameter. The relative performances between the EON and ARPANET topologies can be observed. Although the EON diameter is smaller than the ARPANET diameter their performances are similar, which reveals the importance of how links interconnects nodes.

In the next paragraphs, the effects of the variation of λ/μ is analyzed. As mentioned in section 4.3, the value of λ/μ represents the burst generation ratio, which determines the quantity of traffic in the network.

Figure 5.6 illustrates a graph of the burst lost probability on the last hop as a function of the λ/μ parameter for D2T(1,15), D3T(1,15,5), D4T(1,15,5,13), Mesh-torus (N=16) and NSFNET (N=16) network topologies using E-JIT$^+$, JIT$^+$ and E-JIT. The number of channels per link is 64 (F=64).

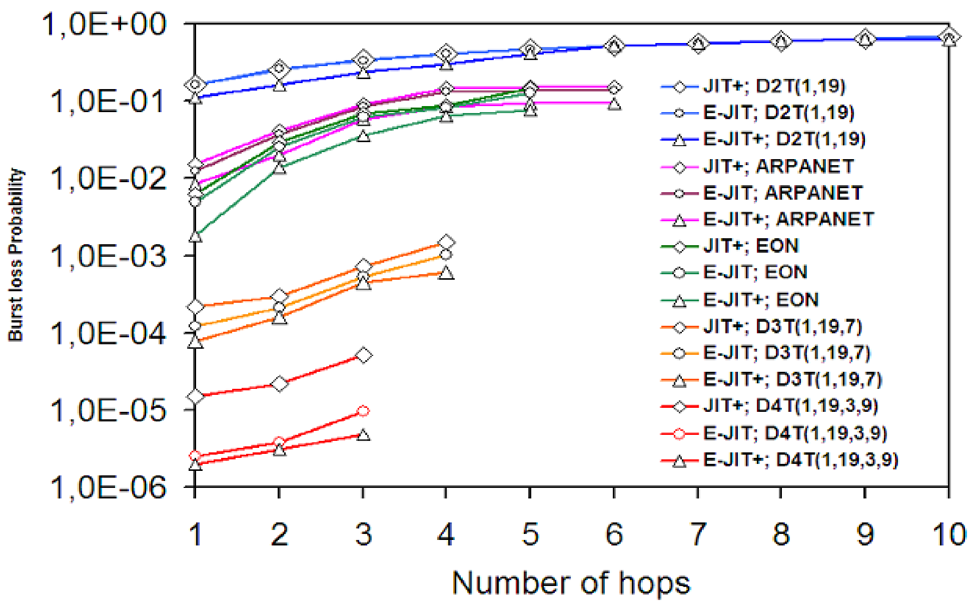


Figure 5.5: Burst loss probability, as a function of the number of hops. T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=T_{Setup}(JIT$^+$)=12.5 μs, λ/μ=32.

As may be seen in figure 5.6 when the value of λ/μ increases the value of the burst lost probability also increases. This fact is due to the quantity of traffic in the network.

Figure 5.6 shows also that the performance of E-JIT$^+$ is slightly better than the performance of E-JIT and JIT$^+$ in all the network topologies considered. As may be seen when the λ/μ value is less than 32 for chordal rings, less than 25,6 for NSFNET and less than 19,2 for the ring D2T(1,15), the burst loss probability is zero.

Figure 5.7 illustrates a graph of the burst lost probability on the last hop as a function of the λ/μ parameter for D2T(1,19), D3T(1,19,7), D4T(1,19,3,9), EON and ARPANET network topologies using E-JIT$^+$, JIT$^+$ and E-JIT. The number of channels per link is 64 (F=64). As may be seen for the networks considered with N=20 the E-JIT$^+$ protocol also performs better than JIT$^+$ and E-JIT. When the value of λ/μ is less than 32 for chordal rings, less than 19.2 for EON and ARPANET and less than 12,8 for the ring D2T(1,19), the burst loss probability is zero.

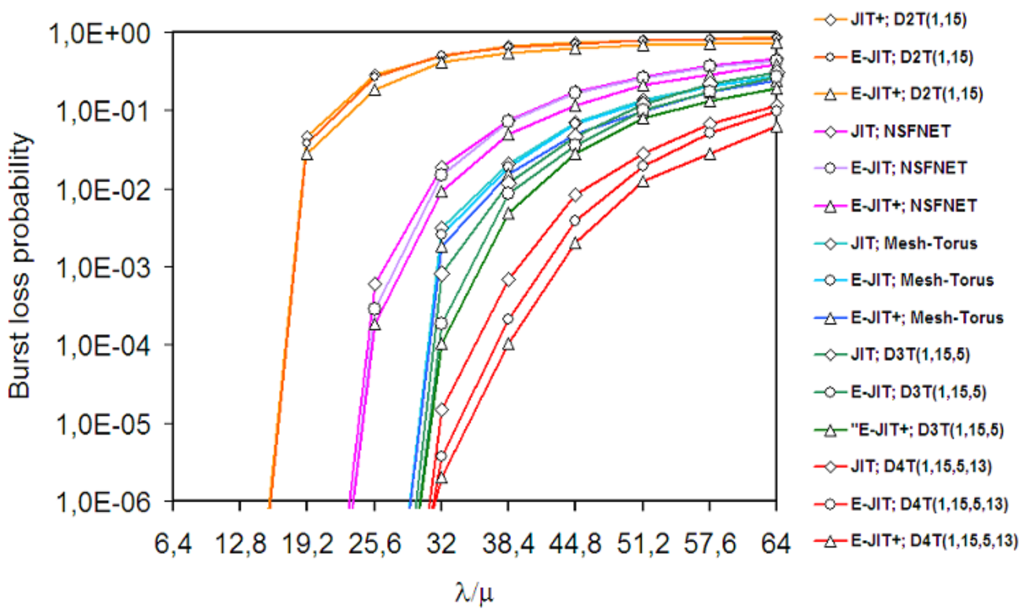


Figure 5.6: Burst loss probability, as a function of λ/μ. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64, N=16.

