JumpStart: A Just-in-Time Signaling Architecture for WDM Burst-Switched Networks

Ilia Baldine, MCNC
George N. Rouskas, North Carolina State University
Harry G. Perros, North Carolina State University
Dan Stevenson, MCNC

ABSTRACT

We present an architecture for a core dWDM network which utilizes the concept of optical burst switching coupled with a just-in-time signaling scheme. It is a reservation-based architecture whose distinguishing characteristics are its relative simplicity, its amenability to hardware implementation, and the ability to support multicast natively. Another important feature is data transparency — the network infrastructure is independent of the format of the data being transmitted on individual wavelengths. We present the signaling protocol designed for this architecture, as well as an unified signaling message structure to be used in conjunction with the protocol. We also present the future directions of this research.

INTRODUCTION

The adoption of dynamic wavelength-division multiplexing (dWDM) as the primary means for transporting data across large distances in the near future is a foregone conclusion, since no other technology can offer such vast bandwidth capacities. What is being studied in the scientific community and industry are the various architectural frameworks that take advantage of the specific properties of dWDM as a medium. The current dominant technology for core networks appears to be wavelength routing with permanent or semi-permanent circuits set up between endpoints for data transfer. Some of the emerging technologies include optical packet switching with various approaches applied to allow parsing headers electronically.

In this article we present an overview of an architecture and a signaling protocol for a core dWDM network. The architecture is based on wavelength routing and burst switching. Burst switching, as opposed to circuit or packet switching, implies that the network is capable of switching data in variable-sized parcels. Signaling is just-in-time (JIT), indicating that signaling messages travel slightly ahead of the data they describe. Signaling is out of band, with signaling packets undergoing electro-optical conversion at every hop. Data, on the other hand, travels transparently.

The work we present in this article is being performed as part of the JumpStart project [1]. JumpStart is a joint North Carolina State University (NCSU)/MCNC effort supported by the Advanced Research and Development Agency (ARDA) that is investigating issues associated with control protocols for optical burst-switched networks. The eventual 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.

Burst switching refers to a principle first proposed within the Highball [2] project; the concept can also be traced to asynchronous transfer mode (ATM) block transfer (ABT) standardized by the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T). The idea is to use no buffering inside the network and to switch variable-sized bursts on the fly using a reservation mechanism; intermediate switches are only configured for a brief period of time, just enough to pass the burst, and are available to switch other bursts immediately after. The main difference from the packet switching paradigm is the lack of buffering and the much wider range of burst lengths, from very short (i.e., packets), to very long (i.e., circuits).

JIT signaling approaches to optical burst switching (OBS) have been previously studied in the literature [3–8]. These approaches are characterized by the fact that the signaling messages are sent just ahead of the data to inform the intermediate switches. The common thread is the elimination of the round-trip waiting time

