An Optical Burst Switching Control Plane Architecture and its Implementation

J. Triay, J. Rubio and C. Cervelló-Pastor
Dep. Enginyeria Telemàtica, Universitat Politècnica de Catalunya
E-mail: joan.triay@upc.edu, jesus.rubio@entel.upc.edu , cristina@entel.upc.edu

Abstract—**This paper proposes a new design and implementation of a control plane for Optical Burst Switched networks. The design is based on the principles of generality, transparency, portability and efficiency. In this way, the control plane is designed to be easily reused in any type of network node, low-level Data Plane or high-level Wavelength Reservation Scheme. Making efforts to address these issues, we implement a general-purpose, flexible and feasible OBS network testbed using field programmable gate arrays (FPGA).**

I. Introduction

THE rapid growth of Internet traffic requires high transmission rates between network nodes, specially in core networks. These rates fall beyond conventional electronic router's capabilities. The exploitation of huge bandwidth in optical fiber cost effectively is essential for the development of the next-generation optical Internet.

Current WDM networks operate over point-to-point links, being required an optical-to-electrical-to-optical (O/E/O) data conversion at every step. The future designs are focused on optical networks where the user data travel entirely in the optical domain without any O/E and E/O conversion. Three main optical network solutions have been proposed as Gauger *et al.* presents in [1]: Wavelength-Routed Networks (WRNs), Optical Burst Switched Networks (OBSNs), and Optical Packet Switched Networks (OPSNs). Due to the lack of a practical pure-optical packet switching solution for an optical Internet, OBS has been proposed as the best feasible alternative.

OBSNs have some advantages in comparison to the other two technologies [2] [3]. On one side, it efficiently uses statistical multiplexing in the optical layer to overcome the wavelength switching inefficiency of WRNs. On the other side, it is more feasible to be implemented than OPSs since requirements of the last one demand for very high switching rates, all optical processing, and faster optical memory technologies that, in fact, are not commercially available yet. Optical approaches are under development to address the problem of switching capacity in the data network but commercial solutions are not expected within a few next years.

Another advantage of OBS, and maybe the most important one, is that the switching time requirements are much more relaxed, since the length of a burst generally is much longer than the length of a packet. Currently, optical switches of higher performance provide a switching time between 25 ns to 1 ms and a switching rate of about 10000 operations per second depending on their switching technology. OBS's switching requirements can easily fall between those values.

In optical burst switching (OBS), data is transported in various-size units, which are called bursts. This is the basic data block that is transferred, and it can be defined as a collection of data packets that are assembled into a bigger unit. These packets can share any of these features: they can have the same network egress address, or have common attributes, like QoS requirements.

Nodes in an OBS network can either be edge nodes or core nodes. Edge nodes are responsible for assembling input packet within burst. It is a task of these nodes to determine the way of assembling these bursts.

Core nodes are responsible for receiving, processing and forwarding the control packets and reserving the optical resources, specifically, those designated to carry out the switching on the optical cross-connect matrix (OCX) at a concrete time between input and output data ports.

OBS differentiates two planes, as it is shown in Fig. 1: the data plane and the control plane. The all-optical (OOO) data plane is responsible for the transportation of the bursts as optical signals; meanwhile the opto-electronic (OEO) control plane is responsible for the signaling, routing, network management and other network control functions. The data plane is entirely optics; this means that bursts are just switched in the optical domain. On the other way, control plane requires the processing of the control packets at nodes, and so, an O/E/O conversion is required. A control packet is transmitted ahead of the burst in order to configure the switches along the burst's path. An OBS node does not wait for confirmation that an end-to-end connection has been set up, instead it starts transmitting a data burst after an offset time, following the transmission of the control packet.

This strong separation between data and control planes offers a better network manageability and flexibility. For this, an innovative way of implementing this technology could be on providing the best performance at each plane.

