An Alternative Implementation Perspective for the Scheduling Switch Architecture

George Theophilopoulos, *Member, IEEE*, Marios Kalyvas, Konstantinos Yiannopoulos, Kyriakos Vlachos, *Member, IEEE*, Emmanouel A. Varvarigos, and Hercules Avramopoulos, *Member, IEEE*

***Abstract*—In this paper, a novel configuration is proposed for the implementation of an almost all-optical switch architecture called the scheduling switch, which when combined with appropriate wait-for-reservation or tell-and-go connection and flow control protocols provides lossless communication for traffic that satisfies certain smoothness properties. An all-optical 2×2 exchange/bypass (E/B) switch based on the nonlinear operation of a semiconductor optical amplifier (SOA) is considered as the basic building block of the scheduling switch as opposed to active SOA-based space switches that use injection current to switch between ON and OFF states. The experimental demonstration of the optically addressable 2×2 E/B, which is summarized for 10–Gb/s data packets as well as synchronous digital hierarchy (SDH)/STM-64 data frames, ensures the feasibility of the proposed configuration at high speeds, with low switching energy and low losses during the scheduling process. In addition, it provides reduction of the number of required components for the construction of the scheduling switch, which is calculated to be 50% in the number of active elements and 33% in the fiber length.**

***Index Terms*—All-optical packet switching, all-optical signal processing, exchange/bypass (E/B) switch, lossless communication, scheduling switch architecture, semiconductor optical amplifier (SOA), ultrafast nonlinear interferometer.**

I. Introduction

In the quest toward high-capacity data networks, all-optical packet switching is set to provide a path for the deployment of more efficient transport networks, offering a variety of optical services in an affordable way [1]–[3]. For optical packet switching, the optical layer must be transformed from a static transmission medium to a dynamically reconfigurable facility. This should possess the ability of changing the connectivity between nodes during the time scale of a packet, possibly allowing for some limited bit-level processing [4]–[6]. To this end, semiconductor optical amplifier (SOA)-based switch modules have been demonstrated at data rates up to 100 Gb/s, operating with low switching energies ($<$ 100 fJ) and having the potential to be integrated in single chips [7]. Thus, the optical-to-electronic (O/E) and electronic-to-optical (E/O) conversion of the data signal at intermediate switches can be eliminated. However, O/E conversion is still required for header processing and consequently for the control of the switch fabric [8]. To efficiently perform packet switching in the so–called (almost) all-optical packet switches [9], where the data remains in the optical domain and the header is processed electronically, the architecture of the switch should provide lossless communication, efficient capacity utilization, packet arrival in the correct order, and design modularity. The scheduling switch architecture, which was first proposed in [9] and further studied in [10], meets the aforementioned requirements when combined with appropriate connection and flow control protocols. It is the purpose of this paper to show that the scheduling switch can be implemented with the experimentally demonstrated SOA-based 2×2 exchange/bypass (E/B) switch [11], offering the advantages of all-optical operation, high speed (given the potential of operation of the switch at 40 Gb/s [12]), low energy consumption, as well as major reduction in the number of active and passive optical components required for the construction of the switching fabric.

The remainder of the paper is organized as follows. In Section II, the principle of operation of the scheduling switch architecture is revised. A typical and the proposed configurations for the realization of the scheduling switch are described in Section III, while in Section IV the operation of the experimentally implemented 2×2 optically addressable E/B switch is summarized. Finally, in Section V, the cost for the construction of the scheduling switch is analyzed, component-wise, using both the initial and the proposed configuration.

II. The Scheduling Switch

The scheduling switch is designed to provide lossless communication for sessions that have certain smoothness properties or can be transformed to sessions with such properties, tolerating the corresponding delay. The time axis on a link is divided into frames of length equal to T slots, where we assume that all packets have the same length and require one slot for transmission. A session is said to have the (n, T)-burstiness property [9], [13] at a node if at most n packets of the session arrive at that node during a frame of size T. A session can easily be made to have the (n, T)-burstiness property at a source, and the property is automatically preserved throughout a network consisting of scheduling switches, since such switches maintain frame integrity.

We let n_{ij} be the number of packets that arrive during a frame over incoming link i and have to be transmitted on link j, and N be the number of incoming (and outgoing) links of a node.

Manuscript received October 7, 2003; revised July 1, 2004. This work was partially funded by EC within the frame of E-Photon/ONe project.

G. Theophilopoulos is with the Research Academic Computer Technology Institute (RACTI), Patras, Greece (e-mail: gtheof@cti.gr).

M. Kalyvas, K. Yiannopoulos, and H. Avramopoulos are with the Department of Electrical and Computer Engineering, National Technical University of Athens (NTUA), 15773 Athens, Greece.

K. Vlachos and M. Varvarigos are with the Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece, and also with the Research Academic Computer Technology Institute (RACTI), Patras, Greece.

Digital Object Identifier 10.1109/JLT.2004.838857

If the connection and flow control protocols used guarantee that the number of packets that require the same outgoing link j in a frame is less than or equal to the frame size T, i.e.,

$$\sum_{i=1}^{N} n_{i,j} \leq T \tag{1}$$

then all of the incoming packets can be assigned slots in the required outgoing links so that no packets will have to be dropped. In order to ensure that (1) holds for all transmitting sources, a flow control mechanism must be enforced at the edge, which may result in maximizing network traffic load (and thus link utilization) by the actual statistical multiplexing that is being performed. Both wait-for-reservation and tell-and-go protocols can be used to ensure that this property is met, as described in [14]–[16].