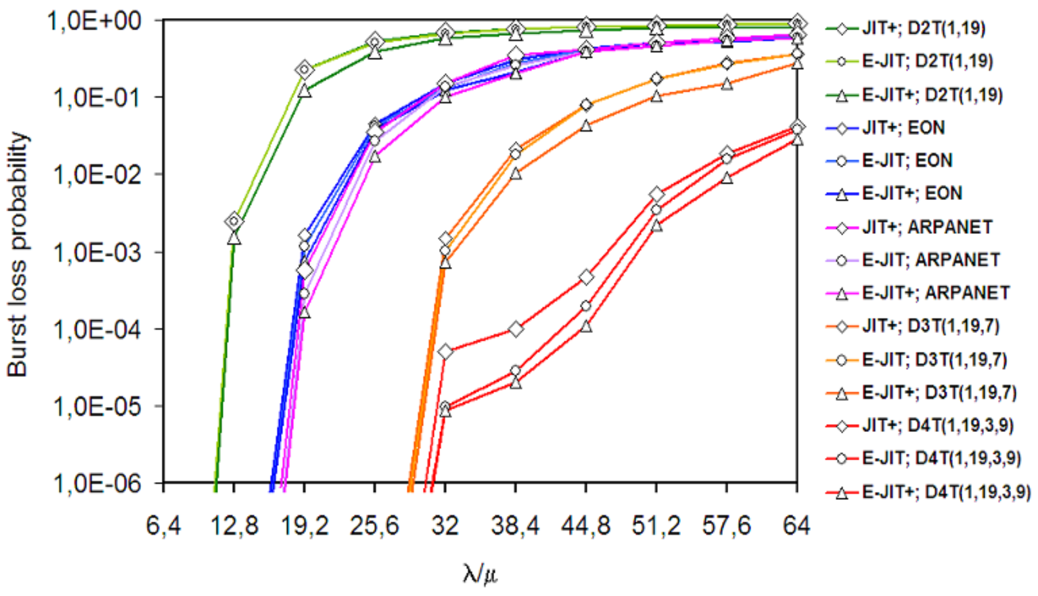


Figure 5.7: Burst loss probability, as a function of λ/μ. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64.

5.2 Impact of Number of Nodes

In this section the impact of the number of nodes on the burst loss probability for the E-JIT$^+$, JIT$^+$ and E-JIT protocols is analyzed. Hence different mesh topologies having between 10 and 30 nodes are used. In figure 5.8 networks with nodal degree around two and three are used, and figure 5.9 uses networks with nodal degree around three and four. For comparison purposes a network topology of degree three (D3T(1,N-1,5)) is used on both figures referred. Hence, degree two D2T(1,N-1) is used. Examples of networks with nodal degree around 3 are D3T(1,N-1,7) having nodal degree equal to 3, NSFNET with N=14 having degree equal to 3, NSFNET with N=16 having degree equal to 3.125 and ARPANET having degree equal to 3.2.

Examples of networks with nodal degree equal to four are Mesh-torus with N=16 and Mesh-torus with N=25. Finally having nodal degree equal to 3.89 EON is used as nodal degree around four.

As may be seen ring networks with nodal degree equal to 2 has the worst performance for all the number of nodes used. The best performance belongs to the chordal ring D3T(1,N-1,5) when the number of nodes used is less and equal than 18 and equal to 22 and to the chordal ring D3T(1,N-1,7) for nodes equal to 20 and more than 22. Figure 5.8 shows also that NSFNET with N=14 performs better than NSFNET with N=16 and ARPANET.

FCCN-NET with N=14 has a bad performance when compared to the degree two ring. Moreover, as may be seen the performance of E-JIT$^+$ when compared to JIT$^+$ and E-JIT is better and significant for all the network topologies used, expect for degree two chordal rings when the number of nodes used is more than 16.

Figure 5.9 shows that when using 18 or more nodes nodal degree four D4T(1,N-1,5,9) performs better than nodal degree three D3T(1,N-1,5). It shows also that Mesh-torus with N=25 performs better than Mesh-torus with N=16. As may be seen, E-JIT$^+$ also performs better then JIT$^+$ and E-JIT for networks with nodal degree around four and for all the number of nodes used.