0163-6804/02/$17.00 © 2002 IEEE

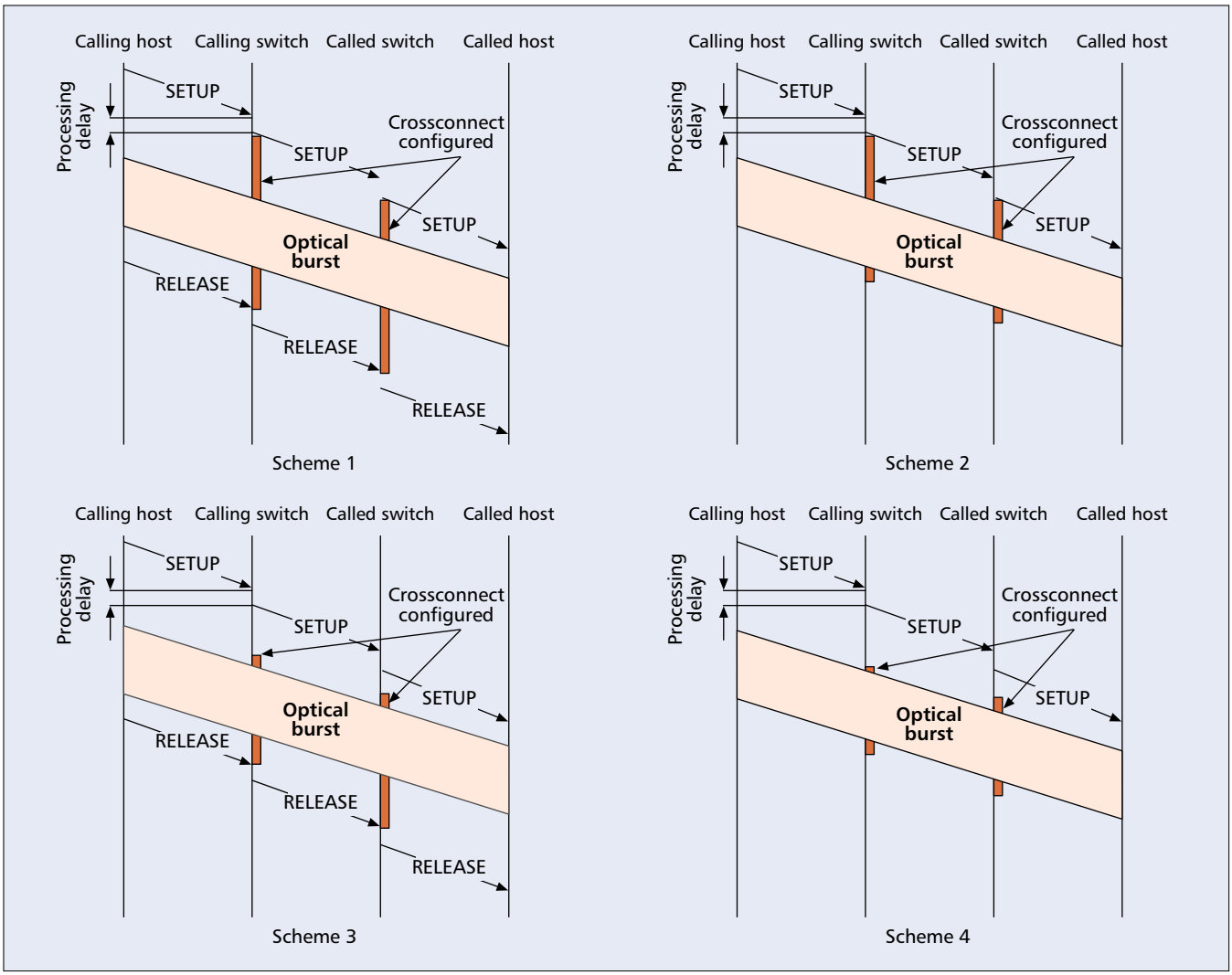

Figure 1. *Signaling schemes for optical burst switching.*

before the information is transmitted (the so-called tell-and-go approach): the switching elements inside the switches are configured for the incoming burst as soon as the first signaling message announcing the burst is received. The variants on the signaling schemes mainly differ in how soon before the burst arrival and how soon after its departure the switching elements are made available to route other bursts. An example is the Just-Enough-Time (JET) scheme proposed in [5], which uses information to predict the start and the end of the burst, thus reserving the switching elements needed to route the burst inside a switch for the shortest amount of time possible. Schemes have also been proposed for introducing quality of service (QoS) into the architecture by varying the delay between the signaling message and the burst, thus increasing the probability of successful transmission of the burst through the network [7]. While it has been shown that predictive reservation schemes like JET have a positive effect on the overall burst blocking probability in the network, this effect comes at a price of higher complexity at the scheduler in the switch.

Figure 1 demonstrates the main differences between various JIT reservation schemes as discussed in the OBS literature. Each of the illustrations depicts a burst passing through an OBS switch. Each burst is preceded by a SETUP signaling message that arrives at the switch shortly prior to the burst on the out-of-band signaling channel. The burst may be followed by a RELEASE message that signals the end of the burst. Figure 1 compares four different schemes:

Scheme 1: Explicit setup and explicit release, in which the switching elements inside the switch are configured for the incoming burst immediately after the arrival of the SETUP message, and remain in that configuration until a RELEASE message arrives.

Scheme 2: Explicit setup and estimated release, in which the SETUP message carries information about the duration of the burst so that, unlike scheme 1, no RELEASE message is needed to mark the end of the burst; this information is determined by the switch from the arrival time of the SETUP message and the information about the length of the burst contained in it.

Scheme 3: Estimated setup and explicit release, a scheme that is the opposite of scheme 2: instead of estimating the end of the burst, the start of it is estimated based on information con-

The JumpStart project has as its goal the definition of a signaling protocol and associated architecture for a WDM burst-switching network. The basic premise of the architecture is that data, aggregated in bursts, can be transferred from one end-point to another by setting up a lightpath just ahead of the data arrival.