The scheduling switch consists of a scheduler with N input and N output ports, and an $N \times N$ nonblocking space switch, as shown in Fig. 1. The purpose of the scheduler is to rearrange the incoming packets so that packets appearing during the same slot at its output request different outgoing links of the space switch. If this procedure is done successfully, there will be no collisions at any of the space switch output ports.

The function of the scheduler can be described through a *frame arrival matrix* (N), defined as the $N \times N$ matrix, whose (i, j) component is equal to the number of packets that arrive during a given frame $F(i)$ of the incoming link i and require the same frame $F(j)$ of outgoing link j. By defining the *permutation matrix* as an $N \times N$ matrix with the property that each line has at most one nonzero element, indeed equal to "1", the *frame matrix* can be written as the sum of at most T *permutation matrices* P_S, $s = 1, 2, \ldots, T$, when condition (1) is satisfied. The matrix P_S can be used to determine the packet (if any) that will appear during slot s at each of the output ports of the scheduler. In particular, if the (i, j) element of this matrix is equal to "1", then a packet arriving over link i and departing over link j is assigned to the outgoing slot s of the scheduler.

The scheduler is comprised of N parallel branches, one for each input, where each branch delays the packets arriving on an incoming link until their assigned slots on their desired outgoing links. This is equivalent to a time-slot interchanger and is implemented using $(2\log_2 T - 1)$ three-state delay blocks (Fig. 1), where we have assumed T is a power of 2. The ith block (in Fig. 1, we illustrate the details for block $i = m$) consists of a three-state switch and three fiber delay paths, corresponding to delays equal to 0, 2^{i-1}, and 2^i time slots (we set one time slot to be equal to the duration of one packet). To ensure that the packets in the incoming frame can be assigned to any slot in the outgoing frame, the outgoing frame must start at least $(3T)/2 - 2$ after the incoming frame begins [10].

Using the approach of [17], each branch of delay blocks can be expanded into a corresponding graph, using a space–time representation, as shown in Fig. 2 for $T = 4$ and accordingly for $(2\log_2 4 - 1) = 3$ delay blocks. By using the previously described concept, the problem of scheduling packets through a branch of delay blocks to avoid collisions becomes a problem of routing every packet of the incoming frame through the space–time graph to the appropriate slot of the outgoing frame, where node-disjoint paths in the graph correspond to collision-free transmission through the delay blocks of the scheduler. In Fig. 2, the incoming frame corresponds to time slots 0–3, while the outgoing frame corresponds to time slots 4–7. Each line, either solid or dashed, represents a feasible state transition in the time–space domain; movement along a horizontal line between nodes corresponds to passing through a delay block without being delayed, while movement along a cross line between nodes corresponds to passing through a delay block and being delayed. Packet collisions can be avoided within a branch of delay blocks if the paths followed by the packets are node disjoint. Among the feasible state transitions in the space–time representation of Fig. 2, we observe a Beneš structure (solid lines), which is known to be rearrangeably nonblocking [18]. This means that it is possible to schedule the packets in a collision-free manner through a branch of delay blocks using only the transitions corresponding to the solid lines in the space–time graph. As mentioned previously, for

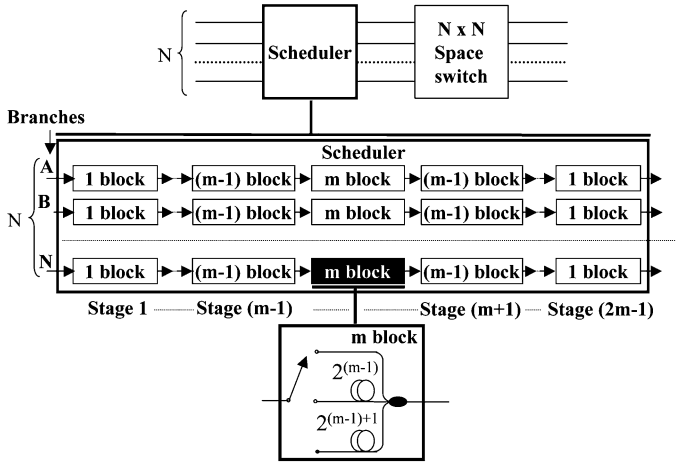


Fig. 1. Scheduling switch architecture, consisting of the scheduler (N inputs) and an $N \times N$ space switch. The scheduler is comprised of N branches, each of which consists of $2\log 2T - 1$ delay blocks. The ith delay block consists of one three-state switch and three delay lines of length $0, 2i-1$, and $2i$ time slots.

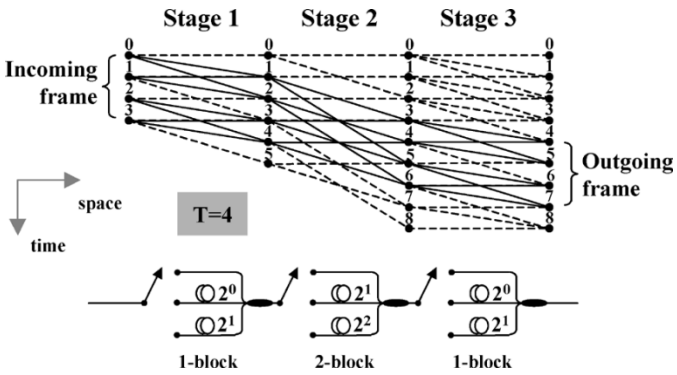


Fig. 2. Space–time representation of a branch of the scheduler when $T = 4$. Each line (solid or dashed) represents a feasible state transition in the time–space domain. The solid lines show that the time–space representation includes a Beneš subgraph.

the Beneš structure to stand, the outgoing frame must start $(3T)/2 - 2$ time slots after the incoming frame begins [10].

