

Signaling Support for Multicast and QoS within the JumpStart WDM Burst Switching Architecture *

Ilia Baldine¹, George N. Rouskas^{†2}, Harry G. Perros², Dan Stevenson¹

¹MCNC
3021 Cornwallis Rd.
P.O.Box 12889
Research Triangle Park, NC 27709 USA
{ibaldin,stevenso}@anr.mcnc.org
<http://www.anr.mcnc.org>

²North Carolina State University
Department of Computer Science
Box 7534
Raleigh, NC 27695-7534
{rouskas,hp}@csc.ncsu.edu
<http://www.csc.ncsu.edu>

Abstract

JumpStart is a joint NCSU/MCNC research project that is developing a specification and implementation of a signaling protocol for a network environment employing optical burst switching technology. The signaling protocol operates on a just-in-time basis. In this paper we present an overview of the **JumpStart** signaling architecture, and then we describe the aspects of the protocol pertaining to the support of multicast connections and quality of service.

*This work was supported by the Advanced Research and Development Activity (ARDA), <http://www.ic-arda.org>.

[†]Correspondence Author, Phone: 919-515-3860, Fax: 919-515-7925.

1 Introduction

Wavelength division multiplexing (WDM) appears to be the solution of choice for providing a faster networking infrastructure that can meet the explosive growth of the Internet. Since this growth is mainly fueled by IP data traffic, WDM networks employing circuit switching may not be the most appropriate solution for the emerging optical Internet. Optical packet switching appears to be the optimum choice. However, with current technology [1], it is difficult to implement switching operations at the packet level. Optical burst switching (OBS) [2, 3, 4], which operates at the burst level, has the potential to lead to the design of WDM networks which are practical to implement and which can offer a viable solution to the ever increasing demand for more bandwidth.

Burst switching is a transport technique that occupies the middle of the spectrum between the well-known circuit switching and packet switching paradigms, borrowing ideas from both to deliver a completely new functionality. The unit of transmission is a *burst*, which consists of several packets sent back-to-back. The transmission of each burst is preceded by the transmission of a *control packet*, which usually takes place on a *separate signaling channel*. Unlike circuit switching, a source node does not wait for confirmation that a path has been set up; instead, it starts transmitting the data burst soon after the transmission of the control packet. We will refer to the interval of time between the transmission by the source node of the first bit of the control packet and the transmission of the first bit of the data burst as the *offset*. The control packet carries information about the burst, including the offset value, the priority of the burst, etc. The purpose of the control packet is to inform intermediate nodes of the upcoming data burst, so that they can make a routing decision and configure their fabric to switch the burst to the appropriate output port. However, in case of congestion or output port conflicts, an intermediate node may drop bursts. Thus, as in connectionless packet switching, there is no guarantee of delivery. Because it combines the advantages of circuit and packet switching, namely, out-of-band signaling and small setup delay, respectively, OBS technology holds the promise of minimizing the latency for information transfer while at the same time maximizing throughput. Therefore, it is widely believed that it can be instrumental in realizing networks in which the optical layer is in essence a *buffer-less transparent transport media* for applications.

The authors are actively involved in the **JumpStart** project [2, 5], a joint research effort between NCSU and MCNC that is addressing a number of *control plane issues* in OBS networks. Specifically, **JumpStart** is concerned with the specification and implementation of a *signaling protocol* for an OBS WDM network. The protocol operation is based on the *just-in-time* (JIT) approach which was first introduced in [6]. A recent article [2] by the same authors outlines the general aspects of the **JumpStart** signaling architecture, the message format, and the message flows for point-to-point burst transmission. In this article we present the signaling mechanisms to support (i) multicast burst communication, and (ii) the provision of quality of service (QoS) to bursts with diverse requirements.

The rest of the paper is organized as follows. In Section 2 we present an overview of the **JumpStart** signaling architecture. In Section 3 we present the aspects of the signaling protocol pertaining to multicast session establishment and tree management, and in Section 4 we discuss the signaling support for QoS. We conclude the paper in Section 5 by presenting a set of open problems and directions for future research.

2 The JumpStart Signaling Architecture

Jumpstart [2, 5] is an ARDA-supported research project between NCSU and MCNC that is investigating issues associated with *control protocols* for OBS networks. It anticipates a future technology and an economic environment where fiber networks are widely deployed and optical technology provides the means of supporting thousands of wavelengths per fiber. In this setting, lightpaths (rather than bandwidth) are a communications market commodity. The network architecture in this environment assumes that communications service converges at the optical layer provided by optical burst switches that efficiently route lightpaths between endpoints. The holding time for these lightpaths may range from a few microseconds to hours. Optical networks will be, in essence, a buffer-less transparent transport media for applications. The control protocols [2] that have been developed in the project are based on the *just-in-time* (JIT) approach [6]. JIT control protocols launch the data signal into the network moments after a request is made. Unlike traditional telephony signaling protocols, JIT does not wait for the connection to be established across the network. As a consequence, latency for information transfer is minimized.

The scope of the completed research effort within the Jumpstart project includes the development of a *specification* for a JIT signaling protocol. The defined protocol supports point-to-point [2] and multicast communications, and the control message format is optimized to permit protocol implementation in hardware. We have recently received additional funding to develop *implementations* of the specification in hardware and software, and evaluate these implementations in testbed network environments. The ultimate goal of the project is the creation of a *prototype network* which will make use of the hardware optical layer of the ATDNet (formerly MONET) testbed.

2.1 Guiding Assumptions

The basic premise of the JumpStart architecture is that data, aggregated in bursts, are transferred between end points by configuring the cross-connect fabric in the optical switches along the path just ahead of the data arrival. Switch configuration is performed by a signaling message which travels ahead of the data burst and sets up the path. Upon completion of the data transfer, the associated cross-connect state either times out or is torn down explicitly. Thus, the OBS network acts as a broker of time on each wavelength with high temporal resolution.

Our design of the JumpStart signaling architecture was guided by the following fundamental assumptions:

- *Out-of-band signaling.* The signaling channel undergoes electro-optical conversions at each node to make signaling information available to intermediate switches.
- *Data transparency.* Data is transparent to the intermediate network entities, i.e., no electro-optical conversion takes place at intermediate switches. The JumpStart architecture makes no assumptions about the type of traffic carried (e.g., in terms of data rate or signal modulation), it simply reserves time periods on wavelengths within the network.
- *Signaling protocol implemented in hardware.* To avoid bottlenecks in the control plane and to achieve operation at wire speeds, the message format (refer to Section 2.4) was designed to be easily implementable in hardware while also being flexible enough to accommodate future needs.