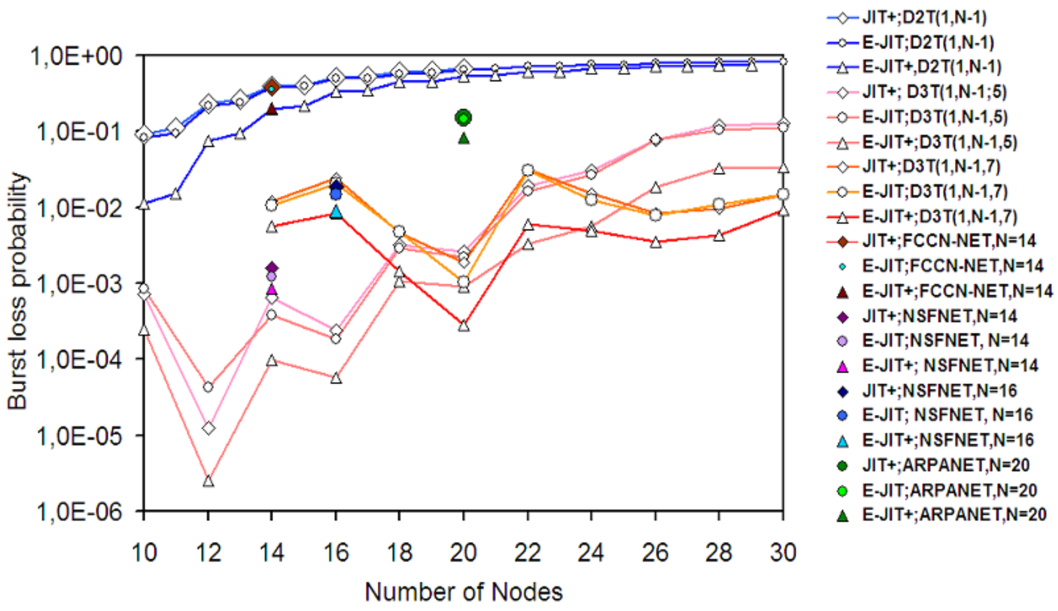


Figure 5.8: Burst loss probability, as a function of number of nodes for degree 2 and degree 3 network topologies T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 32$.

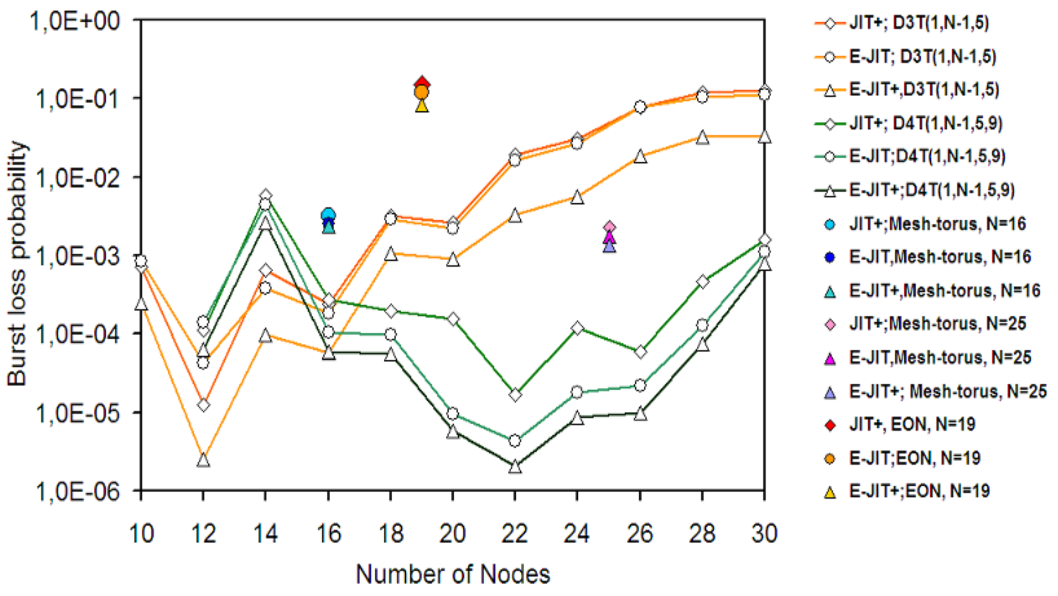


Figure 5.9: Burst loss probability, as a function of number of nodes for degree 3 and degree 4 network topologies T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 32$.

5.3 Impact of the Nodal Degree on the Nodal Degree Gain

In this section the impact of the nodal degree on the nodal degree gain is analyzed. Hence, the performance of the E-JIT$^+$ protocol is compared with the performance of JIT$^+$ and E-JIT protocols, based on the nodal degree gain.

As mentioned in section 4.4 the nodal degree gain is a new performance metrics to evaluate the benefits of increasing the nodal degree. Therefore network topologies with nodal degree between three and five are considered.

Figures 5.10 and 5.11 show the nodal degree gain for the E-JIT$^+$, JIT$^+$ and E-JIT protocols. In the first figure network topologies having $N \leq 16$ are used and figure 5.11 groups network topologies having $N \geq 19$. Figure 5.10 shows the nodal degree gain in the last hop of each topology achieved by E-JIT$^+$, JIT$^+$ and E-JIT as a function of the nodal degree, due to the increase of the nodal degree from 2 using D2T(1,14) to 3 using NSFNET with N=16 and from 2 using D2T(1,15) to 3 using D3T(1,15,5), to 3.125 using NSFNET with N=16, to 4 using D4T(1,15,5,13) and Mesh-torus with N=16, and to 5 using D5T(1,15,7,3,9). The number of data channels is 64 (F=64) and the burst generation ratio is $\lambda/\mu = 32$.