III. Typical and Alternative Configuration of the Scheduling Switch Architecture

We propose a novel configuration for the implementation of the aforementioned "scheduling switch" architecture that is based on the use of an experimentally demonstrated all-optical 2×2 E/B switch as the basic building block. The 2×2 switch, which is described in detail in Section VI, ensures the feasibility of the proposed configuration at high speeds, with low switching energy and low losses during the scheduling process. In addition, it provides 50% reduction in the required active elements (semiconductor optical amplifiers, or SOAs) components and 33% in the fiber length, if compared with a typical SOA-based implementation. From now on, we will refer to the proposed configuration as the "E/B implementation," while to the typical configuration as the "SOA-based implementation."

A. SOA-Based Implementation of the Scheduling Switch

One of the most common techniques applied for optical switching is the use of SOAs as active space switches [19]–[22]. Current injection into semiconductor pn-junctions generates free carriers, and this carrier modulation varies the loss and/or gain characteristics. Employing these characteristics, switchable SOAs can be realized, and many different configurations for different applications have already been demonstrated [23]. The modulation speed that has been achieved with this method is in the order of 1 ns.

A typical implementation of the ith block of the scheduler using the aforementioned technique is shown in Fig. 3. Each of the three SOAs can be operated in "transparent" (ON) or in "block" (OFF) mode, allowing or preventing, respectively, the corresponding signal to pass through. In this way, the incoming packet may experience zero delay, delay equal to 2^{i-1} time slots, or delay equal to $2^{\subset}$ time slots, depending on which of the three SOAs is ON. In every time slot, only one of the three SOAs is ON during the slot, while the other two are OFF, depending on the delay that is required to be inserted.

Following a similar rationale, the $N \times N$ space switch can be implemented using N^2 SOAs, as shown in Fig. 4. A fully connected shuffle network between input and output ports is realized using adequate power splitters and combiners. Each connecting path can be switched ON or OFF using the corresponding SOA. In particular, only one of these SOAs shall be ON during a slot, depending on the desirable outgoing link, while the rest $(N-1)$ shall be OFF. Such a switching matrix generates splitting losses that may be important for large switching arrays. To compensate for the losses, additional booster amplifiers can be included with the drawback of noise accumulation. The configuration shown in Fig. 4 forms a strictly nonblocking space switch with broadcast capabilities.

B. E/B Implementation of the Scheduling Switch

In this subsection, we describe the proposed E/B implementation for the scheduler and the $N \times N$ space switch that follows it,

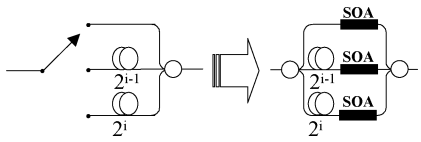


Fig. 3. Implementation of the ith delay block of the scheduler, using three switchable SOAs.

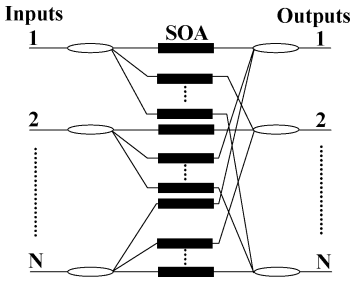


Fig. 4. Common implementation of the $N \times N$ space switch using N^2 SOAs.

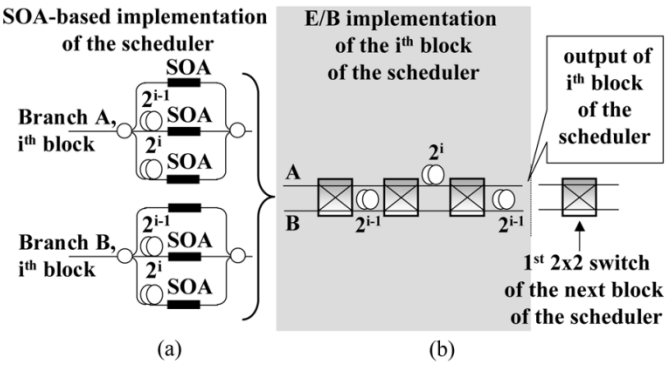


Fig. 5. (a) SOA-based implementation and (b) E/B implementation of the ith block of the scheduler.

using appropriate number of 2×2 E/B switches. A 2×2 switch has two operating states: the BAR state, where the two input signals pass through unaffected to the corresponding output ports, and CROSS, where the two input signals are interchanged at the output ports (the principle of operation of the 2×2 E/B switch is explained in more detail in Section IV and in [11]).

Fig. 5(a) shows the SOA-based implementation of the ith blocks for two branches of the scheduler, while in Fig. 5(b) the corresponding E/B implementation is depicted. We will show that the two implementations are functionally equivalent. In the E/B implementation, every two branches of the scheduler (corresponding to a pair of input ports) are integrated in the two ports of a 2×2 switch, and the whole block uses three such switches. The fourth 2×2 switch shown on the right of Fig. 5(b) is the corresponding first switch of the next block. We can change at will the state of each 2×2 switch at every time slot. In the following paragraphs, we prove that the space–time representation of the configuration of Fig. 5(a) can be emulated in a collision-free manner by the space–time representation of the configuration of Fig. 5(b). This means that we can replace the configuration of Fig. 5(a) with the configuration of Fig. 5(b), without affecting the essential functionality and properties of the scheduling switch, except for some additional fixed delay

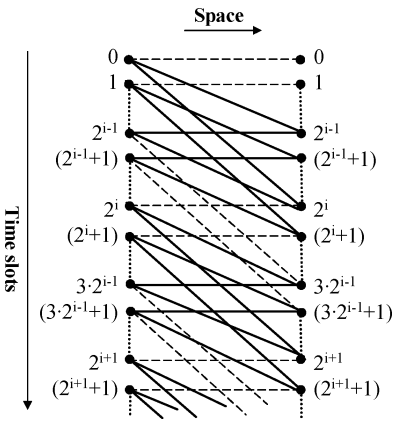


Fig. 6. Space-time representation of the ith block of the scheduler (or equivalent, of the ith block of the Beneš structure). We can see that possible collisions may appear at the output of the ith block only between some of the input packets that are 2^{i-1} time slots apart.