The main objective of the data plane is to execute the assembling/disassembling of bursts based on the packet's

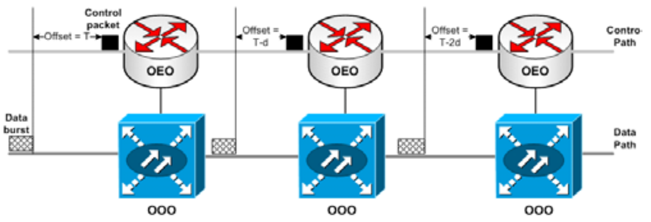

Fig. 1. Separated data and control planes on an OBS network.

attributes as fast as possible.

On the other hand, the objective of the control plane is to carry out the reservation of the optical resources demanded by the control packet.

Due to the one-pass reservation strategy and statistical multiplexing, burst loss can occur. In the case of contention, efficient resolution strategies in OBS core nodes are essential. These schemes achieve a low burst blocking probability as required in transport networks.

This paper focuses on the definition and design of the control plane's features and architecture providing a way for implementing them on real hardware. Currently, there is no standard definition of how this implementation should be done. What is more, there is not any standard protocol of the control plane's operation, for example, the control packet's structure and the control message types remain unstandardized. However, several wavelength reservation schemes have been proposed and research on this field continues.

The paper is organized as follows. In Section II control plane design and architecture model are presented, along with the description of the high level operation of the control plane. We also propose a format of the control packets. In Section III, we present some highlights of the technologies that are currently used at the implementation phase. Finally, Sections IV and V are focused on future research and efforts, and the conclusions related to the present work.

II. Control Plane Design and Architecture

This section describes the operation protocol of the control plane, and the proposed design and architecture to implement it. As the design will be implemented into a real testbed, some considerations about it should be met. These can be summarized on:

- *Generality*. The control plane must provide a common framework where different reserve protocols could be implemented on the top.
- *Transparency*. Control plane must be independent of the data plane implementation. The same control plane should be easily adapted whatever the data plane is.
- *Portability*. The control plane implementation must be designed for being easily portable to any kind of network node. For example, the same control plane implementation could be used on a node that is acting only as an edge or core router, or a node that has both capabilities.
- *Efficiency*. The last principle focuses on providing the more efficient way of processing control packets, and as a result, to allow the reservation and switching of data bursts as fast as possible.

An example of network topology and possible proof-of-concept implementation is shown on Fig. 2. Three OBS nodes compose this network: two nodes act as edge, and the last operates both as edge and core. The three nodes share the same OXC and each one can be origin or destination of user data.

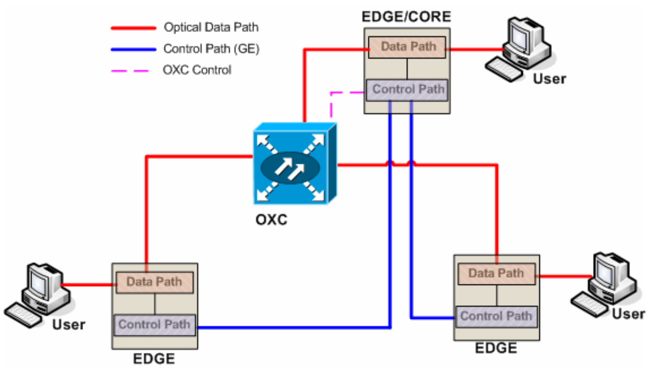

Fig. 2. Data and control plane interactions

A. *High Level Protocol Operation*

As previously pointed, generality is a constraint in the design. A common framework must be provided in which different and concrete reserve schemes can be used.

To support this principle a common high level scheme must be implemented. This scheme operates as follows: Considering a flow of bursts between two different endpoints, A and B, for each of these flow a SETUP message will be sent on the control path to the first node, demanding channel resources. The transmission of the burst will be delayed a period of time known as *offset*. During this time, the CPU of the current node must process the message and calculate the ingress and egress ports on which the burst will be received and transmitted, and the exact time and duration of the optical cross-connected matrix switching. Moreover, the control process will regenerate the SETUP message according to the burst attributes. This message is then delivered to the next node through the correct control channel.