Out-of-band signaling on a single channel	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*, that is, no electro-optical conversion takes place at intermediate nodes, and no assumptions are made about the data rate or signal modulation methods
Network intelligence at the edge	Most "intelligent" services are supported only at edge switches; core switches are kept simple
Signaling protocol implemented in hardware	To avoid becoming a bottleneck and to achieve wire-speed operation, the signaling protocol must be implemented in hardware
No global time synchronization	Following the "keeping it simple" principle, we do not assume time synchronization between nodes.

Table 1. *Basic architectural assumptions.*

tained in the SETUP message. This scheme, however, requires an explicit RELEASE message release the switching elements so they become available for routing other bursts.

Scheme 4: Estimated setup and estimated release, where both the start and end of the burst are predicted based on information contained in the SETUP message.

From the figure it is clear that these schemes differ in the amount of time the same burst would occupy the switching elements inside a switch. The tighter the estimate of the start and the end of the burst, the smaller the overhead of keeping the switching elements configured, and the lower the overall burst blocking probability in the network. Explicit notification schemes give the worst estimates (their own arrival time), while the predictive schemes are the best (assuming the estimates are accurate). However, there is a trade-off in the form of the number of signaling messages required to facilitate a particular scheme, as well as the complexity of the switch scheduler.

Consider a hypothetical idealized scheduler inside an OBS switch. The scheduler needs to keep state information for each burst traversing it in order to configure the switching elements to route the burst. With this hypothetical minimal scheduler, scheme 1 requires a single on/off bit for each switching element involved in routing a particular burst: *on* — the switching element is busy routing a burst; *off* — the switching element is free to route a new burst. A change in state takes place upon receipt of explicit SETUP and RELEASE messages. Scheme 2, which predicts the end of the burst, needs to associate a deadline with each switching element that indicates when the element will become available to route another burst. Schemes 3 and 4, which predict the start of the burst, require even more complex data structures within our hypothetical scheduler: under both schemes a finite horizon [4] needs to be kept for each switching element. This horizon is expressed as a vector of deadlines and must be consulted by the scheduler to determine at which times in the future a particular element will be available. The length of this horizon vector depends on the maximum possible delay between a SETUP message and the associated burst, and therefore is a function of the size of the network. As we can see, while predictive reservation schemes may have a potential positive effect on the overall blocking probability of the network, the switch hardware becomes significantly more complex [9].

Architectural Overview

The JumpStart project has as its goal the definition of a signaling protocol and associated architecture for a WDM burst switching network. Some of the basic architectural assumptions are summarized in Table 1. The basic premise of the architecture is that data, aggregated in bursts, can be transferred from one endpoint to another by setting up a lightpath just ahead of the data arrival. This is achieved by sending a signaling message ahead of the data to set up the path. Upon completion of data transfer, the connection either times out or is torn down explicitly.

Functional Requirements

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

- Asynchronous single bursts with a holding time shorter than the diameter of the network
- 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 of 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. In a burst-switched network, which essentially acts as a broker of time on a particular wavelength with high temporal resolution, this feature becomes relatively easy to implement due to the fact that signaling is out of band. Thus, the JumpStart architecture makes no assumptions about the types of traffic it carries and instead schedules time periods on wavelengths within the network. The particular format an end node uses to transmit its data to the destination 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 — For multicast connections the optical signal needs to be split at certain points along the paths of the routing tree in order for the network to remain all-optical. Such splitting presents a number of implementation and routing challenges; for instance, switches must be equipped with power splitting capabilities, while the number of signal splits is limited by the optical power budget. For our network architecture we presume that multicast-capable switches are not common in the network. Each end node is assigned one such switch as its multicast server through either administrative means or a separate signaling mechanism. These multicast servers also take care of setting up routing trees for multicast connections. Thus, all signaling messages from a source node that pertain to its multicast connections are routed by the network to its assigned multicast switch.

Within the JumpStart architecture we support both source-managed and leaf-initiated multicast models. An article currently under preparation addresses the native multicast support in greater detail.

Persistent Connections — 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. Defining a persistent route service allows the network to "pin" a route that all bursts follow. There are some network traffic engineering implications of persistent routes. If a significant fraction of network connections require such routes, dynamic load balancing through routing changes may become ineffective. To minimize this potential problem, network service providers may choose to treat persistent route connections as a premium service offering. The JumpStart architecture supports a persistent connection service to both unicast and multicast sessions.