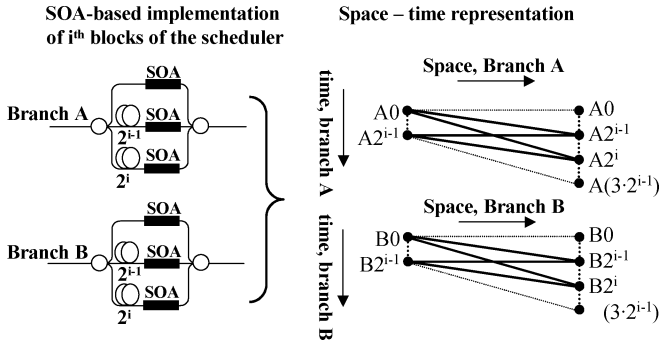


Fig. 7. Time–space representation of the ith block of two branches A and B of the SOA-based implementation of the scheduler. We only illustrate time–space locations of packets (nodes) that are 2^{i-1} slots apart from each other, because they are the only ones that could generate collisions at the outputs of block i. The solid lines correspond to delay-block states that may be used.

that is introduced. The new implementation has several performance and cost advantages over the previous implementation, as will be shown subsequently.

In the SOA-based implementation, collisions may occur at each of the branches of the scheduler and specifically at the output of block i, only between packets that arrive at its input being 2^{i-1} time slots apart from each other. This can become clear by generalizing Fig. 2 for N stages. In this case, the time–space representation of the ith block of the scheduler (or equivalently, of the ith block of a Beneš structure) is shown in Fig. 6 (solid lines). The solid lines are used as possible space–time paths by the scheduling algorithm, while the dashed lines are not. We can see that possible collisions at the output of the ith block appear only between some of the input time slots (packets) that are 2^{i-1} apart (e.g., input slots 0 and 2^{i-1}, 1 and $2^{i-1}+1$, 2 and $2^{i-1}+2 \ldots 2^{i-1}-1$ and 2^i-1, 2^i and $3\cdot 2^{i-1}$, 2^i+1 and $3\cdot 2^{i-1}+1$ etc). Actually, this is a property of the ith block of a Beneš structure.

We consider two branches (A and B) of the scheduler and two packets (that may collide) at each branch, assuming without loss of generality that they appear in time slots "0" and "2^{i-1}." In Fig. 7, we show the time–space location of four such packets at the inputs of the ith block (two packets for branch A and two packets for branch B) of the scheduler (SOA-based implementation), and the possible time–space locations of the four packets at the outputs of block i. Packets that arrive at the input of the ith block but are not 2^{i-1} time slots apart from each other have no possibility of colliding at the outputs of the ith block and are not depicted

In Fig. 8, we give the space–time representation of the ith block of the E/B implementation. "Node" An (or Bn) of a given substage of the time–space representation of Fig. 8 corresponds to the nth time slot of the upper (or lower, respectively) input of the 2×2 switch of that substage. For example, $\mathrm{A}2^{i-1}$ of substage 2 stands for time slot 2^{i-1} at the upper input (branch A) of the second 2×2 switch. In Fig. 8, we only depict the time–space paths (either by solid or by dashed lines) that can be followed by the four packets arriving at the inputs of block i of the scheduler on slots "0" and "2^{i-1}" for branches A and B. Packets that arrive on slots that are not 2^{i-1} time slots apart from each other, do not collide (since all delays introduced in block i are multiples of 2^{i-1} slots) and are not depicted in detail. In addition, packets on other branches are not related to the packets on branches A and B since they follow space-disjoint paths (they use different 2×2 switches). Note that the incoming packets at time–space slots A0, $\mathrm{A}2^{i-1}$ are routed to time–space slots $\mathrm{A}2^{i-1}$, $\mathrm{A}2^i$ at the output of block i of the SOA-based implementation (Fig. 7), while they are routed to time–space slots $\mathrm{A}2^i$, $\mathrm{A}3\cdot 2^i$ at the output of block i of the E/B implementation (Fig. 8). This additional delay of 2^{i-1} slots, introduced by block i in the E/B implementation, results in the imposition of an extra total delay to the outgoing frame, in relation to the incoming frame. This fixed additional delay does not play any role in the switch ability to schedule packets in a collision-free way and is calculated later on in this section of the paper (2). The additional propagation time per hop can be viewed as a transmission constraint and affect delay sensitive traffic. However, this delay can be considered as the necessary buffering time as in conventional electronic Internet protocol (IP) routers. Therefore, the same traffic delay constraints apply also in the case of an optical network consisting of scheduling switches as core nodes.