It is worth standing out the importance of correctly assigning the switching times of allocating and releasing of resources. Two types of resource release can be noted: implicit, if the node does not need any explicit message or notification (see Fig. 3); or explicit, if a RELEASE message is received from the previous OBS node according to the protocol definition (see Fig. 4). The burst will be deleted if the OBS node has not enough resources to switch it.

B. *Wavelength Reservation Schemes*

This subsection describes different wavelength reservation schemes for OBS. Based on the setup and release mechanisms, Baldine *et al.* [4] describe four types of wavelength reservation schemes.

i) Explicit setup and explicit release. In this scheme, the control packet contains the offset of the burst, but not the duration of it. The reservation of resources starts immediately after the switch receives the setup message, and ends when the release message is received. *Just-In-Time* (JIT) proposed by Wei and McFarland [5] is an example of this scheme.

ii) Explicit setup and an estimated release. In this case, the setup message contains both the offset and the duration of the burst. Each wavelength has an associated deadline indicating the resource will become free. The reservation starts after the switch receives the setup message and ends when the burst is switched and transmitted, a time that is

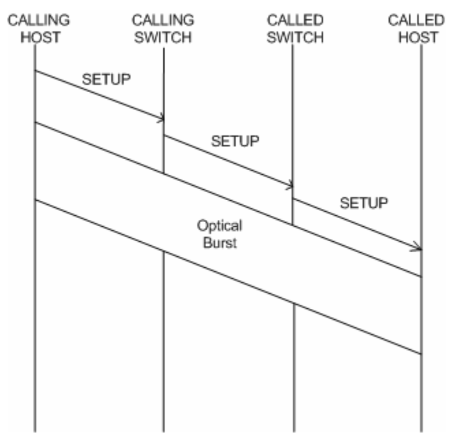

Fig. 3. Example of explicit setup signaling.

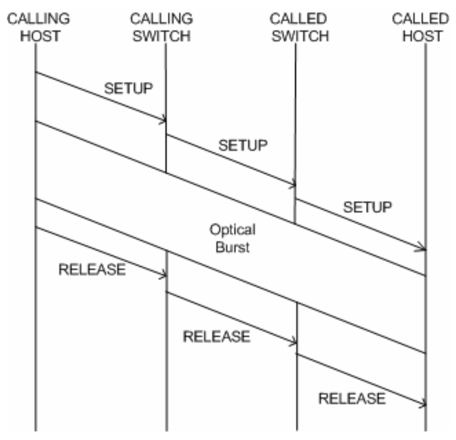

Fig. 4. Example of explicit setup and explicit release signaling.

calculated using the duration field. An example of this scheme type is the *Horizon* scheme proposed by Turner [6].

iii) Estimated setup and an explicit release. In this scheme, the setup control message contains only the offset of the burst. The reservation starts at the beginning of the burst. This time is calculated using the previous offset information. The release of resources is executed once the release message arrives.

iv) Estimated setup and an estimated release. The setup message of this scheme has the offset and the duration of the burst. Reservation and releasing of resources are calculated using previous data. Qiao and Yoo proposed an example of this scheme, the *Just-Enough-Time* (JET) scheme [2].

The JIT protocol is significantly simpler than either JET or Horizon, since it does not involve complex scheduling or void filling algorithms. Therefore, JIT is amenable to hardware implementations. The difference between JET and JIT resides on the idea of how JET intends to estimate when the data burst will be received, and therefore it won't reserve resources until that instant. This can increase the efficiency and performance of the network in terms of channel utilization, but it can arouse burst looses if the instant is not well calculated. On the other hand, JIT reserves the resources on the instant at which the setup control packet is receive. This method decreases the performance of the network, but it improves the burst losses rate and its implementation is more feasible.

Over the last years, research in OBS networks has progressed to prototypes and proof-of-concepts demonstrations. For example, the JITPAC hardware [7], which was developed by MCNC-RDI, implements the JIT signaling protocol and in [8] the authors present the design and implementation of the JET protocol.

In this work, we propose to implement OBS nodes operating under diverse wavelength reservation schemes in order to obtain low latency and high bandwidth utilization, combined with contention resolution protocols.