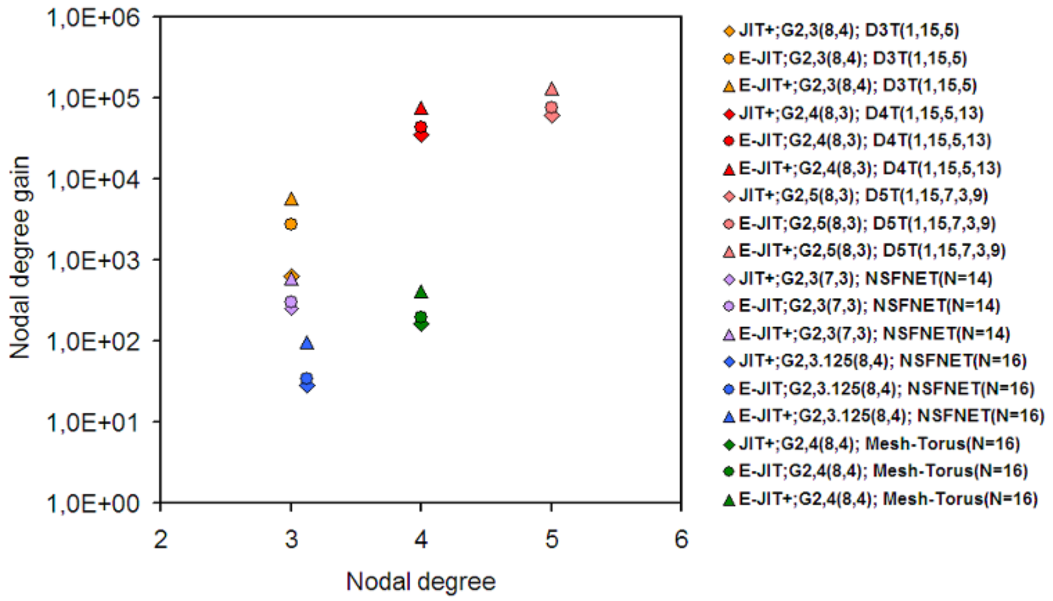


Figure 5.10: Nodal degree gain in the last hop of network topologies with $N \leq 16$ as a function of de nodal degree. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 32$.

Figure 5.11 shows the nodal degree gain in the last hop of each topology achieved by E-JIT$^+$, JIT$^+$ and E-JIT as a function of the nodal degree, due to the increase of the nodal degree from 2 using D2T(1,18) to 3,89 using EON with N=19, from 2 using D2T(1,19) to 3 using D3T(1,19,7), to 3.2 using ARPANET with N=20, to 4 using D4T(1,19,3,9) and to 5 using D5T(1,19,3,7,11) and from

D2T(1,24) to 4 using Mesh-torus with N=25. The number of data channels is 64 (F=64) and the burst generation ratio is $\lambda/\mu = 32$.

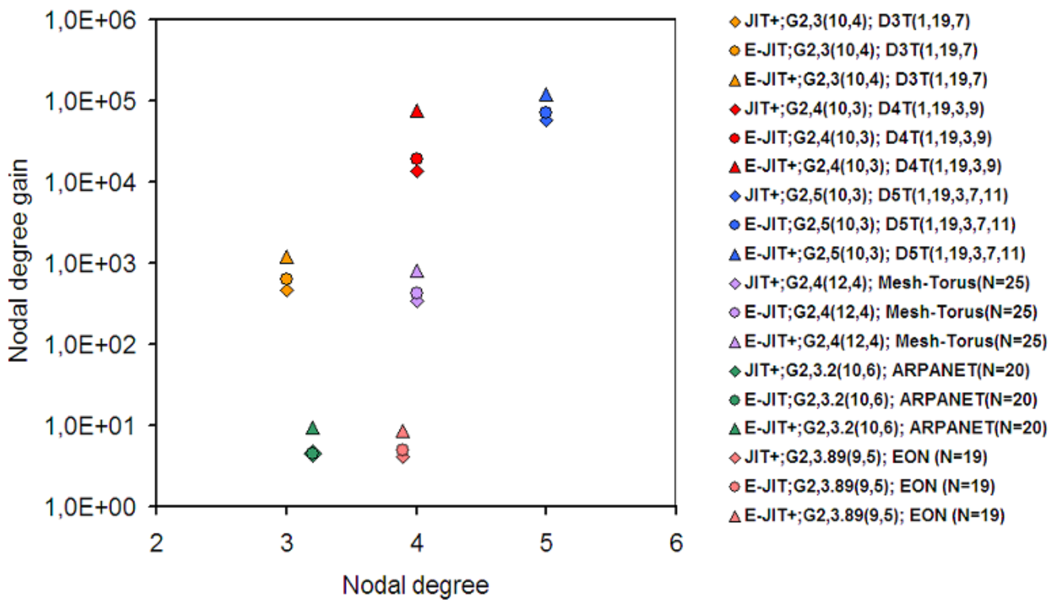


Figure 5.11: Nodal degree gain in the last hop of network topologies with $N \geq 16$ as a function of de nodal degree. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 32$.

As may be seen in both figures the highest gain when the nodal degree increases from 2 to around 4 is achieved for degree four chordal ring D4T(1,19,3,9) having N=20. On the other hand the smallest gain when the nodal degree increases from 2 to around 4 is obtained for the EON with N=19 topology.

When the node degree increases from 2 to 5, the gain is around five orders of magnitude for the chordal ring D5T(1,15,7,3,9) (N=16) and for the chordal ring D5T(1,19,3,7,11).

The nodal degree gain obtained with the E-JIT$^+$ protocol is higher than that obtained by the E-JIT and JIT$^+$ protocols. The highest nodal degree gain of E-JIT$^+$ compared with JIT$^+$ and E-JIT is achieved when the nodal degree increases from 2 to 4 for the ring with degree four D4T(1,19,3,9).

From these results it is possible to conclude that the way of how nodes are connected is very important because the burst lost probability is very sensitive on different topologies. Moreover, more connections between nodes does not mean better performance of these topologies.

5.4 Impact of Setup Message Processing Time and OXC Configuration Time

In this section an analysis of the Setup Message Processing time and the OXC configuration time is made comparing the assessment of E-JIT$^+$, E-JIT and JIT$^+$ protocols. Therefore in the figures presented in this section, network topologies are grouped considering the nodal degree: a group with nodal degree around three and other group with nodal degree around four.

Figure 5.12 plots the burst loss probability as a function of the OXC configuration time in the last hop of NSFNET with N=14 and N=16, ARPANET and D3T(1,19,7) for the E-JIT$^+$, E-JIT and JIT$^+$ protocols.

In this evaluation the number of data channels is F=64 and the burst generation rate is $\lambda/\mu = 32$. The value of T_{Setup} time is fixed on 12.5 μs because is the value achieved on the current available technology (JITPAC controlllers [20]). The value of T_{OXC} ranges from the value estimated for a near future ($T_{OXC} = 20\,\mu s$) up to ten times the value achieved by the current available technology ($T_{OXC} = 10 \times 10ms = 100\,ms$).