In order to make the principle of operation of the E/B implementation and the role of the 2×2 switches more clear, we mention the following example, referring to Fig. 8; the first packet in branch A (which is initially placed in slot "0") can a) experience zero delay (with the 2×2 E/B switch of substage 1 operating in the BAR state) and remain in slot "0" (in particular in "A0" since it is placed in the upper line—branch A—of the delay block) or b) experience delay equal to 2^{i-1} time slots (if the exchange/bypass switch of substage 1 operates in CROSS state) and be carried to slot "2^{i-1}" (in particular in "$\mathrm{B}2^{i-1}$," since it is now placed in the lower line—branch B—of the delay block). At the same time, the first packet in branch B (initially placed in slot "0") will in case a) experience delay equal to 2^{i-1} time slots and be carried to slot "$\mathrm{B}2^{i-1}$" or, in case b), experience zero delay and remain in slot "0" but in the upper branch (slot "A0").

In Fig. 8, we have limited the transitions between nodes to a subset of transitions: the solid lines correspond to delay-block states that may be used, while the dashed lines correspond to delay-block states that will not be used. The resulting structure of solid lines in the space–time representation does not follow a known structure (e.g., in Fig. 2, a Beneš structure was formed),

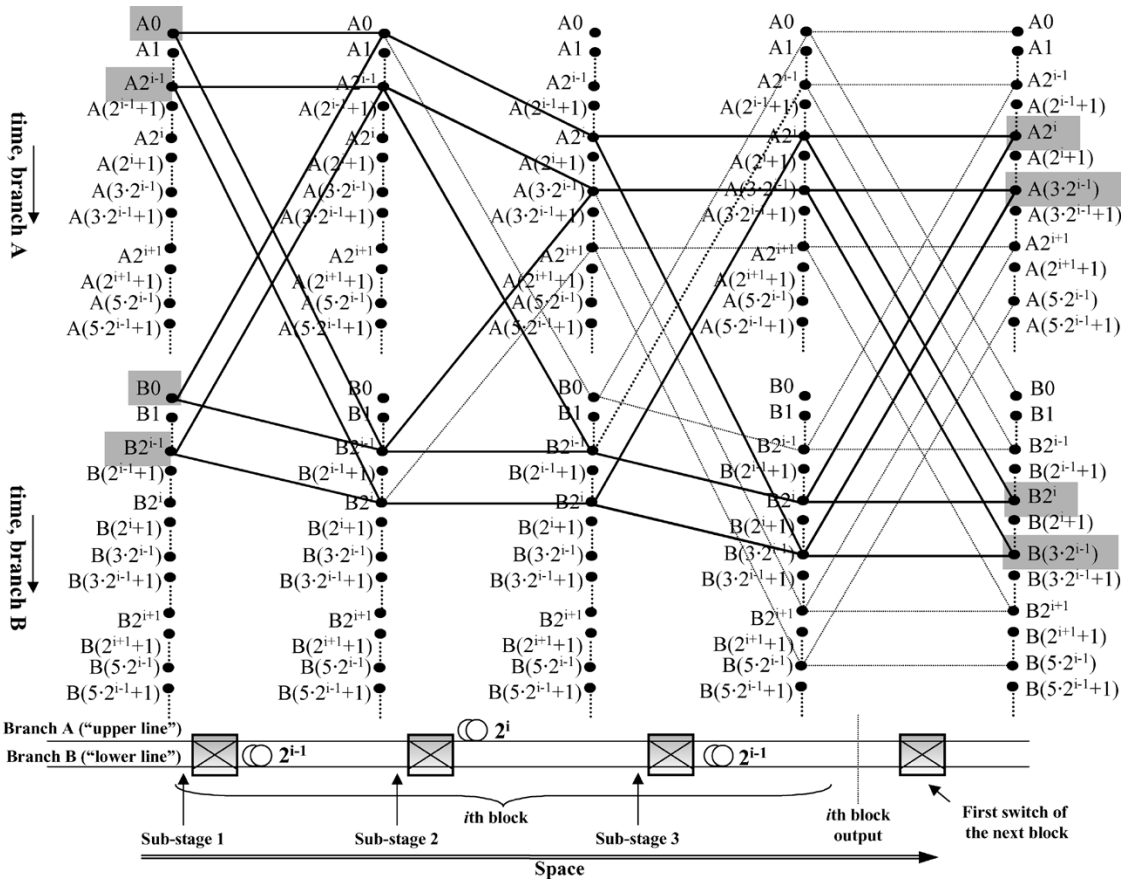


Fig. 8. Space–time representation for the ith block of the E/B implementation of the scheduler architecture for packets that are 2^{i-1} slots apart from each other. The lines (dashed and solid) represent all the feasible state transitions in the time–space domain. The solid lines correspond to delay-block states that may be used.

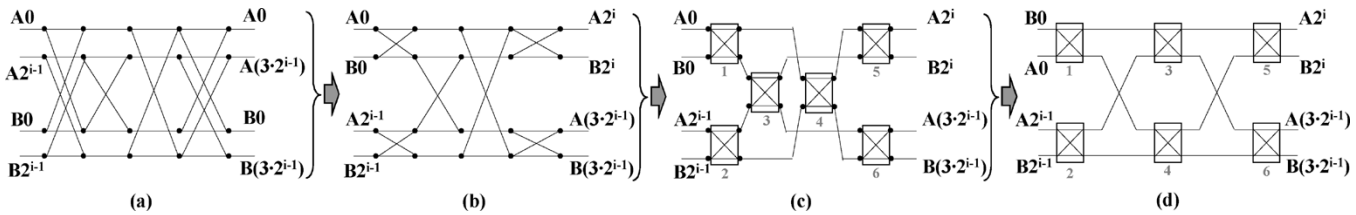


Fig. 9. (a) Network formed by the solid lines of Fig. 8. (b) Equivalent network, formed by rearranging these lines. (c) Equivalent network, formed by substituting the "crossed" lines with elementary 2×2 switches. (d) Same topology as (c). A Beneš switch is formed, i.e., in every case, node-disjoint paths exist for all of the four packets.

and it is not thereby profound that packets can be routed to the desired time slots in a collision-free manner, equivalent to that shown in Fig. 7.