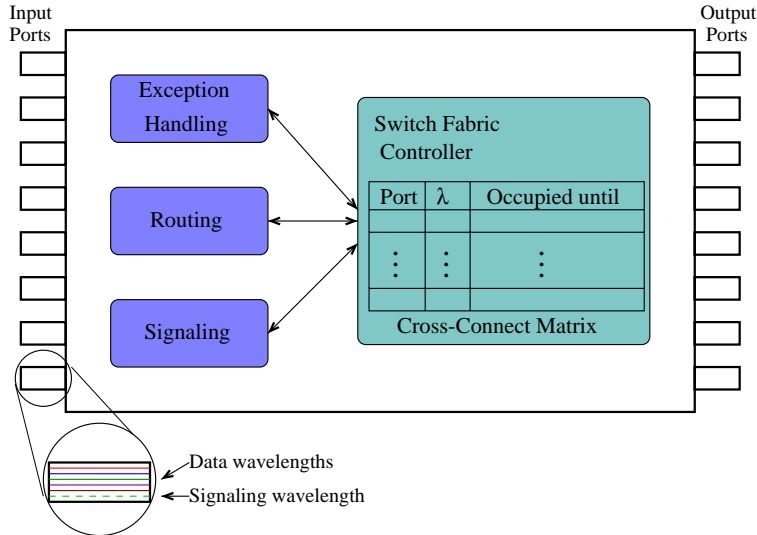


Figure 1: Logical diagram of OBS network switch

- *Network intelligence at the edge.* Most “intelligent” services are supported only by edge switches. Core switches are kept simple.
- *Explicit setup.* The cross-connect elements of each switch are configured for the incoming burst immediately after the arrival of the associated control message. The release of the cross-connect elements can be either explicit (e.g., upon receipt of a release message) or estimated (e.g., upon expiration of a timer whose value is determined from burst length information in the original control message). Explicit setup significantly simplifies the switch hardware [7] and overcomes the scalability problems of the horizon scheduling [4] approach that is required for estimated setup schemes such as JET [2, 8].
- *No global time synchronization.* In keeping with the “keeping it simple” principle, we do not assume time synchronization among nodes in the OBS network.

Figure 1 illustrates the logical view of a switch in a *JumpStart* OBS network. The fiber that terminates at each input or output port of the switch carries a (potentially very large) number of wavelengths, one of which is dedicated to carrying the signaling traffic. Any wavelength on an incoming port can be switched to either the same wavelength on any outgoing port (no wavelength conversion) or some other wavelength on any outgoing port (partial or full wavelength conversion). The switch fabric may use MEMS micro-mirror arrays as optical cross-connect elements, or it may employ some other suitable switching technology. The switch setup time (the time to configure the switch fabric to connect an input wavelength/port pair to some output wavelength/port pair) is presumed to be in the tens of μs to ms range, but it is anticipated that switch performance will improve dramatically as optical switching technology matures.

The switch fabric controller keeps track of the state of output wavelength/port pairs, and is responsible for configuring the cross-connect elements to make the appropriate connections in response to signaling protocol messages received at the switch. The controller may have to consult a next-hop table maintained by the routing protocol to determine the output port for an incoming burst. Note that since setup is explicit, cross-connect elements are not reserved for future bursts, and thus no scheduling takes place inside the switch.

Upon receipt of the control message preceding a burst, the state of the desired output wavelength/port pair is checked. If it is available, the corresponding cross-connect element is configured to make the connection; otherwise, no switching action is taken and the burst is dropped upon arrival (this discussion assumes no alternate routing and no wavelength conversion).

2.2 Functional Requirements

Within the `JumpStart` architectural framework we support the following traffic types:

- *asynchronous short bursts* with a holding time shorter than the diameter of the network, and
- *switched lightpaths* with a holding time longer than the diameter of the network.

The following functional requirements have guided the design of the architecture.

Data Transparency. Data transparency is commonly viewed as a desirable property for a core network of the future. Indeed, the ability to transmit optical digital signals of different formats and modulations, as well as analog signals, simplifies many problems commonly associated with adaptation layers. The particular format that an end-node uses to transmit its data to the destination(s) is of no consequence to the network itself. It is the responsibility of the signaling protocol, however, to assist endpoints in negotiating the data format.

Multicast Support. Within the `JumpStart` architecture we support both the source-managed and leaf-initiated multicast models. Section 3 discusses signaling support for multicast in detail.

QoS Support. While the network makes no assumptions about the traffic it carries, it has the responsibility to assist connections to obtain a desirable level of QoS; this issue is discussed in Section 4.

Persistent Path Service. For some applications there is a need for all bursts to travel the same route through the network. These applications are particularly sensitive to jitter or out-of-order delivery of messages. A persistent path service allows the network to “pin” a route that all bursts follow. Thus, in addition to on-the-fly path setup for individual bursts, the `JumpStart` architecture also provides a persistent path service to both point-to-point and multicast sessions.

Label Switching. Due to the flexible signaling message structure (described in [2]), it is possible to incorporate into the `JumpStart` architecture elements of the generalized multiprotocol switching (GMPLS) framework [9]. This can be achieved by adding properly formatted GMPLS labels to the signaling messages which establish new connections.

2.3 Signaling Protocol

Signaling in the `JumpStart` architecture is out-of-band. For instance, a single channel (wavelength) per fiber may be dedicated to signaling functions. Our design presumes that the signaling channel possesses a low bit error rate on the order of 10^{-12} to 10^{-15} . Signaling messages are queued and processed by each intermediate switch and message loss due to buffer overflow is possible. However, due to its just-in-time nature, the signaling protocol operates in a best-effort basis link-by-link, and no attempt is made to retransmit lost messages.

Session Declaration	Announce the connection to the network
Path Setup	Configure resources needed to set up an all-optical path from source to destination
Data Transmission	Inform intermediate switches of burst arrival time and length
State Maintenance	Refresh the necessary state information to maintain the connection
Path Release	Release resources taken up to maintain the lightpath for the connection

Table 1: Signaling protocol functions

As we mentioned earlier, in **JumpStart** we have adopted two approaches to determining the amount of time the switches along the path of a given burst are configured for this burst: “Explicit setup and explicit release” and “Explicit setup and estimated release.” As we discuss in [2, 7], explicit setup results in the simplest and most scalable switch controller implementations among all possible just-in-time schemes. The **JumpStart** signaling protocol supports both approaches; it is up to the caller to decide which one to use.