Label Switching — Due to the flexible signaling message structure described in a later section, it is possible to incorporate into the JumpStart architecture elements of the generalized multiprotocol label switching (GMPLS) framework [2]. This can be achieved by adding properly formatted GMPLS labels to the signaling messages that establish new connections. The introduction of GMPLS into the architecture remains a topic of further study, however, since we also have plans for an alternative non-IP-driven control framework.

Signaling Flows and Messages

As described in the introduction, a number of signaling schemes are JIT in nature. In JumpStart we have adopted schemes 1 and 2, "explicit setup and explicit release" and "explicit setup and estimated release" (Fig. 1), since they require the simplest schedulers. The signaling protocols presented in this section support both schemes; it is up to the caller to decide which scheme will be used.

The signaling protocol functions are described in Table 2. While each connection in the OBS network has to go through a series of these phases, the signaling protocol is flexible in that a number of 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:

- Announce the session to the network (session declaration)
- Set up the path of the session (path setup)
- Announce the arrival of the burst (data transmission)

Combining the first three phases in Table 2 into one is useful to speed up the transmission of short bursts. However, this approach may not be appropriate for long-lived persistent connections. For these, the path setup phase must be separate from the data transmission phase, as discussed later.

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 2. *Signaling protocol functions.*

Both multicast and unicast connections can be mapped onto the phases in Table 2, although due to space limitations, in this section we will limit our discussion to unicast connections. Note that in order to facilitate robust network operations, additional protocols may need to be defined in the future (e.g., a routing protocol). While the message flows for new protocols will be different from those defined for connection setup, these protocols will use the same flexible message structure described later.

Underlying Assumptions — The design of the signaling protocol was guided by the following assumptions:

- Signaling is out of band.
- Signaling channel is best-effort link by link.
- Signaling messages are queued and processed by each intermediate node (queuing losses are possible).
- Signaling channel is presumed to possess a low bit error rate (e.g., 10^{-12} to 10^{-15})

The second assumption requires some explanation. Making the signaling protocol reliable link by link requires positive acknowledgments and the ability to retransmit lost messages. In a JIT environment where a burst travels with a short delay behind the signaling message, retransmitting a lost message may delay it enough to render it useless (i.e., the burst may arrive at the switch before the retransmitted signaling message that sets up its path). Taking into account the low bit error rate and the increased burden on the signaling engine for making the signaling protocol reliable, we have come to the conclusion that signaling protocol reliability is not desirable in a JIT OBS network.