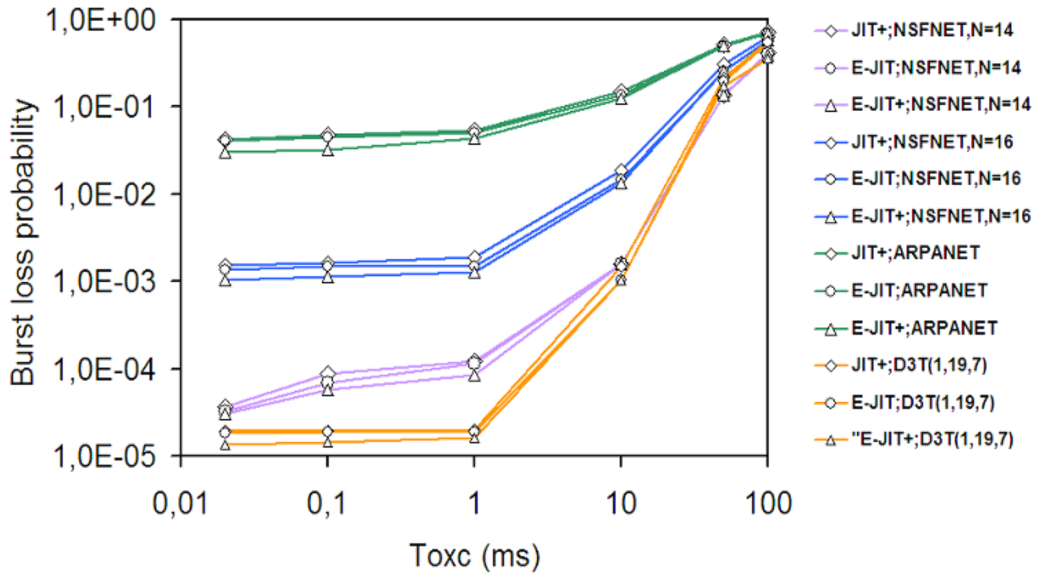


Figure 5.12: Burst loss probability as a function of OXC configuration time. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 32$.

As shown, the degree free ring network D3T(1,19,7) has better performance than other networks when $T_{OXC} \leq 10ms$. On the other side ARPANET has the worst performance. Moreover, is shown that when the T_{OXC} decreases less than 1 ms the performance of this networks remains almost constant, which means that efforts on reducing the T_{OXC} value less than 1 ms are not necessary. Figure 5.12 shows also that the performance of the E-JIT$^+$ protocol when compared with E-JIT and JIT$^+$ is better in all the network topologies considered.

After the analysis with fixed T_{Setup}, in figure 5.13 the value of the T_{Setup} is a function of the variation of the T_{OXC} according to a linear interpolation given by the following expression [6]:

$$T_{Setup}(X) = 1 + \frac{11.5}{10^4 - 20} \times (T_{OXC}(X) - 20)\ (\mu s) \tag{5.1}$$

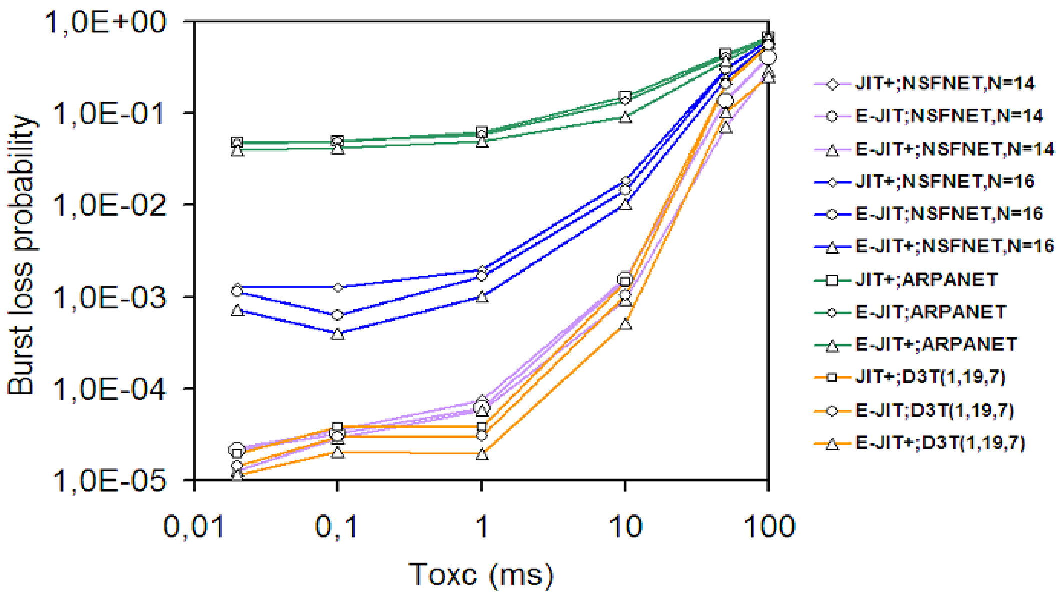


Figure 5.13: Burst loss probability as a function of OXC configuration time. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64, $\lambda/\mu = 32$ and T_{Setup} changes according to (5.1).

As may be seen in figure 5.12 when $T_{OXC} = 0.1$ ms the performance of the chordal ring with degree three, D3T(1,19,7), slightly increases when compared with the figure 5.12.

In figure 5.14 the burst lost probability is plotted as a function of the setup message processing time in the last hop of the same networks topologies considered in the figures above, with F=64 and $\lambda/\mu = 32$. The value of the message processing time ranges from 1 ms and 12,5 ms. The value of the TOXC time is computed according to a linear interpolation given by the following expression [6]:

$$T_{OXC}(X) = 20 + \frac{(T_{Setup}(X) - 1) \times (10^4 - 20)}{11.5}\ (\mu s) \tag{5.2}$$

As may be seen when the setup message processing time increases, generally the value of the burst lost probability also increases. Another observation is related with the better performance of the E-JIT$^+$ protocol when compared to E-JIT and JIT$^+$ protocols.