The signaling protocol functions are listed in Table 1. While each connection in the OBS network has to go through a series of the phases defined in Table 1, the signaling protocol is flexible in that a number of the phases can be combined into a single step. For example, depending on the type of connection being set up, a SETUP message (defined shortly) may serve to: (1) announce the session to the network (Session Declaration), (2) set up the path of the session (Path Setup), and (3) announce the arrival of the burst (Data Transmission). Combining the first three phases in Table 1 into one is useful for speeding up the transmission of short bursts. However, doing so may not be appropriate for long-lived connections, multicast sessions, or applications requiring persistent path service. For these, the path setup phase must be separate from the data transmission phase, as we discuss in Section 3.

Both multicast and point-to-point connections can be mapped onto the phases listed in Table 1. Note that in order to facilitate robust network operations, additional protocols may need to be defined in the future, for example, a routing protocol. While the message flows for new protocols will be different from the ones defined for connection setup, these protocols will use the message structure described in [2].

Connections in a **JumpStart** network can be classified along the following dimensions:

- point-to-point vs. multicast
- short bursts vs. lightpaths
- timed vs. explicit release (pertains to data transmission phase only)
- persistent vs. on-the-fly path setup

In general, any combination is allowable (e.g., point-to-point short burst with timed release and on-the-fly path setup), except that multicast connections must always use persistent path setup.

The basic signaling messages used to set up a point-to-point connection in a **JumpStart** OBS network are listed and explained in Table 2. For the sake of completeness, in the remainder of this section we describe the message flows required for the point-to-point transmission of a short burst with on-the-fly path setup.

Message Name	Message Function	Connection Phases
SESSION DECLARATION	Notifies the network that a persistent-path point-to-point or a multicast connection is being set up.	Session Declaration Path Setup
SETUP	Notifies the network that a burst is arriving. Carries the burst length and delay information in the “timed release” scheme. Can be used to combine path setup with data transmission	Session Declaration Path Setup Data Transmission
SETUP_ACK	Sent from the ingress switch to the calling party. Acknowledges the SETUP message and returns the burst delay estimate.	Data Transmission
DECLARATION_ACK	Acknowledgment of SESSION DECLARATION by the called party	Session Declaration
CONNECT	Returned by the called party to the calling party to acknowledge path setup (optional)	Data Transmission
SESSION RELEASE	Releases the path of a connection previously established by SESSION DECLARATION	Session Release
RELEASE	Informs intermediate nodes that connection is released (“explicit release” scheme only)	Session Release Data Transmission
KEEPALIVE	State refresh message, periodically sent by the calling party. Resets timeout counters. Optionally carries the remaining duration of the connection for “timed release” scheme.	State Maintenance
FAILURE	Indicates general failure of connection setup	Any

Table 2: Basic message types

For a description of the sequence of messages involved in transmitting long bursts and in persistent path setup for point-to-point communication, the reader is referred to [2].

With on-the-fly path setup, a path is established for the duration of a *single* burst only; consecutive bursts from the same source to the same destination may thus take different routes through the OBS network since their routes are determined independently. For these connections, the Session Declaration, Path Setup, and Data Transmission phases are combined into a single message. The sequence of messages is presented in Figure 2. The connection is initiated with a SETUP message sent by the source of the burst to its ingress switch. The ingress switch uses a delay estimation mechanism to determine an appropriate delay (offset) value for the incoming burst; this value is based on (i) congestion information available at the switch and (ii) the destination address in the SETUP message. The ingress switch then transmits a SETUP ACK to the source node to acknowledge the receipt of the SETUP message by the network. The SETUP ACK message also includes the burst delay information and informs the source about the channel (wavelength) to use when sending the data burst.

The source node waits the required balance of time left based on its knowledge of the round-trip time to the ingress switch, and then sends the burst on the indicated wavelength. At the same time, the SETUP message is traveling across the network informing the switches on the path of the burst arrival. If no blocking occurs along the path, the SETUP message eventually reaches the destination node; the data burst

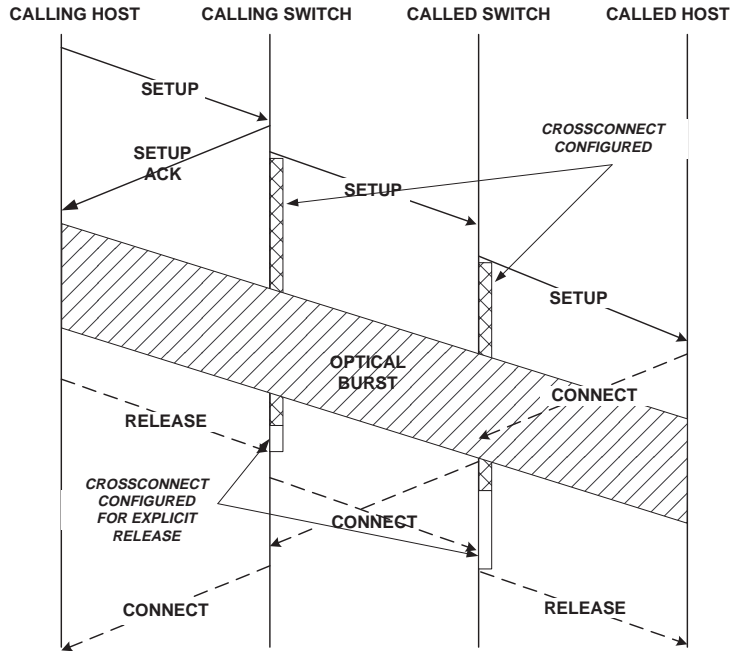


Figure 2: Signaling flows for on-the-fly path setup of a short point-to-point burst

follows shortly thereafter. Upon the receipt of the **SETUP** message, the destination node may choose to send a **CONNECT** message acknowledging the successful connection. (The receipt of the **SETUP** by the destination only guarantees that the path has been established; it does not guarantee successful receipt of the burst, since a connection may be preempted somewhere along the path by a higher-priority connection. The use of preemption is discussed in Section 4.)