However, we will show that the solid-line network of Fig. 8 is a Beneš structure. Fig. 9(a) depicts the solid lines of Fig. 8, excluding the time-domain representation on the vertical axis (time slots A0 and $\mathrm{A}2^i$ are depicted at the same level—only the space domain is depicted). If we rearrange the solid lines, they form the space representation graph shown in Fig. 9(b). Moreover, by substituting the parallel/crossed lines with 2×2 elementary switches, we form the network of Fig. 9(c). By placing the 2×2 switches properly in space, the Beneš graph of Fig. 9(d) is formed. Since the Beneš graph is rearrangeably nonblocking, every input can be routed to any output using a node-disjoint path. This means that the input packets at branches A and B may come out at any desired slot in a collision-free manner by using the proposed architecture.

By expanding this conclusion, we can claim that when we can find collision-free paths to reschedule the packets in any way allowed by the original architecture in Fig. 5(a), then we can always find collision-free paths to reschedule the packets in the same way using the proposed architecture, shown in Fig. 5(b). The only difference is that packets that enter in slots 0 and 2^{i-1}, instead of appearing in slots 2^{i-1} and 2^i, appear in slots 2^i and $3 \cdot 2^{i-1}$. This means that by using the new architecture, an extra time delay is inserted. In order to calculate the minimum number of time slots that the incoming and the outgoing frame must be apart when using the E/B implementation, we calculate the number of time slots between the first packet of the incoming frame (slot "0") and the first packet of the outgoing frame. This number corresponds to the delay

of 2^i time slots through each one of the i blocks, where $i = 1, 2, \ldots, \log_2 T - 1, \log_2 T, \log_2 T - 1, \ldots, 2, 1$. By summing all these delays, we get

$$L_{\text{branch}} = 2 \cdot \sum_{i=1}^{\log_2 T - 1} 2^i + T = 2^{\log_2 T + 1} - 4 + T = 3T - 4. \quad (2)$$

This means that the outgoing frame must start at least $3T - 4$ time slots after the incoming frame begins. Consequently, we need double the time in order to rearrange the packets in the proposed scheme (in the original architecture, this number was $[3T]/2 - 2$).

It is easy to show that all the other packets of the two frames (entering in branches A and B) can be rearranged in a collision-free manner as well, since in each substage of Fig. 8, two standard time slots are used for the two packets of each frame that are 2^{i-1} apart from each other. This observation ensures that no internal collisions occur inside the ith block of the scheduler.

With regard to Fig. 8, one other note must be made as to the role of the fourth E/B switch (the first 2×2 switch of the next block of the scheduler). At the input of this switch, all the packets have been assigned to the desired time slot, but it is possible to be in the wrong line of the delay block (branch A or B). For example, a packet from frame A may be in the lower line but assigned to the right time slot. The line that a packet is (upper or lower) does not have any meaning, since it can be corrected from the E/B switch of the next block. Yet, at the output of the scheduler, an extra E/B switch is obligatory in order to assign all the packets from frame A to frame A and all the packets from frame B to frame B. This can be done in such a way that frame A occurs in the upper line at the input of the space switch and frame B at the lower, or vice versa, accomplishing elementary switching this way. The last observation has an impact on the design of the $N \times N$ space switch as well, as will be shown subsequently.

Consequently, each stage of the SOA-based implementation (two branches) can be emulated by the E/B implementation, at the cost of a fixed extra delay that is inserted between the incoming and the outgoing frame. If all the stages of the SOA-based implementation are replaced by the E/B implementation, then the whole scheduler is emulated, except that packets in the input frame are routed to a frame that starts $(2T - 4)$ time slots after the end of the input frame, instead of $(T/2 - 2)$ when using the original configuration.

By using the 2×2 E/B switch, it is possible to simplify the space switch as well and implement it not with a SOA-based crossbar configuration (see Fig. 4) but with a different one that must be at least rearrangeably nonblocking (e.g., Beneš). Fig. 10 shows a 4×4 rearrangeably nonblocking Beneš structure for the space switch. The corresponding crossbar switch according to the SOA-based implementation would use $N^2 = 16$ active elements, while this one uses only six. It is known that for any N, a rearrangeably nonblocking $N \times N$ Beneš structure consists of three stages and requires $(N/2)$ 2×2 switches at its input, $(N/2)$ 2×2 switches at its output and $2(N/2) \times (N/2)$ Beneš switches at its middle stage. The 2×2 E/B switches at the output of the scheduler branches (i.e., before the input ports of the space switch) can be used as the first stage of the Beneš configuration, thus reducing the required switches in the space switch.

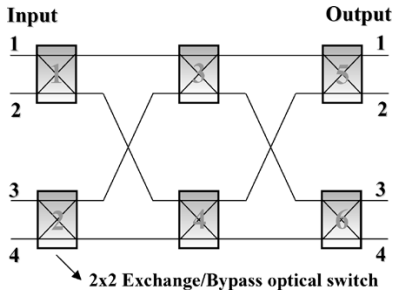


Fig. 10. 4×4 rearrangeably nonblocking Beneš structure for the space switch, using six 2×2 E/B switches.

IV. Experimental Demonstration of the 2×2 All-Optical E/B Switch

For the operation of the optically addressable 2×2 E/B switch, three optical signals are needed (two data signals and one control signal), as shown in Fig. 11. Data signals enter the switch from the input ports 1 and 2. If there is no control signal, the switch is in the BAR state, and both data signals pass straight through to the output ports 1 and 2. If the control signal is present, the switch is in the CROSS state, and the two data streams are interchanged at its output. The length of the bit sequence that is interchanged through the switch is determined by the length of the control signal and may be arbitrarily long or short, depending on the length of the incoming packet.