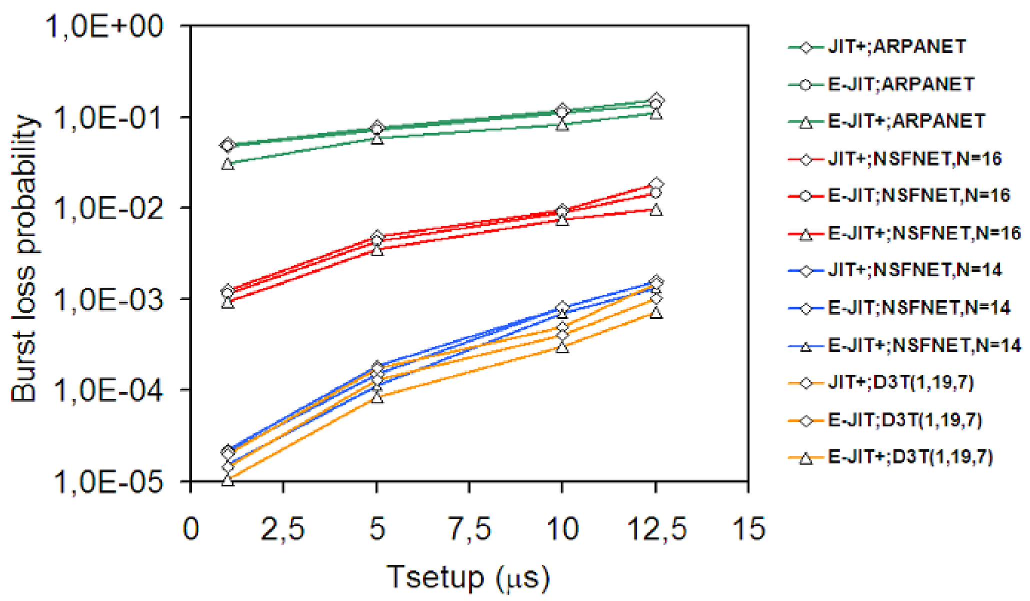


Figure 5.14: Burst lost probability as a function of T_{Setup}. F=64, $\lambda/\mu = 32$ and T_{OXC} changes according to (5.2).

Figures 5.15, 5.16 and 5.17 plots values of the burst lost probability on the same conditions of figures 5.12, 5.13 and 5.14 respectively, for network topologies with nodal degree around four and changing the value of λ/μ to 44.8. Network topologies used in these figures are Mesh-torus with N=16 and N=25, EON and D4T(1,19,3,9).

The degree four chordal ring has the best performance as may be seen in figures 5.15 and 5.16 . On the other hand EON has the worst performance. The same nodal degree and the same connections between nodes are possible explanations to this fact. It may also be observed that E-JIT$^+$ performs better in these conditions than E-JIT and JIT$^+$, mainly for degree four chordal ring when $T_{OXC} \leq 10ms$.

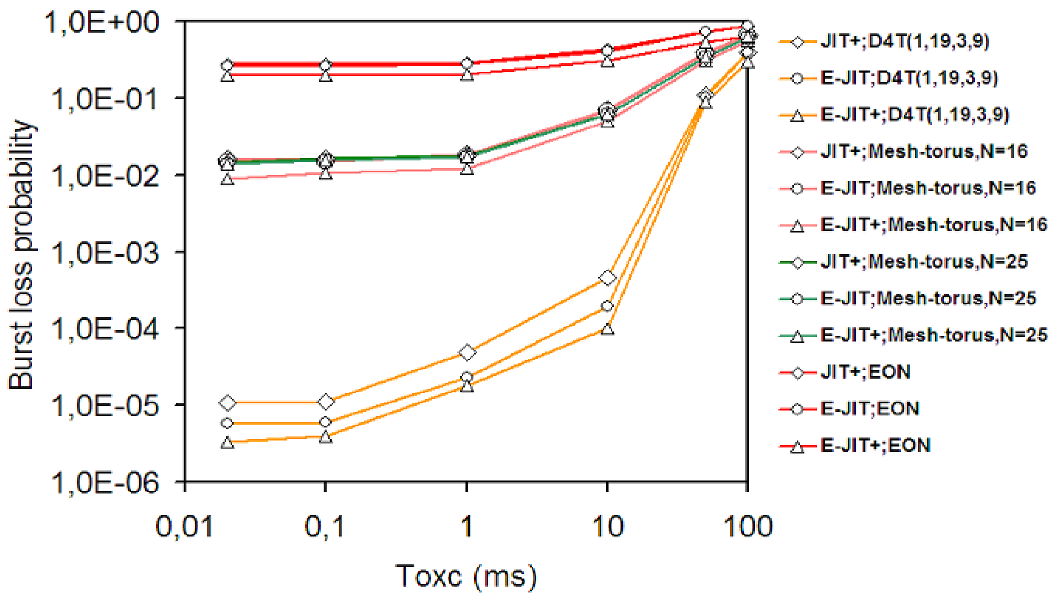


Figure 5.15: Burst lost probability as a function of T_{OXC}. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64 and $\lambda/\mu = 44.8$.