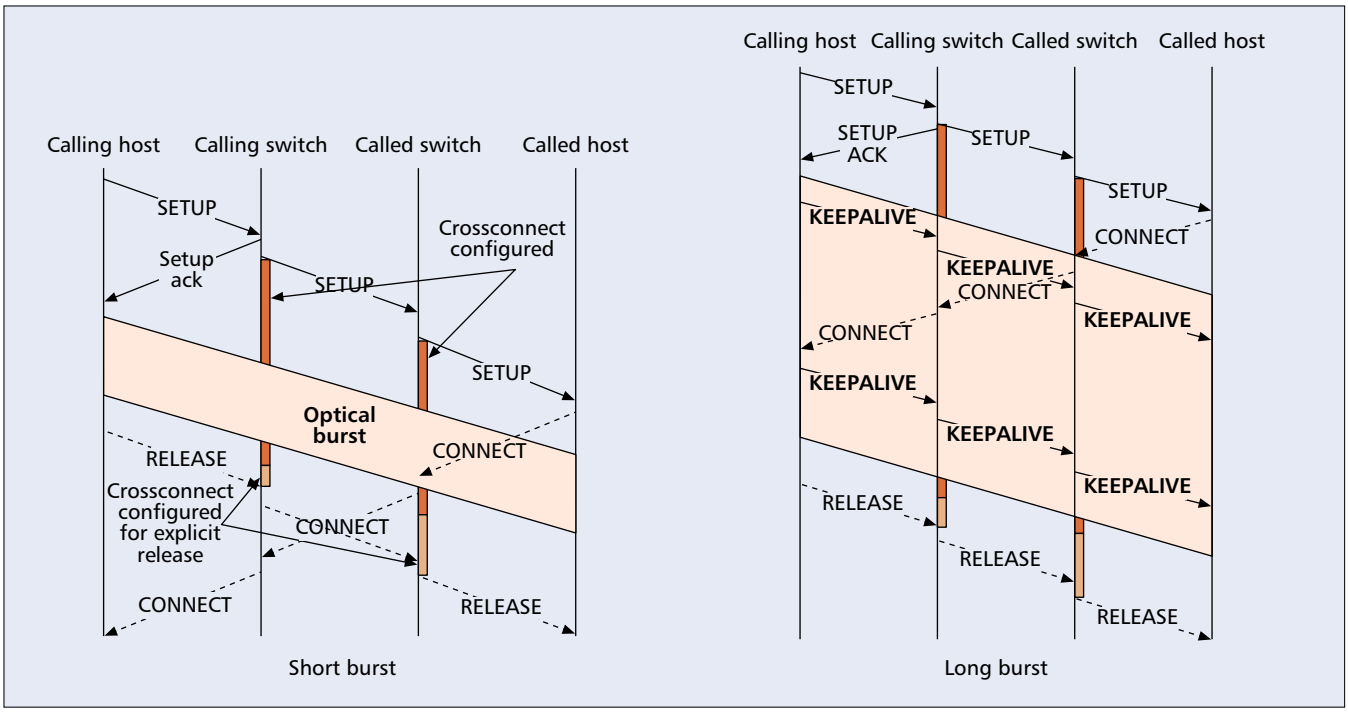

Figure 2. *Signaling flows for short bursts and lightpaths.*

Burst Delay — The transfer of data across the network is achieved by sending a signaling message ahead of the data burst in order to set up the path for the latter. As the signaling message traverses the network, it needs to stay ahead of the burst in order to give intermediate switches time to configure their switching elements. Note that the delay between the burst and the signaling message shrinks as they propagate across the network, since, unlike the burst that incurs no delay as it traverses the network, the signaling message experiences queuing and processing delay at every intermediate switch. Thus, the problem arises of estimating the initial burst delay ahead of time (i.e., before the signaling message is sent). This estimate is a function of the number of hops on the connection path. While the exact mechanism to perform and refine this estimation remains a topic for further investigation, it is important to understand that this estimation mechanism must be present in the network in order for it to function properly.

In our architecture, it is the switch through which the client node attaches to the network that provides this estimate. The estimate is returned to the client in response to an initial notification (SETUP) message by the client indicating its intention to send a burst. The client then uses this estimate to determine the time to start transmitting the burst.

Connection Classification — Connections in a JumpStart network can be classified along the following dimensions:

- Unicast 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., unicast short burst with timed release and on-the-fly path setup), except that multicast connections must always use persistent path setup.

Basic Unicast Message Types — The basic signaling messages used to set up a unicast connection in a JumpStart OBS network are listed and explained in Table 3.

The next two subsections present the flow of signaling messages for unicast connections for both on-the-fly and persistent path setup.

On-the-Fly Unicast Signaling Flow

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 the routes are determined independently for the two transmissions. For these connections, the session declaration, path setup, and data transmission phases are combined into a single message. The message flows are presented in Fig. 2. As seen in the figure, we distinguish two cases depending on the length of the burst; the flow of messages differs slightly depending on the duration of the transmission.

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 value for the incoming burst; this value is based on congestion information available at the switch and 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 message also includes the burst delay information and informs the source about the channel (wavelength) to use when sending the data burst.

Message name	Message function	Connection phases
SESSION DECLARATION	Notifies the network that a persistent-path unicast 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 intermediate ("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 3. *Basic message types.*

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 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 connection 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 actual use of preemption is a subject of further study.)

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 timeout 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.

Due to lack of 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, for example, blocking, preemption by a connection of higher priority, lack of route to host, and refusal by destination.

Persistent Path Unicast Signaling Flow

A persistent connection is established when it is necessary to guarantee that a series of bursts between the same source-destination pair will travel along the same path through the network. To this end, the session declaration and path setup phases are now separate from the data transmission phase (contrast this to on-the-fly connections in which all three phases are combined in one step). During the first two phases, a routing decision is made that is cached at all intermediate switches. The data transmission phase consists of a series of burst transmissions very similar to the on-the-fly connections discussed in the previous subsection. The main difference is that the SETUP message for each of the burst transmissions carries an identifier (similar to an MPLS label) used to access the cached routing decision.