The 2×2 E/B switch that was experimentally demonstrated [11] is constructed with an ultrafast nonlinear interferometer (UNI)-based optical gate [24] properly configured to provide two input, two output, and one control port.

Initially, the performance of the switch was investigated at the data packet level at 10 Gb/s and corresponding traces, recorded on a sampling oscilloscope, verified successful operation. In addition, the error performance of the switch was evaluated in static configuration with input data of synchronous digital hierarchy (SDH)/STM 64 format and was calculated to be less than 10^{-11} for both data signals in both switch states.

It is important to note that if this E/B unit is used for optical packet switching, it relaxes the requirement for guard bands between the packets, since the switch changes state within the bit period. In avoiding guard bands, the improvement in throughput becomes more pronounced as the packet length decreases.

V. Cost Analysis

A basic cost analysis, concerning the SOA-based and the E/B implementation of the scheduler and the space switch, is presented in this section. As measures of cost, we will compare the total number of elementary components and the insertion losses of the original and of the proposed architecture.

The SOA-based scheduling switch architecture is composed of a scheduler and an $N \times N$ space switch. The scheduler has N parallel branches, one for each input port of the space switch. Consequently, for the whole scheduler

$$K_{\text{SOAs}}^{\text{Sched.}} = 3N(2\log_2 T - 1) \quad (3)$$

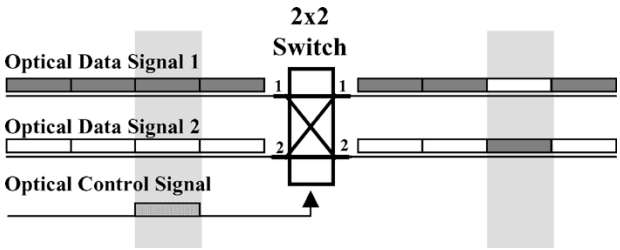


Fig. 11. Principle of operation of the 2 × 2 E/B switch.

active elements (SOAs) are required, as well as $[2{\cdot}N{\cdot}(2\log_2 T-1)]$ 3:1 couplers. In addition, for each delay block (the ith), we need a length of fiber that inserts $(2^i + 2^{i+1})$ time slots delay. Accordingly, for every branch of the scheduler, we need

$$L_{\text{branch}} = 2 \cdot \sum_{i=0}^{\log_2 T-2} (2^i + 2^{i+1}) + \left(\frac{T}{2} + T\right) \tag{4}$$

total length of fiber, measured in time slots. The summation term in (4) corresponds to all the symmetrical blocks of the scheduler, while the last term to the middle delay block (the $[\log_2 T-1]$th). For all the branches, the required fiber is calculated to be

$$\begin{aligned} L = N \cdot L_{\text{branch}} &= 2 \cdot N \cdot \sum_{i=0}^{\log_2 T-2} (2^i + 2^{i+1}) + N \cdot \left(\frac{T}{2} + T\right) \\ &= 6 \cdot N \cdot \sum_{i=0}^{\log_2 T-2} (2^i) + 3N \cdot \frac{T}{2} = 9 \cdot N \cdot \frac{T}{2} - 6N \end{aligned} \tag{5}$$

time slots.

Concerning the $N \times N$ space switch in the SOA-based implementation (see Fig. 4), the number of active elements (SOAs) required is

$$K_{\text{SOAs}}^{N\times N} = N^2. \tag{6}$$

In the E/B implementation, the number of active elements in the scheduler is calculated as following: the scheduler has $(N/2)$ two-port branches, each consisting of $(2\log_2 T - 1)$ blocks, while each block has three E/B switches. Consequently, $[3 \cdot (N/2) \cdot (2\log_2 T - 1)] = [3N \cdot (\log_2 T - 1/2)]$ active elements (SOAs) are required for the scheduler. At the output of the scheduler (and at the input of the space switch), another $(N/2)$ E/B switches are needed, and these will be subtracted from the active elements that the space switch needs. Consequently, the number of the active elements for the scheduler is

$$K_{\text{SOAs}}^{\text{Sched.}} = 3N\left(\log_2 T - \frac{1}{2}\right) + \frac{N}{2} = 3N\left(\log_2 T - \frac{1}{3}\right). \tag{7}$$

The total length of fiber used in the delay stages of this case is calculated by following the same rationale as in the SOA-based implementation and is given by

$$L = \left(\frac{N}{2}\right) \cdot \left[2 \cdot \sum_{i=0}^{\log_2 T-2} (2 \cdot 2^i + 2^{i+1}) + \left(\frac{T}{2} + T + \frac{T}{2}\right)\right]$$

$$= N \cdot \sum_{i=0}^{\log_2 T-2} (2^{i+2}) + N \cdot T = 4 \cdot N \cdot \left(\frac{T}{2} - 1\right) + N \cdot T$$

$$= 3 \cdot N \cdot T - 4 \cdot N. \tag{8}$$

TABLE I
COST ANALYSIS FOR THE SOA-BASED AND THE E/B IMPLEMENTATION OF THE SCHEDULING SWITCH ARCHITECTURE IN TERMS OF ELEMENTARY COMPONENTS COST