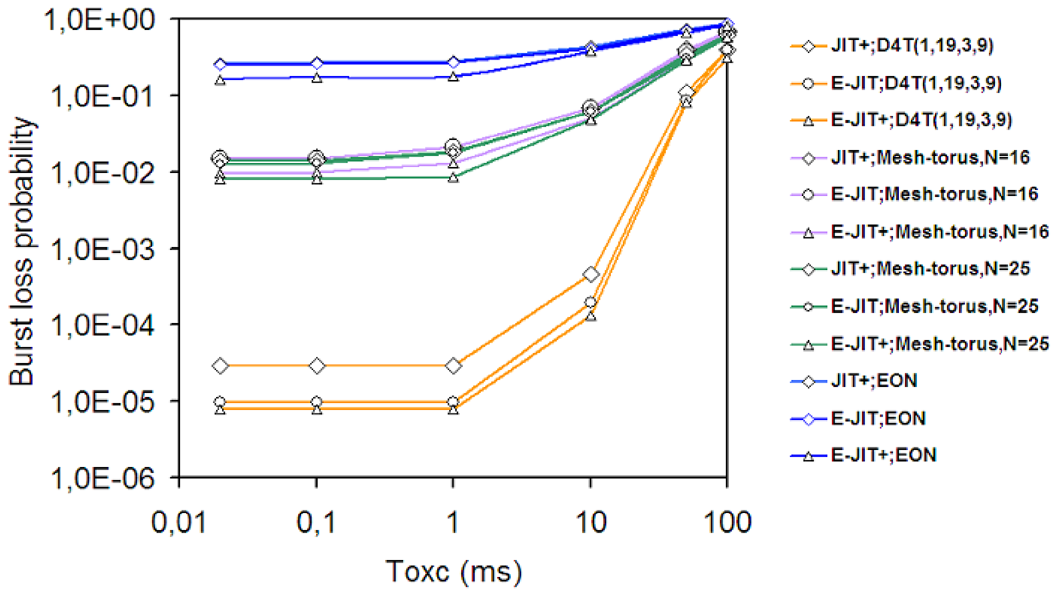


Figure 5.16: Burst lost probability as a function of T_{OXC}. T_{Setup}(JIT$^+$)=T_{Setup}(E-JIT)=T_{Setup}(E-JIT$^+$)=12.5 μs, F=64, $\lambda/\mu = 44.8$ and T_{Setup} changes according to (5.1).

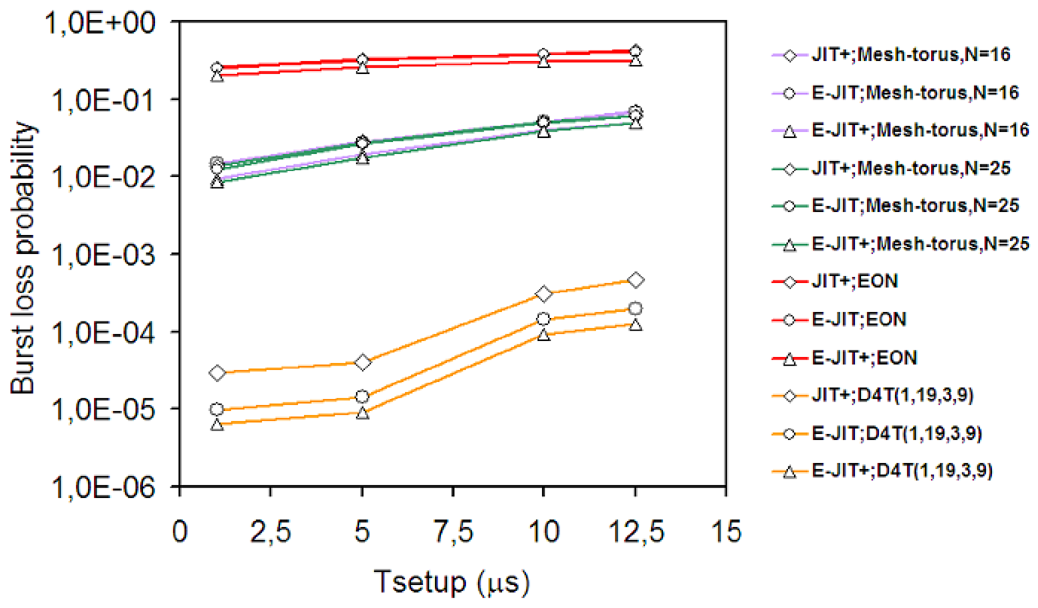


Figure 5.17: Burst lost probability as a function of T_{Setup}. F=64, $\lambda/\mu = 44.8$ and T_{OXC} changes according to (5.2).

Chapter 6

Conclusions

In this dissertation a study in the scope of OBS networks was made. The first chapter presented the objectives and main contributions to the state-of-the art.

The second chapter focus on the state-of-the art of OBS Burst Networks and the underlying concepts. A special attention was given to the description of the OBS network architecture and the operation of edge nodes and core nodes.

After, in the third chapter a detailed description of the operation mode of the existent one-way resource reservation protocols was presented. Hence, we described the JIT, JIT$^+$, JET, Horizon and E-JIT resource reservation protocols operation mode. Furthermore, a new resource reservation protocol for OBS networks, called E-JIT$^+$ was presented and its correspondent formal description also.

In chapter 4, the existent OBS simulators where described, and the motivations to use the OBSSimulator on the performance assessment of E-JIT$^+$ compared with E-JIT and JIT$^+$ where enumerated.

Chapter 4 presents also the description of the network topologies used in the performance assessment of E-JIT$^+$. It describes ring and chordal ring topologies as well as the NSFNET, ARPANET, EON and Mesh-torus network topologies. After that, performance metrics used to evaluate the E-JIT$^+$ protocol compared with E-JIT and JIT$^+$ where presented. Metrics based on the burst loss probability like the impact of the number of nodes on the burst loss probability, the increasing network traffic, the nodal degree gain, and the effects of the switch fabric configuration time and the setup message processing time where defined.