C. OBS Control Packet Format

To support a common framework for the wavelength reservation schemes and their high-level operation, it is necessary to define the format of the control packets. This common format will allow defining easier methods to generate and to process each and every kind of possible control message.

Fig. 5 shows the proposal for control message format specifying its fields and the length. Following, the meaning of each field is described:

- NDA (2 bytes): It is the Node Destination Address. For instance, the egress OBS node's address. It is a 2-byte length field, resulting 65.536 possible addresses. This number is sufficient for deploying an operator's optical transport network. Moreover, the proposed addressing is not hierarchical (there are not different classes of addresses), but reserve 0x0000 and 0xFFFF for operational purposes.
- NSA (2 bytes): The Node Source Address specifies the address of the ingress OBS node. It has the same length of the previous NDA field.
- IDBURST (2 bytes): It is the ID of the burst. Its length, 2 bytes, provides 65.536 possible IDs. This ID is generated by the ingress OBS node. The set of the previous three fields (NDA + NSA + IDBURST) identifies uniquely a burst in the network. Once the OBS node reaches the end of IDs, it should restart its ID counter to 0, and so on.
- TYPE (1 byte): This field specifies the type of control packet. A 1-byte length field gives us up to 256 types of messages. As previously introduced in the figure, some messages can be defined, for instance: SETUP, ACK, NACK and RELEASE messages.
- QOS (1 byte): The QoS field provides information about any QoS requirement that should be met when

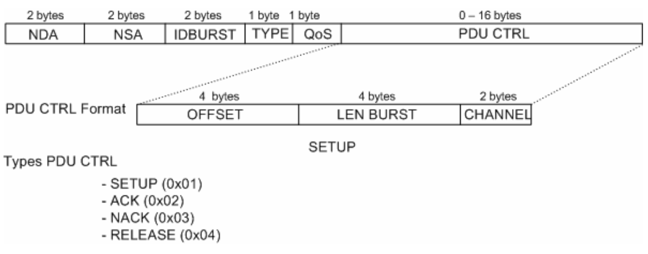

Fig. 5. Fields and length of the control packet.

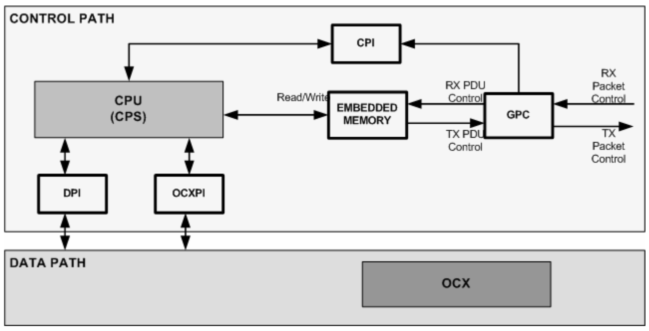

Fig. 6. High level architecture of the control plane.

the burst is being processed. For example, this field could be filled with any QoS profile for adapting the content resolution of bursts at the core routers or when disassembling the burst into packets, if for instance, the packets need a special treatment or de-queuing method.

- PDUCTRL (0-16 bytes): This field varies its contents depending on the type of control packet specified by the TYPE field. For example, the format of a SETUP PDU could be as follows: 4 bytes to specify the OFFSET of the burst, 4 more bytes to inform of the burst length in bytes, and finally 2 bytes to specify on which wavelength (CHANNEL) the burst will be received.

Some of the previous fields can be modified at each node hop. For example, depending on the wavelength reservation scheme that is being used, the offset can be recalculated or not. The input and output wavelengths at core routers could change, and so, the value of CHANNEL field should have a value according to the output wavelength to be used.

The proposed addressing format follows a routing perspective. This means, that at each network hop, the NDA is checked, and the next hop on the path is determined from the routing tables. Moreover, this gives to the nodes the possibility of acknowledging to the ingress OBS node (it just has to check the NSA of the control packet) if any of the bursts are lost because there were not enough resources for switching it.