Figure 3 demonstrates the flow of signaling messages. A SESSION DECLARATION message first travels from the source to the destination node and sets up a persistent path. This message is acknowledged by the destination with a DECLARATION ACK message. During the data transmission phase, a number of data bursts are transmitted; the source may also send KEEPALIVE messages to maintain the routing state at the switches. The source node releases the session by sending a SESSION RELEASE message to the network. A session may also be released by the network; in this case, a SESSION RELEASE message is sent to the source and destination nodes.

We would like to emphasize that the cross-connect elements at intermediate switches are not permanently configured for the path between the arrivals of the SESSION DECLARATION and SESSION RELEASE messages. Rather, the cross-connect configuration necessary to route a burst to the destination is simply cached when the SESSION DECLARATION message arrives. Then, each time a SETUP

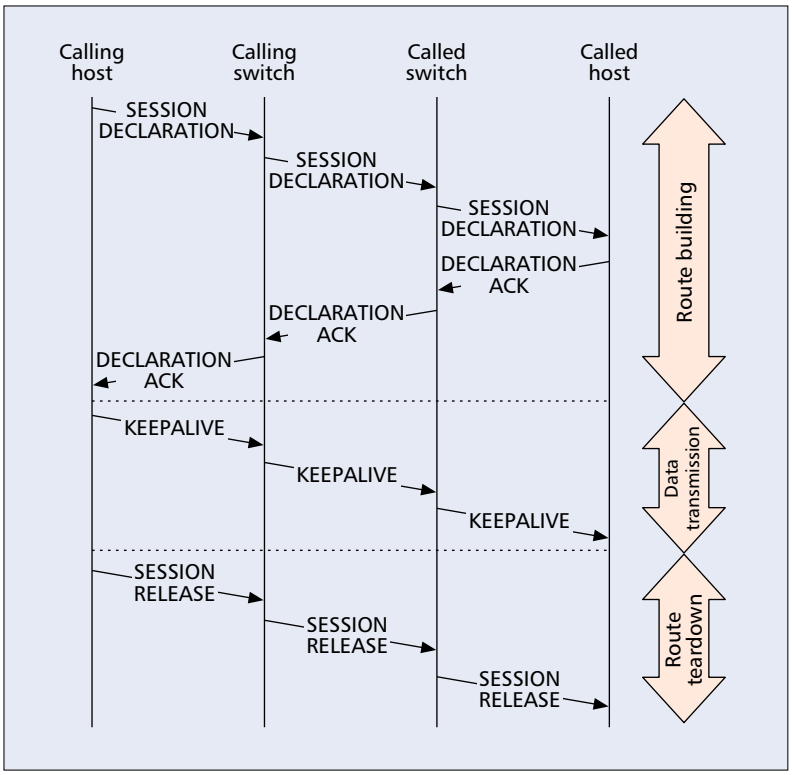

Figure 3. *Signaling flow for persistent path setup.*

message announces the arrival of a new burst for the connection, this 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).

Signaling Message Format

We have designed a new message structure for the JumpStart signaling protocols. The message format was designed to be easily implementable in hardware, and is flexible enough to accommodate future needs of signaling protocols. Figure 4 presents the structure of a signaling message. It consists of three parts: a common *header*, a number of *hardpath* information elements (IEs), and a number of *softpath* IEs (the concept of an information element is borrowed from ATM). Each IE carries information that pertains to a particular aspect of the signaling protocol in which it is being utilized. IEs are classified as hardpath or softpath depending on whether they are intended to be processed by hardware or software, respectively. The structure of both hardpath and softpath IEs is the same: a TLV [type, length, value] 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 hardpath and softpath subheaders provide information about the number of IEs present in each message. The hardpath subheader contains a bit mask such that a single bit set in the mask in a specific position corresponds to a specific IE type present in the subheader. This feature makes it easy for the hardware to parse the subheader and determine which IEs are present, as well as recognize invalid IE combinations. The number of hardpath IEs in the subheader is limited by the length of the bit mask (64 in our implementation). It also contains 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 deter-