In chapter 5 the relative performance of the existent one-way reservation protocols was analyzed. The performance of the existent resource reservation protocols is very similar, however E-JIT performs slightly better than others. Hence, considering that E-JIT$^+$ is a JIT$^+$ based protocol and E-JIT has the best performance, in the performance assessment of E-JIT$^+$ it was compared with both the referred resource reservation protocols. The resting resource reservation protocols where not considered to preserve the readability of the figures presented.

The values of the burst loss probability obtained with the new protocol E-JIT$^+$ decreases in all cases considered when compared with JIT$^+$ and E-JIT protocols. Hence, it is possible to conclude that the performance achieved by E-JIT^{+} is better in all network topologies studied. Moreover the nodal degree gain was greater when E-JIT^{+} was used, which reveals a better performance. Regarding the nodal degree gain, the best performance results where obtained when the nodal degree increases from 2 to 4 using the degree four chordal ring with 20 nodes denoted by D4T(1,19,3,9).

Considering the variation of the network traffic, it was observed that when the network traffic increases the burst lost probability also increases.

The analysis of the effects of the OXC configuration time on the burst loss probability reveals that, generally when this time increases the burst loss probability also increases, but when the OXC configuration time is less than 1 ms (T$_{OXC} \leq 1ms$) the burst loss probability remains constant which means that efforts on reduction of T_{OXC} are not justified.

After, the analysis of the effects of the setup message processing time on the burst loss probability was presented. As observed in chapter 5 when T_{Setup} increases the burst loss probability also increases, which reveals that a reduction of T_{Setup} can improve the performance of an OBS network.

References

[1] R. J. Bates and D. W. Gregory, “Voice & Data Communications Handbook”: McGraw-Hill Professional, 2006.

[2] E. Haselton,“A PCM frame switching concept leading to burst switching network architecture”, Communications Magazine, IEEE, Vol. 21, pp. 13-19, 1983.

[3] S. R. Amstutz, “Burst switching-an update”, Communications Magazine, IEEE, Vol. 27, pp. 50-57, 1989.

[4] C. G. K. Dolzer, J. Späth, and S. Bodamer, ”Evaluation of reservation mechanisms for optical burst switching” Vol. 55, January 2001.

[5] E. Monteiro and F. Boavida, “Engenharia de Redes Informáticas”, Lisboa: FCA - Editora de Informática, 2000.

[6] J. Rodrigues, “Optical Burst Switching Networks Architectures and Protocols”, Covilhã: Serviços Gráficos da Universidade da Beira Interior, 2008.

[7] T. Battestilli and H. Perros,“An introduction to Optical Burst Switching”, IEEE Communications Magazine, Vol. 41, pp. S10-S15, August 2003.

[8] C. Kan, H. Balt, S. Michel, and D. Verchere,“Information model of an optical burst edge switch”, Proceedings of IEEE International Conference on Communications (ICC 2002), pp. 2717-2721, April- 28 May 2, 2002 New York.

[9] V. Vokkarane and J. Jue,“Prioritized Burst Segmentation and Composite Burst Assembly Techniques for QoS Support in Optical Burst-Switched Networks”, EEE Journal on Selected Areas in Communications, Vol. 21, pp. 1198-1209, September 2003.

[10] J. Turner, “Terabit Burst Switching”, Department of Computer Science, Washington University, St. Louis, Technical Report WUCS-97-49, December 1997.

[11] C. Qiao and M. Yoo, ”Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet”, Journal of High Speed Networks, Vol. 8, pp. 69-84, January 1999.

[12] J. Y. Wei and R. I. McFarland, ‘Just-in-time signaling for WDM optical burst switching networks”, Journal of Lightwave Technology, vol. 18, pp. 2019-2037, 2000.

[13] I. Baldine, G. N. Rouskas, H. G. Perros, and D. Stevenson, “JumpStart: a just-in-time signaling architecture for WDM burst-switched networks”, Communications Magazine, IEEE, Vol. 40, pp. 82-89, 2002.

[14] J. Teng and G. N. Rouskas, “A Detailed Analisys and Performance Comparison of Wavelength Reservation Schemes for Optical Burst Switched Networks”, Photonic Network Communications, Vol. 9, pp. 311-335, May 2005.

[15] A. H. Zaim, I. Baldine, M. Cassada, G. N. Rouskas, H. G. Perros, and D.Stevenson, “The JumpStart Just-In-Time Signaling Protocol: A Formal Description Using EFSM,” Optical Engineering, Vol. 42, pp. 568-585, February 2003.

[16] V. N. G. J. Soares, I. D. C. Veiga, and J. J. P. C. Rodrigues, “OBS Simulation Tools: A Comparative Study”, in Proceedings of IEEE International Conference on Communications (IEEE ICC 2008)Workshops (13th Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD 2008)) Beijing, China, May 19-23, 2008.

[17] D. N. R. Lab, “DAWN Research Lab”, DAWN Research Lab, January 15th, 2004.

[18] M. M. Freire, J. J. P. C. Rodrigues, and R. M. F. Coelho, “The Role of Network Topologies in the Optical Core of IP-over-WDM Networks with Static Wavelength Routing”, Telecommunications Systems, Vol. 24, pp. 111-122, 2003.

[19] M. J. O Mahony, “A European optical network: design considerations”, Proceedings of IEE Colloquium on Transparent Optical Networks: Applications, Architectures and Technology, London, UK, IEE, pp. 1/1-1/6, April 22, 1994.

[20] I. Baldine, M. Cassada, A. Bragg, G. Karmous-Edwards, and D. Stevenson, “Just-in-Time Optical Burst Switching Implementation in the ATDnet All-Optical Networking Testbed”, in Proceedings of IEEE GLOBECOM 2003, IEEE (Ed.). Vol. 5 San Francisco, USA, December 1-5, 2003, pp. 2777-2781.