D. *OBS Control Path Design*

According to the principles of design specified in the section II, the control plane and its associated control path should be transparent to lower data planes, and so, it must remain as much independent as possible to the concrete data path implementation.

To succeed in performing a fast and efficient treatment of the bursts, most of the elements involved in the design must be implemented directly into hardware, and so, software should only be executed on very specific tasks. Taking this into mind, it is necessary specifying which operations are suitable to be processed by the Control Path Software (CPS). Depending on the kind of router, edge or core, the node must execute different operations. On one hand, an edge node has the following capabilities:

- Assembling and disassembling of bursts depending on the egress destination address or other attributes.
- Transmission and reception of bursts at a certain time over a specified wavelength.
- Creation of control packets depending on the type of traffic to be transmitted.
- Reception and processing of control packets transmitted by other nodes.

On the other hand, a core switch should be capable to perform:

- Reception and processing of control packets.
- Modification of the control packet, if required, and transmission to the next OBS node on the path to the egress node.
- Reservation and release of resources in the OXC according to the information contained in the received control packets.

From above operation, the Control Path Software must be only responsible of creating and processing the contents of the control packet depending on the upper reservation schemes that could be used, or other capabilities, as specific QoS processing. It won’t carry out any operation needed to support storing or releasing of packets into/from memory. Fig. 6 shows our control plane architecture, in which the Control Path clearly differentiates to the Data Path and the OCX, but it is not totally isolated. The CPU which runs the Control Path Software, can interact with both through their respective interfaces, the Data Path Interface (DPI) and the OCX Path Interface (OCXPI).

The DPI is responsible for querying the CPS for signaling a new Data Path when one or some bursts are ready to be transmitted as a set of data packet are assembled into a burst/bursts. This task identifies an edge node capability, and so, the core nodes do not interact to the DPI. Analogously, the OCXPI is responsible of translating the CPU queries of switching to the OCX. In contrast to the previous interface, the OCXPI must always interact with the CPU whatever type of node is being implemented.

As a part of the control plane, we also need to receive, process and transmit the control packets. To alleviate the CPS, some pre-process of the control packets is entirely run from hardware. This comprises the Gigabit Packet Controller (GPC) and the embedded memory. The GPC is responsible for assembling control messages into low-level frames in order to be transmitted over any kind of link layer protocol, for instance, a Gigabit Ethernet interface; or de-assembling these link layer frames, extracting the control message and storing it in the embedded memory, from which the CPU and its CPS can process it.

E. *Control Path Software and Use Case Examples*

Control Path Software (CPS) is responsible for calculating the offset and duration of the bursts. Once this information is ready and recorded, it must be able of constructing the control PDU if a new control packet is going to be transmitted to the next OBS node. Fig. 7 and Fig. 8 show some use cases that the CPS has to carry out.

Fig. 7 exposes the case in which a new burst is ready to be transmitted. The CPS receives the query of signaling a path for transmitting the last burst. Before its transmission, it is necessary to perform a setup and wavelength reservation along the light path. In this way, CPS and its

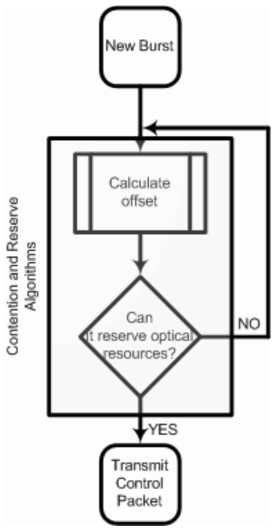
New Burst
Contention and Reserve Algorithms
Calculate offset
Can it reserve optical resources?
NO
YES
Transmit Control Packet

Fig. 7. Control Packet creation algorithm.

higher contention and reserve algorithms calculate the required amount of delay time (offset) depending on the available resources. If transmission is possible, the Control packet is finally assembled and transmitted to the next OBS node. Depending on higher level protocols or QoS profiling, if there are not enough resources, the burst can remain in memory and the CPS can reinitialize the operation of reservation later.