If explicit release is selected, the source node sends a **RELEASE** message immediately after the completion of the burst transmission. In timed release, on the other hand, no such message is needed: the intermediate switches configure their cross-connects to automatically release the connection based on burst length information present in the original **SETUP** message. Note that to guard against lost **RELEASE** messages, the switches may associate a time-out value with each burst. Therefore, a source transmitting a very long burst must periodically send **KEEPALIVE** messages to the network to prevent the switch state from timing out (not shown in Figure 2).

To conserve space we do not show the flow of messages when failures occur during any phase of the connection. In general, any node detecting a failure sends a **FAILURE** message to the source node and includes the cause of the failure, e.g., blocking, preemption by a connection of higher priority, lack of route to host, refusal by destination, etc.

2.4 Signaling Message Format

Figure 3 shows the structure of the **JumpStart** signaling messages. We designed this new message format to be easily implementable in hardware, and flexible enough for future extensions. It consists of three parts:

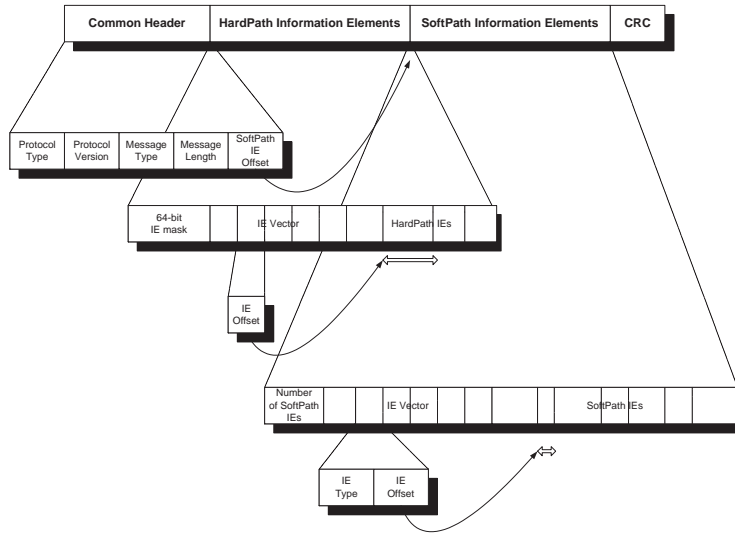


Figure 3: Signaling message structure

a common *header*, a number of *hardpath* information elements (IEs), and a number of *softpath* IEs. IEs are classified as hardpath or softpath, depending on whether they are processed by hardware or software, respectively. The structure of both hardpath and softpath IEs is a [type, length, value] (TLV) triple. This feature allows for future migration of IEs from softpath to hardpath as hardware matures and is capable of processing more complex functions.

The subheaders provide information about the IEs present in the message. The hardpath subheader contains a bit mask such that the value of each bit corresponds to the presence/absence of a specific IE type. The hardware can thus parse the subheader and determine which IEs are present, as well as recognize invalid IE combinations. There is also a variable length vector of offsets for each IE present in the subheader, so that the IEs can be easily identified inside the message and parsed. The softpath subheader simply contains the number of IEs present and a vector of <type, offset> tuples for each IE. Thus, the software must scan the entire vector before it can determine the types of IEs present in the softpath header.

The common header has information about the specific signaling protocol (e.g., connection setup, routing, etc.) to which the message belongs, the protocol version used to create the message, the message type and overall length, and the offset to the beginning of the softpath subheader. Each message also includes a CRC-32 sequence for integrity verification.

3 Signaling Support for Multicast

3.1 Optical Multicast

The JumpStart project envisions a future backbone network environment in which multicast in the optical domain is made possible by multicast capable optical switches [10, 11]. These devices will have the ability to split an incoming optical signal into two or more outgoing signals that are independently switched to

different output ports. Optical multicast switches will provide a greater functionality than today's optical drop-and-continue devices, including the ability to control the fanout of the split signal and, possibly, the ability to all-optically convert each outgoing signal to a wavelength different than the incoming one. At the same time, optical multicast switches are expected to be quite expensive, since there are significant costs associated with the splitting and amplification functions (specifically, the EDFAs needed to amplify each of the outgoing signals) that a multicast switch must support. Consequently, it is conceivable that, at least initially, not all switches in an OBS network will be multicast capable.

Support for multicast traffic in point-to-point mesh networks introduces a number of complex issues, including the construction of trees over which this traffic is forwarded. These issues have been extensively studied in the context of packet-switched networks, including IP and ATM networks, and a number of algorithms and protocols have been designed to deal with the complexities of multicast [12].

In OBS networks (as well as circuit-switched optical networks) employing multicast switches, the routing tree problem becomes even more difficult for several reasons. Splitting an optical signal introduces losses, a problem not encountered in electronic packet- or circuit-switched networks, and thus, not addressed by existing routing tree algorithms. Even in the presence of optical amplifiers, this signal loss imposes a hard upper bound on the number of times a signal can be split, as well as on the number of hops that the signal can travel after every split operation. In the absence of wavelength conversion in the network (or even in networks with limited or sparse conversion capability), multicast routing is tightly coupled to wavelength allocation, an issue that does not arise in current networks. Also, OBS networks may only have a sparse multicast switching capability, i.e., only a subset of the switches will be multicast capable. Finally, the problems of capacity planning of multicast switches and multicast routing strongly depend on one another. Although these research problems have attracted attention recently [11, 13, 14], including recent and ongoing work by some of the authors as part of the `JumpStart` project [15] there are many unanswered questions yet to resolve.

While multicast routing in optical networks remains an open research issue, any solutions to be developed will require signaling support. Therefore, one of the primary design goals of the `JumpStart` project was to address the signaling aspects of multicast connections in OBS networks in which multicast in the optical domain is provided by specialized switches. In this section we present the signaling mechanisms provided within the `JumpStart` architecture in support of native multicast. The mechanisms were designed independently of any particular routing tree algorithm, and we believe that they are sufficiently general that they can be utilized by a wide range of such algorithms. We also note that while the scope of this work only included multicast session establishment and tree management, any additional protocol that might need to be defined (e.g., a routing protocol) can use the signaling message format described in Section 2.4 and simply introduce new message types.