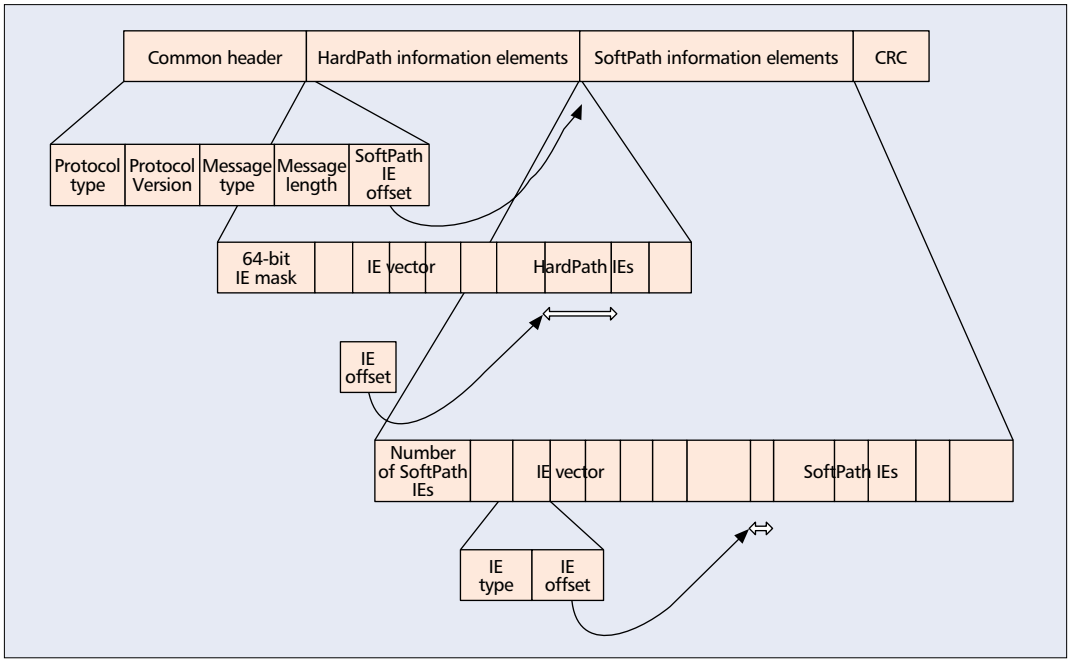

Figure 4. *The signaling message structure.*

mine the types of IEs present in the softpath header. However, unlike the hardpath subheader, the number of IEs that can be present in the softpath subheader is unlimited.

The common header has information about the specific signaling protocol (e.g., connection setup, routing) 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 is also appended with a CRC-32 sequence for integrity verification.

Conclusions and Future Work

In this article we present an architecture for an all-optical burst switching network using a just- intime signaling protocol. This architecture is intended to be used as a core network, and combines simplicity in control and signaling with a rich feature set, including native support for multicast and a variety of connection types (short bursts, lightpaths, persistent path connections with low jitter, etc.). We believe that such an architecture has the potential to take advantage of the vast amount of bandwidth and the special properties of the optical medium more fully than other approaches derived from electronic switching paradigms.

We have recently received additional funding to continue our work on the JumpStart architecture. Current activities include the design of prototype software and hardware that implement the JumpStart signaling protocol. Our ultimate goal is to deploy this software and hardware within the ATDNet (formerly MONET) environment in order to create an experimental OBS testbed to be made available to the research community.

Acknowledgments

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

References

[1] The Jumpstart project. http://jumpstart.anr.mcnc.org

[2] D. Mills *et al.*, "Highball: A High Speed, Reserved Access, Wide Area Network," Tech. rep. 90-90-1, Elec. Eng. Dept., Univ. Delaware, Sept. 1990.

[3] J. Y. Wei and R. I. McFarland, "Just-in-time Signaling for WDM Optical Burst Switching Networks," *J. Lightwave Tech.*, vol. 18, no. 12, Dec. 2000, pp. 2019–37.

[4] J. S. Turner, "Terabit Burst Switching," *J. High Speed Networks*, vol. 8, no. 1, Jan. 1999, pp. 3–16.

[5] M. Yoo and C. Qiao, "Just-enough-time (JET): A High Speed Protocol for Bursty Traffic in Optical Networks," *IEEE/LEOS Tech.. Global Info. Infra.*, Aug. 1997, pp. 26–27.

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

[7] M. Yoo, C. Qiao, and S. Dixit, "QoS Performance of Optical Burst Switching in IP-over-WDM Networks," *JSAC*, vol. 18, no. 10, Oct. 2000, pp. 2062–71.