Fig. 8 presents an example of control packet reception/forwarding. In this case, the offset must be checked in order not to reserve resources if the burst is going to be received wrongly. If there is enough offset in which carrying out the control packet processing and reservation of resources, the reception/forwarding of a control packet is possible. If not, this control packet should be discarded, and depending on the reservation protocol, some kind of control message should be transmitted back to the previous OBS node notifying about this error.

In the next step, CPS filters if the control packet's destination address is the current node. In this case, the node only needs to reserve optical resources for reception, and so, forwarding of control packet will not occur. When forwarding is required, the program must check the availability of optical resources to switch the burst. If there are not available resources an explicit notification method should take the responsibility of sending back an error message. In the case of availability of resources, the offset will be recalculated subtracting the processing time. Finally, the control packet is transmitted to the next OBS node on the egress path.

III. Implementation Technologies

A. *FPGA: A suitable hardware solution*

FPGA is a technology that allows programming and reusing of hardware. These devices are a compromise between general purpose processors used in software implementations and Application Specific Integrated Circuits (ASIC). FPGA can be reprogrammed as control plane protocols evolve. Moreover, new FPGA provide high speed serial interfaces, from 1 Gb/s to 10 Gb/s. These

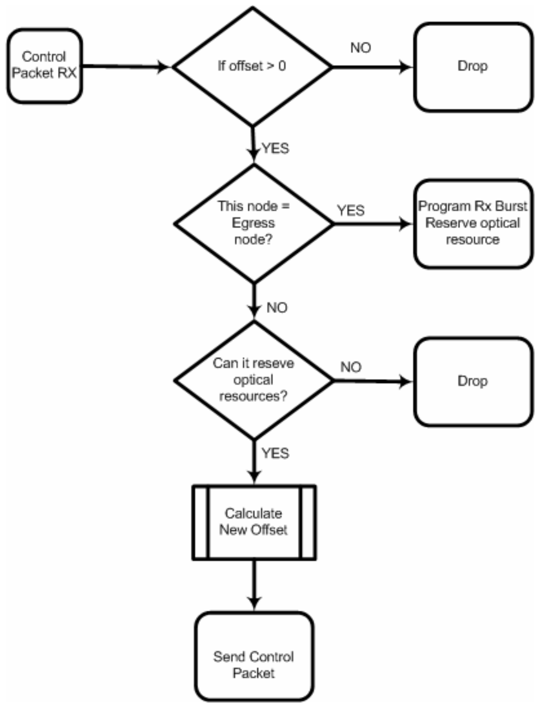

Fig. 8. Control Packet reception/forwarding algorithm.

interfaces allow the implementation of proprietary protocols and variations of standards. Therefore, this technology becomes the most appropriate to develop the control plane, in which a process of design, implementation and testing is required.

Another interesting capability of the current generation of FPGA is that they can assemble one or more RISC CPUs. This kind of CPUs are characterized by their capacity and processing power for running any type of software, which for instance could manage any wavelength reservation scheme for an OBS network.

Currently we are working with a Xilinx's Virtex II Pro FPGA [9] [10]. This FPGA model contains all the characteristics specified previously. Specifically, this chipset has RocketIO Multi-Gigabit Transceiver [11] interfaces that allow serial transmissions with rates among 622 Mb/s up to 3.125 Gb/s. It also contains 2 IBM's PowerPC 405 [12] CPUs of 32 bits that have a maximum clock frequency of 400 MHz and give a maximum capacity of 600+ DMIPS. This is enough calculation power for the purposes on the implementation of the CPS.

The development kit is Avnet's badge PCI board (Xilinx Virtex-II Pro Development Kit, see Fig. 9) that has the following components:

- Xilinx Virtex-II Pro XC2VP30 that contains

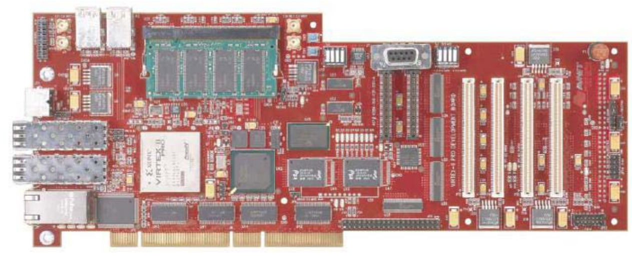

Fig. 9. Avnet's Virtex-II Pro Development Kit.