Before we proceed, a discussion of the assumptions that guided our design is warranted. As we have already mentioned, we presume that multicast switches capable of splitting the optical signal are not common in the network, i.e., multicast functionality is sparsely available. Multicast switches are also responsible for managing and routing multicast connections, and for this purpose they are equipped with specialized firmware or software; non-multicast-capable switches are not involved in any signaling operations for multicast. Each end node is assigned to one such switch as its multicast server; this assignment is made either via an administrative procedure or a separate signaling mechanism. Consequently, all signaling messages from a

source node that pertain to its multicast connections, or from a destination node wishing to join a multicast session, are routed by the network to the multicast switch assigned to that node.

3.2 Multicast Tree Setup

Let us return to the signaling protocol functions listed in Table 1. Recall that for point-to-point connections consisting of short bursts with on-the-fly path setup, the first three phases (Session Declaration, Path Setup, and Data Transmission) are combined into a single phase. However, for multicast connections there must be a clear separation between the three phases due to the need to establish a multicast routing tree before any data bursts can be sent. Furthermore, since all bursts within a given multicast session are routed over the same tree, multicast connections in a **JumpStart** OBS network always use the persistent path (tree) service.

In this subsection we discuss the supported multicast models and the signaling flows for the Session Declaration, Path Setup and Path Release phases. The next subsection discusses issues pertaining to the data transmission phase of multicast connections.

3.2.1 Multicast Models

There are two generally accepted and well-understood models for setting up a multicast session:

- *Source-managed multicast.* The source of the multicast exercises strict control over the membership in the multicast group. Typically, the source uses a signaling mechanism to inform the network of the address of each multicast group member. The network uses this information to establish a routing tree. The tree remains in place until the source explicitly terminates the multicast session. This model requires the source to know the addresses of all destinations, and usually is limited to static sessions involving a small set of members.
- *Leaf-initiated join.* This model is appropriate for dynamic sessions with a possibly large set of potential members, in which each node may join or leave the session at will and independently of other nodes. Typically, the source announces the start of a new multicast session to the network using a unique session identifier. An end node wishing to join an existing multicast session contacts its assigned multicast switch which takes the necessary steps to connect the end node to the appropriate routing tree. Note that the multicast tree for a given session is dynamically adjusted in response to join or leave requests by individual nodes. In particular, no tree is set up unless at least one end node (besides the source) has joined the session, and bursts transmitted by a source to an empty multicast group are dropped at the first multicast capable switch.

In the **JumpStart** architecture we also support a model that combines features from both the source-managed and leaf-initiated models.

- *Hybrid model.* A source announces a new multicast session to the network, and at the same time it includes in the signaling message an initial set of destination nodes to include in the routing tree. The source also indicates to the network that other nodes may join the multicast group on their own, and/or that the nodes in the initial set may leave the group before the end of the session. In essence, this model introduces a leaf-initiated join capability to source-managed multicast sessions.

Message Name	Message Function
ADD PARTY	Requests that the network add another end node to the multicast routing tree. Can be sent by the source or by the node wishing to join.
ADD PARTY ACK	Acknowledges the ADD PARTY message to its originator.
DROP PARTY	Requests that the network remove a node from the multicast tree. Can be sent by the source or by the node wishing to be dropped.
DROP PARTY ACK	Acknowledges the DROP PARTY message to its originator.

Table 3: Additional message types in support of multicast connections

It can be seen that the hybrid model is a generalization of both the source-managed and leaf-initiated join multicast models. Specifically, if the initial set of destination nodes provided by the source to the network is empty, then the session operates under a pure leaf-initiated join multicast. At the other extreme, if the initial set is nonempty and the source specifies in the signaling message that no nodes can be added to or deleted from the initial multicast tree, we have a pure source-managed multicast. Between these two extremes, we have a wide range of sessions that begin under the source-managed multicast model but continue under the leaf-initiated join model.

3.2.2 Signaling Flows for Session Establishment and Tree Management

Our objective for the `JumpStart` architecture was to support the richest possible functionality for multicast sessions, including both the source-managed and leaf-initiated join models, as well as combinations thereof. At the same time, it was crucial to keep the signaling mechanism for multicast sessions as simple as possible in order to facilitate protocol implementation in hardware. Therefore, we have decided to adopt the hybrid multicast model and treat all multicast sessions in an identical manner. Doing so allowed us to consolidate the signaling flows in as few messages as possible, and eliminated the need to introduce different sequences of messages to support the same function for different types of multicast sessions.

All multicast sessions in `JumpStart` have to go through the same five phases as point-to-point connections, namely, Session Declaration, Path Setup, Data Transmission, State Maintenance, and Path Release (refer to Table 1). Signaling of multicast connections requires only four message types in addition to those listed in Table 2. These new messages (ADD PARTY, DROP PARTY, and the corresponding acknowledgments) are needed to handle join and leave operations on the multicast group (and corresponding tree), as explained in Table 3. A field in the `SESSION DECLARATION` message allows the source of the multicast to specify whether only it has the authority to add nodes to the multicast tree. If this field is set, leaf-initiated joins are prohibited, and the session is purely source-managed. On the other hand, if the source never initiates any ADD PARTY messages, then the session follows the pure leaf-initiated join model. In all other cases, the multicast session follows the more general hybrid model. An optional *session scope* field in the `SESSION DECLARATION` message can be used to limit the availability of the session to nodes belonging only to specific domains. This field indicates to the OBS network nodes which domains are allowed to join the multicast session, regardless of whether nodes are added by the source or through the leaf-initiated join process.

Figure 4 shows the sequence of messages involved in the Session Declaration, Path Setup, and Path