[8] E. A. Varvarigos and V. Sharma, "Ready-to-go Virtual Circuit Protocol: A Loss-free Protocol for Multi- gigabit Networks Using FIFO Buffers," *IEEE/ACM Trans. Net.*, vol. 5, no. 5, Oct. 1997, pp. 705–18

[9] P. Mehrotra *et al.*, "Network Processor Design for Use in Optical Burst Switched Networks," *Proc. Int'l. ASIC/SOC Conf.*, Sept. 2001.

[10] P. Ashwood-Smith *et al.*, "Generalized MPLS — Signaling Functional Description," IETF draft, draft-ietf-mpls-generalized-signaling-06.txt, Apr. 200, work in progress.

Biographies

Ilia Baldine (ibaldin@anr.mcnc.org) received his B.S. degree in computer science from the Illinois Institute of Technology, Chicago, in 1993, and his M.S. and Ph.D. degrees in computer science from North Carolina State University in 1995 and 1998, respectively. He currently works as a network research engineer with the Advanced Networking Research group at MCNC, a non-profit research corporation located in Research Triangle Park, North Carolina. His research interests include all-optical networks, ATM networks, network protocols, and security.

George N. Rouskas [S '92, M '95, SM '01] (rouskas@cs.ncsu.edu) received his diploma in electrical engineering from the National Technical University of Athens, Greece, in 1989, and his M.S. and Ph.D. degrees in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, in 1991 and 1994, respectively. He joined the Department of Computer Science, North Carolina State University in August 1994, and has been an associate professor since July 1999. During the 2000–2001 academic year he spent a sabbatical term at Vitesse Semiconductor, Morrisville, North Carolina, and in May and June 2000 he was an invited professor at the University of Evry, France. His research interests include network architectures and protocols, optical networks, multicast communication, and performance evaluation. He is a recipient of a 1997 NSF Faculty Early Career Development (CAREER) Award, and coauthor of a paper that received the Best Paper Award at the 1998 SPIE Conference on All-Optical Networking. He also received the 1995 Outstanding New Teacher Award from the Department of Computer Science, North Carolina State University, and the 1994 Graduate Research Assistant Award from the College of Computing, Georgia Tech. He was a co-guest editor of *IEEE Journal on Selected Areas in Communications*, Special Issue on Protocols and Architectures for Next Generation Optical WDM Networks, October 2000, and is on the editorial boards of *IEEE/ACM Transactions on Networking*, *Computer Networks*, and *Optical Networks*. He is a member of the ACM and the Technical Chamber of Greece.

Harry G. Perros [M '87, SM '97] (hp@cs.ncsu.edu) received his B.Sc. degree in mathematics in 1970 from Athens University, Greece, his M.Sc. degree in operational research with computing from Leeds University, United Kingdom, in 1971, and his Ph.D. in operations research from Trinity College, Dublin, Ireland, in 1975. From 1976 to 1982 he was an assistant professor in the Department of Quantitative Methods, University of Illinois at Chicago. In 1979 he spent a sabbatical term at INRIA, Rocquencourt, France. In 1982 he joined the Department of Computer Science, North Carolina State University, as an associate professor, and since 1988 he is a professor. During academic year 1988–1989 he was on sabbatical, first at BNR, Research Triangle Park, North Carolina, and subsequently at the University of Paris 6, France. Also, during academic year 1995–1996 he was on sabbatical at Nortel, Research Triangle Park, North Carolina. He has published extensively in the area of performance modeling of computer and communication systems, and has organized several national and international conferences. He also published a monograph entitled *Queueing Networks with Blocking: Exact and Approximate Solutions* (Oxford Press). He is chair of IFIP W.G. 6.3 on the Performance of Communication Systems. His current research interests are in the areas of optical networks and their performance, and software performance evaluation.

Daniel Stevenson (stevenso@anr.mcnc.org) is the director of network research at MCNC. He has a management track record spanning successful project development, team development, directed research, development, and management of multi-institutional/cultural technical projects, technology transfer from research to commercial practice, and a successful startup business. He has numerous publications and conference presentations spanning gigabit networking, ATM security, and optical networking. He received a B.S. in physics in 1974 and an M.S. in physics in 1975. Prior to joining the staff of MCNC in 1990, he worked for Bell Telephone Labs, GTE Government Systems, Northern Telecom, and Bell Northern Research. From 1996 to 1999 he was chief technology officer and founder of Celotek Inc.

This architecture is intended to be used as a core network, and it combines simplicity in control and signaling with a rich feature set, including native support for multicast and a variety of connection types (short bursts, lightpaths, persistent path connections with low jitter, etc.).