30.000 logical cells to implement the Control Path hardware.
- 2 SFP Modules for GigabitEthernet.
- 10/100/1000 GigabitEthernet physical interfaces.
- 32 MBytes of SDRAM memory.
- 128 MBytes of DDR SDRAM memory.
- Four 140-pin general purpose I/O expansion connectors (AvBus).

This kit will provide in future research sufficient capabilities to connect a Data Path by means of their extension connectors.

B. *Link and Physical Layers*

One of the decisions to adopt is which physical level protocol should be used to communicate the control plane between nodes. This protocol should provide simplicity in its implementation, as well as efficiency in transmission. A minimum speed of transmission of 1 Gb/s is also required.

The protocol that endows with these requirements is GigabitEthernet, because it has a great efficiency and it is easy to implement. Therefore the control messages defined in section II.C will be encapsulated in a GigabitEthernet frame.

C. *Control Plane's Implementation*

Next, the control plane proposal implementation explained in previous sections will be explained, specifically, how a GPC, Control Packet Interface, DPI and OXCPI modules work.

The processing of control packets is formed by two blocks: a Gigabit Packet Controller (GPC) and a Control Packet Interface (CPI). The GPC module is directly connected to GigabitEthernet MAC which offers Gigabit encapsulation to control messages. This block takes charge on de-framing the information of the OBS control protocol and writing these data in the embedded memory of the PPC405 CPU. When this process ends, the GPC generates an interruption to the processor. Then the microprocessor would request the GPC for where the control message information is stored. The GPC answers the CPU with a pointer which indicates the memory address of the control messages and the size and type of the message that has been recently received.

Analogously, in the case the CPS needs to deliver a control packet, it would advise the GPC the pointer where control data has been stored, size of this message and the OBS node's destination address. With this information GPC will create a new GigabitEthernet frame filling its data field with the control packet. GPC communicates with the PPC405 by means of the CPI. At the same time, the CPI is a module that is connected to the processor by means of the DCR bus.

This way of implementation is a recommendation of Xilinx in an Application Note [13] where they indicate that this is the best method to obtain a greater performance on packet processing. Xilinx compared this processing to other architectures on a later document [14].

This IBM's DCR bus (CoreConnect Architecture [15]) is composed by a bus of data, which is responsible for writings, and another bus that processes the readings of 32-bit width words. It also uses three signals of control (READ, WRITE, ACK) [16]. The speed of data transferring of this bus is of surroundings 1 Gb/s. The PPC405 gains access to DCR bus data by means of a reading instruction and a writing instruction. In these instructions it has to indicate the address to which data are written or from they are read. The word's width is 10 bits and it is completely independent from the memory system organization. Therefore an interval of addresses should be reserved to enable this operation between CPS and GPC.

The DPI and OXCI modules carry out the same functions of the CPI, that is to say, some addresses of the BUS DCR of the PPC405 are reserved and its mission is decoding these addresses in order to enable writings and readings to/from the DataPath's and OCX's registries.

IV. Future Research

Currently we are at the implementation phase of the Control Plane and one important goal for the immediate future is to analyze the results. An interesting point is that we not only work on simulations but we are implementing a real testbed, and so, results are expected to be more accurate to any future commercial OBS device.

In this testbed we implement our proposed control plane algorithms. These proposed algorithms have been previously simulated.

Despite we have focused on providing a transparent control plane for OBSNs, in order to correctly evaluate this plane we will also evaluate how it can be integrated with a data plane. This is necessary not only for checking the correct control plane behavior, but also for providing to the data plane accurate, fast and efficient optical paths.