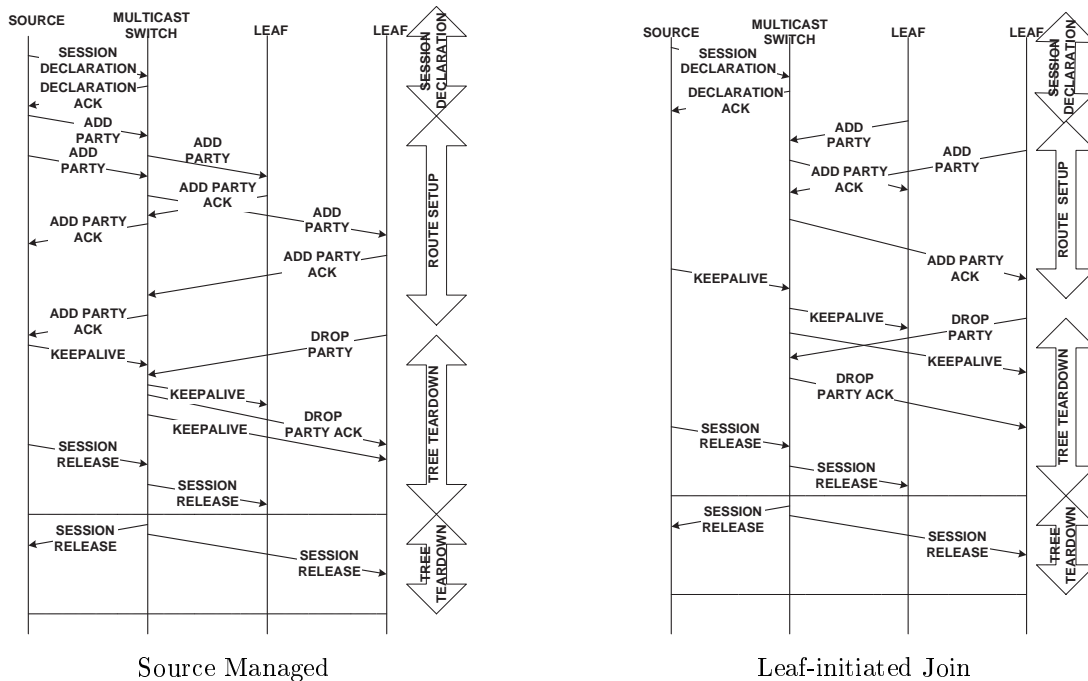


Figure 4: Signaling flows for multicast tree setup and teardown

Release phases of a multicast connection. The diagram on the left corresponds to a purely source-managed multicast session, while the diagram on the right corresponds to a pure leaf-initiated join multicast scenario. From these two diagrams, it should be relatively straightforward to visualize the signaling flows for a session following the more general hybrid multicast model.

Let us first concentrate on the left diagram of Figure 4 which shows the signaling flows for a source-managed multicast session. The source begins by sending a `SESSION DECLARATION` message with the appropriate options set. This message is automatically routed to the nearest or assigned multicast switch, and it informs the network of the multicast session being established. The network responds with a `DECLARATION ACK` message to acknowledge the receipt of the `SESSION DECLARATION` message. This message exchange completes the Session Declaration phase.

During the second phase, Path Setup, the multicast routing tree is established. Since this is an instance of source-managed multicast, the source notifies the network of the multicast group members by originating a number of `ADD PARTY` messages which are also routed to the associated multicast switch. Each `ADD PARTY` message attempts to add a new node to the tree, and is forwarded by the multicast switch to the corresponding destination node. The destination node acknowledges the receipt of this message by returning an `ADD PARTY ACK` message to the source of the multicast. The multicast switches in the OBS network communicate using a multicast routing protocol to build a routing tree spanning the source node and the destination nodes indicated in the `ADD PARTY` messages. Note that the addition of a new node to the tree must take into account the power budget of the optical signal, such that the quality of service of the connection to all the leaves may be maintained. (The actual routing protocol is outside the scope of this work, but it can use the same message format as the signaling protocol, described in Section 2.4.) If for

some reason the network fails to add a destination node to the tree, the multicast switch which detects the failure returns a `FAILURE` message to the source (not shown in the figure). It is up to the source to decide whether to proceed with the multicast session in that case.

Once the routing tree has been established, the connection enters the third phase, Data Transmission, and the source may start transmitting data bursts. The signaling messages involved in the Data Transmission phase are discussed in the following subsection.

During data transmission, any destination node may decide to leave the session by sending a `DROP PARTY` message. Removal of a destination node may involve the update of the multicast tree used for routing data bursts; this functionality is provided by the multicast routing protocol and does not affect the Data Transmission phase. Also, for long multicast sessions the source has to transmit periodic `KEEPALIVE` messages which are forwarded along the branches of the tree to the destination nodes (State Maintenance phase). The purpose of these messages is to refresh the path (tree) state maintained within the network for this multicast session, in order to prevent it from timing out (more on this shortly).

The session can be torn down either by the network or by the source using `SESSION RELEASE` messages. This is illustrated in the diagrams of Figure 4 where two possibilities for the Session Release phase are shown, one in which the `SESSION RELEASE` message originates at the source of the multicast, and one in which it originates at the multicast switch. The reason for a network-originated `SESSION RELEASE` message is to ensure that the session is released when all destination nodes have dropped from the session, or when a network failure occurs.

As we can see in the right diagram of Figure 4, the Session Declaration, Data Transmission, Session Maintenance, and Session Release phases of a multicast session following the pure leaf-initiated join model are identical to those of a source-managed session. The main difference is in how the routing tree is set up. Specifically, we assume that the network provides a mechanism for end nodes to learn about ongoing multicast sessions. While the details of such mechanism are outside the scope of this paper, one possible implementation would be for multicast switches to maintain a directory service of multicast sessions that is updated every time a `SESSION DECLARATION` or `SESSION RELEASE` message is sent. End nodes query this directory service for connections of interest, and submit an `ADD PARTY` message to their (local) multicast switch to join a specific session. The local multicast switch contacts its network peers to include the new destination to the routing tree, and acknowledges the `ADD PARTY` message by sending an `ADD PARTY ACK` message to the destination node. (Note that in Figure 4 it is assumed that the source and both destination nodes are assigned to the same multicast switch. This assumption is not valid in general, but is made here to simplify the presentation.)

As the reader may surmise, a session following the hybrid multicast model (i.e., one that combines the features of source-managed and leaf-initiated join multicast) will differ from the two scenarios described here only in that the `ADD PARTY` messages during the Path Setup phase may originate at either the source or individual end nodes wishing to join the session.