V. Conclusion

In this paper we have given an overview of the OBS networks and presented architecture of the control plane that is currently being implemented. Due to the lack of protocol standards related to OBS, main efforts are designated to design, implement and test a control protocol for optical burst switching. Some problems to tackle are the synchronization between nodes in the control plane or the optimization of processing time in the CPS. This architecture focuses on providing a common framework on which to implement any kind of wavelength reservation scheme, and so, giving an interesting platform to researchers where future developments of OBS could be carried on.

Acknowledgment

This work is founded by the Spanish Ministerio de Ciencia y Tecnología (MCYT) and FEDER within the framework of the TIC2003-09042-C03-02 project, the i2CAT Foundation and the EURO-NGI project.

References

[1] C. Gauger *et al.*, "Service Differentiation in Optical Burst Switching Networks", *ITG Fachtagung Photonische Netze,* 2001, pp. 124-32.

[2] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) – A New Paradigm for an Optical Internet", *J. High Speed Nets*, vol. 8, no. 1, Jan. 1999, pp. 69-84.

[3] Fei Xue, S.J.B. Yoo, H. Yokoyama, and Y. Horiuchi, "Performance comparison of optical burst and circuit switched networks", *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, Volume 3, March 2005, pp 3.

[4] I. Baldine, G. N. Rouskas, H. G. Perros, and D. Stevenson. "JumpStart: A just-in-time signaling architecture for WDM burst-switched networks", *IEEE Communications*, 40(2), pp. 82-89, Februrary 2002.

[5] J. Y. Wei and R. I. McFarland, "Just-in-time signaling for WDM optical burst switching networks", *Journal of Lightwave Technology*, 18(12), pp. 2019-2037, December 2000.

[6] J. S. Turner, "Terabit burst switching", *Journal of High Speed Networks*, 8(1): pp. 3-16, January 1999.

[7] 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 Proc. GLOBECOM'03, vol. 5, pp. 2777-2781, San Francisco, CA, Dec. 2003.

[8] Y. Sun, T. Hashiguchi, V. Q. Minh, X. Wang, H. Morikawa and T. Aoyama. "Design and Implementation of an Optical Burst-Switched Network Testbed'', IEEE Optical Communications, vol. 3, No. 4, pp. S48-S55, Nov. 2005.

[9] Xilinx Inc., *Virtex-II Pro Product Brochure*. [Online document], [visited April 2006], Available HTTP: http://www.xilinx.com/products/virtex2p/v2p_brochure.pdf.

[10] Xilinx Inc., *Table Virtex II Pro Family Device/Package Combinations and available RocketIO and RocketIO X.* [Online document], [visited April 2006], Available HTTP: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_pro_fpgas/product_table.htm.

[11] Xilinx Inc., *RocketIO Multi-Gigabit Transceiver*, [Online document], [visited April 2006], Available HTTP: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_pro_fpgas/capabilities/rocket_io_mgt.htm.

[12] Xilinx Inc., *PowerPC 405 Processor,* [Online Document], [Visited April 2006], Available HTTP: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_pro_fgpas/capabilities/powerpc.htm.

[13] Xilinx Inc., *XAPP669 – PPC405 PPE Reference System Using Virtex-II Pro RocketIO Transceiver,* [Online document], [Visited April 2006], Available HTTP: http://direct.xilinx.com/bvdocs/appnotes/xapp669.pdf.

[14] Xilinx Inc., *XAPP664 – PLB vs. OCM Comparison Using the Packet Processor Software,* [Online document], [Visited April 2006], Available HTTP: http://direct.xilinx.com/bvdocs/appnotes/xapp664.pdf.

[15] Xilinx Inc., *Core Connect Architecture – Device Control Register Bus (DCR)*, [Online Document], [Visited April 2006], Available HTTP: http://www.xilinx.com/ipcenter/processor_central/coreconnect/coreconnect_dcr.htm.

[16] Xilinx Inc., *PowerPC 405 Processor Block Reference Guide*, Chapter2. Section "External DCR Bus Interface", pp. 102-110, [Online Document], [Visited April 2006], Available HTTP: http://www.xilinx.com/ise/embedded/ppp405block_ref_guide.pdf.