As we mentioned earlier, multicast connections in `JumpStart` always use a persistent path service. In other words, once a routing tree is established, it remains for the duration of the session (i.e., until a `SESSION RELEASE` message is sent). This does not mean that the routing tree does not change, since the addition or removal of destination nodes may cause routing path changes. It simply means that the *routing state* associated with the multicast session is maintained inside the network. The periodic `KEEPALIVE` messages

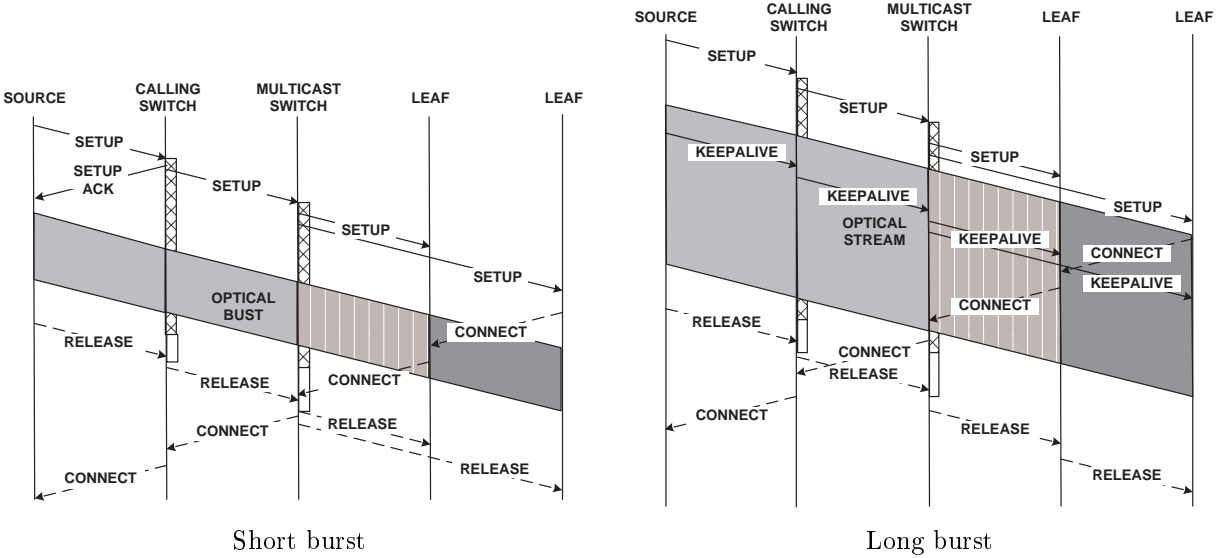


Figure 5: Signaling flows for multicast burst transmission

sent by the source are forwarded along the (current) routing tree to prevent multicast switches from flushing the corresponding routing state.

Finally, we emphasize that the cross-connect elements at intermediate switches are *not* permanently configured for the routing tree between the arrivals of the SESSION DECLARATION and the SESSION RELEASE messages. Rather, the cross-connect configuration necessary to route a burst to the destinations is simply cached by the multicast switches when the tree is built (or when it changes). Then, each time a SETUP message announces the arrival of a new burst for the connection (refer to the next section), this cached information is used to configure the switch while the arrival of a RELEASE message makes the cross-connect elements available for other bursts. Therefore, bursts belonging to the same connection are guaranteed to take the same path, but they are not guaranteed to be successful (i.e., a burst may be blocked if the cross-connect elements at an intermediate switch are not free when its SETUP message arrives).

3.3 Signaling Flows for Data Transmission

Figure 5 depicts the signaling messages exchanged for the transmission of a *single* burst during the Data Transmission phase of a multicast session. If the source wishes to send multiple bursts, the same sequence of signaling messages must be repeated for each burst. Two scenarios are shown in Figure 5, one in which the burst is short (diagram on the left) and one in which the burst is long (diagram on the right). The signaling flow in the short burst case is identical to that for an on-the-fly transmission of a short point-to-point burst shown in Figure 2. However, there are some important differences between Figures 2 and 5. First, the multicast switch assigned to the source of the multicast may not be the same as the ingress switch to which the source node is attached (the same is true for destination nodes). Thus, as Figure 5 shows, the ingress switch must forward the SETUP message to the assigned multicast switch. Second, the optical signal originating at the source is split at the multicast switch and each of the resulting signals takes a different path to different destination nodes. It is possible, therefore, for one of the paths to be unavailable at the

time of the burst transmission, in which case a subset of the destination nodes will not receive the burst (this case is not shown in Figure 5).

The diagram on the right corresponds to the transmission of a long burst. As we can see, as far as signaling is concerned, the only difference from a short burst transmission is the additional KEEPALIVE messages that the source has to send periodically. The purpose of this message is two-fold: it prevents the cross-connect configuration at intermediate switches from timing out before the burst transmission is complete, while at the same time it refreshes the routing state at the intermediate switches, as discussed in the previous subsection.

4 Signaling Support for Quality of Service

While the precise nature of applications that will make use of the OBS networks of the future is not known, it is safe to assume that several classes of applications will emerge, each with different expectations from the network service. Therefore, OBS networks must be designed to offer various levels of service to accommodate applications with diverse QoS requirements. While the details of the mechanisms used by the network to provide a certain level of QoS is outside the scope of the JumpStart project, the signaling protocol must provide a means for applications to communicate their required QoS to the network. In this section we discuss the QoS parameters that are relevant in an OBS network and how they are conveyed by the JumpStart signaling protocol.

In the well-known ATM, Diffserv, and Intserv QoS architectures, the focus is on parameters such as peak or average rate, delay, and packet loss. In these architectures, the network guarantees a given level of service to applications mainly by employing two mechanisms: buffer allocation strategies and link scheduling disciplines which select one of multiple buffered packets for transmission. In a transparent, buffer-less optical network environment such as the one under consideration, each burst captures the whole wavelength channel for its duration. Bursts also travel at the speed of light, experiencing little delay beyond the one-way propagation delay. Furthermore, physical layer impairments (which are not an issue in opaque networks) accumulate along the burst's path, and must be accounted for explicitly.

Based on these observations, we distinguish two sets of QoS parameters that are important to applications in an OBS network. The first set of parameters is related to the quality of the optical signal as it travels from the source to the destination node(s). The second set is concerned with the level of guarantee that a given burst transmission will be successful, i.e., that it will not be dropped at intermediate switches due to lack of resources (output ports and/or wavelengths). Therefore, our signaling protocol provides for the following parameters to be communicated between a source node and its ingress switch:

1. Maximum channel bandwidth
2. Channel bit error rate (BER)
3. Channel dynamic range (for analog signals)
4. Channel signal to noise ratio (SNR)
5. Channel spacing (to control the crosstalk for particularly critical streams)

Field Name	Field Type	Field Length	Notes
Bandwidth requirement	Floating Point	32	Bandwidth requirement in bytes per second
Priority	Integer	8	Priority value of the session
SNR	Integer	16	dB
Bit-error rate	Integer	8	Measured in $-\log_{10}\text{BER}$, i.e., a BER of 10^{-15} yields a value of 15
Dynamic Range	Integer	16	dB/Hz (typical values 100-200)
Channel Spacing	Integer	32	Hz
Wavelength Conversion	Boolean	8	Wavelength conversion is/is not allowed for the connection

Table 4: QoS information element

6. Priority of the connection

The first five parameters are directly related to the desired optical signal quality and affect the choice of path for a particular connection. More specifically, the first four parameters relate to intrinsic channel properties, while the fifth parameter refers to a WDM link. All five parameters must be considered when setting up a connection since both the links of the path and the relationship to other channels are affected by the desired QoS. Note that selecting a path that meets a certain QoS as expressed by these five parameters may involve a constrained-based routing algorithm [16]. However, the details of such an algorithm are outside the scope of this work.

The last QoS parameter (connection priority) will be used to guarantee a certain level of assurance that a given burst transmission will be successful. In other words, the priority is directly related to the burst blocking probability that a connection experiences. It is envisioned that the priority will be used in conjunction with a preemption mechanism employed at OBS network switches. Specifically, a switch may be allowed to preempt (drop) an ongoing burst upon the arrival of a higher priority burst that has to use the same output port/wavelength. Currently, the details and actual use of this preemption mechanism are a subject of further study. We note that a different method for prioritizing bursts based on the offset value is discussed in [17]; however, we plan to use the offset for a different purpose as discussed in the next section.

The **Jumpstart** signaling protocol uses a flexible message format based on the information element (IE) concept also used in ATM signaling, appropriately modified for hardware implementation (for details, refer to [2]). A special QoS IE, supplied along with a **SESSION DECLARATION** message sent by a source node to its ingress switch, is used to describe the application's requirements in terms of the QoS parameters discussed above. This QoS IE whose format is shown in Table 4, is also copied in the **SETUP** messages that set up the path for the connection. The IE can be easily expanded to include additional QoS parameters in the future.

5 Conclusions and Future Work

The ARDA-funded **JumpStart** project is addressing a number of control plane issues in OBS networks, including the specification of an OBS signaling protocol based on the just-in-time approach. In this paper we presented the part of the **JumpStart** signaling protocol that provides support for multicast communication and the provision of QoS. The complete specifications document can be downloaded from the project website [5]. While our work so far has produced solid results, research in networks based on OBS technology is still in its infancy, with many open problems and many promising directions. Ongoing activities within the **JumpStart** project include the implementation of the signaling specification in software and hardware, with the ultimate goal being the deployment of a prototype network within the ATDNet testbed. At the same time, the authors have several research efforts under way that explore performance and algorithmic issues that arise in the data plane of OBS networks. In particular, we are working on: (i) a queueing model of an OBS network with a view to estimating performance measures, such as throughput and burst dropping probabilities, for various input loads, (ii) a novel feedback-based scheme to implement congestion control in OBS networks by dynamically setting the offset of each burst, and (iii) point-to-point and multicast routing and wavelength allocation algorithms that take into account the unique feature of OBS networks. We expect that the experience we gain from implementing and testing the **JumpStart** signaling protocols will provide invaluable insight into developing solutions to these research problems.

Acknowledgments

The authors would like to acknowledge the invaluable input and comments from Ray McFarland of the National Security Agency, Laboratory for Telecommunications Sciences, Linden Mercer of the Navy Research Labs, and Paul Franzon and Pronita Mehrotra of the Electrical Engineering Department, North Carolina State University.

References

- [1] S. Yao, S. Dixit, and B. Mukherjee. Advances in photonic packet switching: An overview. *IEEE Communications*, 38(2):84–94, February 2000.
- [2] 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):82–89, February 2002.
- [3] C. Qiao and M. Yoo. Optical burst switching (OBS)-A new paradigm for an optical Internet. *Journal of High Speed Networks*, 8(1):69–84, January 1999.
- [4] J. S. Turner. Terabit burst switching. *Journal of High Speed Networks*, 8(1):3–16, January 1999.
- [5] The JumpStart project. <http://jumpstart.anr.mcnl.org>.
- [6] J. Y. Wei and R. I. McFarland. Just-in-time signaling for WDM optical burst switching networks. *Journal of Lightwave Technology*, 18(12):2019–2037, December 2000.
- [7] P. Mehrotra, I. Baldine, D. Stevenson, and P. Franzon. Network processor design for use in optical burst switched networks. In *Proceedings of the International ASIC/SOC Conference*, September 2001.
- [8] M. Yoo and C. Qiao. Just-enough-time (JET): A high speed protocol for bursty traffic in optical networks. In *IEEE/LEOS Technol. Global Information Infrastructure*, pages 26–27, August 1997.
- [9] P. Ashwood-Smith *et al.* Generalized MPLS – signaling functional description. IETF Draft <draft-ietf-mpls-generalized-signaling-06.txt>, April 2001. Work in progress.
- [10] L. H. Sahasrabudde and B. Mukherjee. Light-trees: Optical multicasting for improved performance in wavelength-routed networks. *IEEE Communications*, 37(2):67–73, February 1999.
- [11] M. Ali and J. Deogun. Power-efficient design of multicast wavelength-routed networks. *IEEE Journal on Selected Areas in Communications*, 18(10):1852–1862, 2000.
- [12] M. Ammar, G. Polyzos, and S. Tripathi (Eds.). Special issue on network support for multipoint communication. *IEEE Journal Selected Areas in Communications*, 15(3), April 1997.
- [13] X. Zhang, J. Y. Wei, and C. Qiao. Constrained multicast routing in WDM networks with sparse light splitting. *Journal of Lightwave Technology*, 18(12):1917–1927, December 2000.
- [14] M. Ali and J. Deogun. Allocation of splitting nodes in wavelength-routed networks. *Photonic Network Communications*, 2(3):245–263, August 2000.
- [15] Y. Xin, G. N. Rouskas, and H. G. Perros. Light-tree routing under optical layer power budget constraints. Submitted for publication.
- [16] J. Strand, A. L. Chiu, and R. Tkach. Issues for routing in the optical layer. *IEEE Communications*, pages 81–96, February 2001.
- [17] M. Yoo, C. Qiao, and S. Dixit. QoS performance of optical burst switching in IP-over-WDM networks. *Journal on Selected Areas in Communications*, 18(10):2062–2071, October 2000.