	Components	SOA-based implementation	E/B implementation
N-branch Scheduler	Delay blocks	$2{\cdot}N\log_2 T - N$	$3{\cdot}N\log_2 T - 3{\cdot}N/2$
	Fiber length	$9{\cdot}N{\cdot}T/2 - 6{\cdot}N$	$3{\cdot}N{\cdot}T - 4{\cdot}N$
	Active Elements (SOAs)	$3{\cdot}N{\cdot}(2\log_2 T - 1)$	$3{\cdot}N{\cdot}(\log_2 T - 1/3)$
	3:1 couplers	$2{\cdot}N{\cdot}(2\log_2 T - 1)$	–
NxN Space Switch	Active Elements (SOAs)	N^2	$N{\cdot}(\log_2 N - 1)$

Concerning the $N \times N$ space switch, it is known that, in general, a rearrangeably nonblocking $N \times N$ Beneš structure requires

$$S_N = N \cdot \log_2 N - \frac{N}{2} \tag{9}$$

2×2 switches, with N being a power of 2. In our case, this number is further reduced by $N/2$ because of the respective extra number of E/B switches that the scheduler uses at its output. Consequently, the number that the $N \times N$ space switch uses in the rearrangeably nonblocking Beneš structure is

$$K_{\text{SOAs}}^{N\times N} = N \cdot (\log_2 N - 1). \tag{10}$$

Table I summarizes the above cost analysis for the original and the alternative switch architecture in terms of elementary components cost.

Concerning the scheduler, the proposed technique consists of more delay blocks, since two blocks of the original scheduler equals three serially connected E/B switches. However, the total length of fiber used in the alternative technique is 33% less, while the number of the active elements (SOAs) as well as the corresponding electronic circuitry is reduced by almost 50%. In addition, no 3:1 couplers are required, thus limiting the circuit losses. As far as the space switch concerns, the active elements used in the proposed rearrangeably nonblocking Beneš architecture are reduced by $(\log_2 N - 1)/N$. Since the 2×2 E/B switch exhibits 4-dB gain, these switches perform very well when cascaded, dispensing the need of amplification between them. This does not occur in the SOA-based configuration, since the use of two 3:1 splitters in each stage of the scheduler leads to a minimum of 9,5 dB losses per stage. Finally, the switching energy for the demonstrated 2×2 switch is in the order of femto-joule (fJ)/pulse, orders of magnitude lower than the power that the switchable SOAs demand.

VI. Conclusion

This paper proposed a novel configuration for the implementation of an almost all-optical switch architecture, called the "scheduling switch" architecture by using an experimentally demonstrated all-optical 2×2 E/B switch as the basic building block, as opposed to active SOA-based space switches that use injection current to switch between ON and OFF states. The 2×2 switch ensures the feasibility of the proposed configuration at high speeds, with low switching energy and low losses during the scheduling process. In addition, it provides 50% reduction in the required active elements (SOAs) components and 33% in the fiber length.

References

[1] M. Renaud, F. Masetti, C. Guillemot, and B. Bostica, "Network and system concepts for optical packet switching," *IEEE Commun. Mag.*, vol. 35, no. 4, pp. 96–102, Apr. 2001.

[2] M. J. O'Mahony, D. Simeonidou, D. K. Hunter, and A. Tzanakaki, "The application of optical packet switching in future communication networks," *IEEE Commun. Mag.*, vol. 39, no. 3, pp. 128–135, Mar. 2001.

[3] D. Benjamin, R. Trudel, S. Shew, and E. Kus, "Optical services over the intelligent optical network," *IEEE Commun. Mag.*, vol. 39, no. 9, pp. 73–78, Sep. 2001.

[4] V. W. S. Chan, K. L. Hall, E. Modiano, and K. A. Rauschenbach, "Architectures and technologies for high-speed optical data networks," *IEEE J. Lightw. Technol.*, vol. 16, no. 12, pp. 2146–2168, Dec. 1998.

[5] K. E. Stubkjaer, "Semiconductor optical amplifier-based all-optical gates for high-speed optical processing," *IEEE J. Sel. Topics Quantum Electron.*, vol. 6, no. 6, pp. 1428–1435, Nov.–Dec. 2000.

[6] H. Avramopoulos, "TDM devices and their applications," in *Proc. Optical Fiber Communication Conf. (OFC 2001)*, pp. WE1–1.

[7] S. A. Hamilton, B. S. Robinson, T. E. Murphy, S. J. Savage, and E. P. Ippen, "100 Gb/s optical time-division multiplexed networks," *IEEE J. Lightw. Technol.*, vol. 20, no. 12, pp. 2086–2100, Dec. 2002.

[8] R. L. Cruz and J.-T. Tsai, "COD: Alternative architectures for high speed packet switching," *IEEE/ACM Trans. Netw.*, vol. 4, no. 1, pp. 11–21, Feb. 1996.

[9] E. A. Varvarigos, "The 'Packing' and the 'Scheduling' packet switch architectures for almost all-optical lossless networks," *J. Lightw. Technol.*, vol. 16, no. 10, pp. 1757–67, Oct. 1998.

[10] J. P. Lang, E. A. Varvarigos, and D. J. Blumenthal, "The λ-scheduler: A multiwavelength scheduling switch," *J. Lightw. Technol.*, vol. 18, no. 8, pp. 1049–1063, Aug. 2000.

[11] G. Theophilopoulos *et al.*, "Optically addressable 2×2 exchange bypass packet switch," *IEEE Photon. Technol. Lett.*, vol. 14, no. 7, pp. 998–1000, Jul. 2002.

[12] N. S. Patel, K. A. Rauschenbach, and K. L. Hall, "40-Gb/s demultiplexing using an ultrafast nonlinear interferometer (UNI)," *IEEE Photon. Technol. Lett.*, vol. 8, no. 12, pp. 1695–1697, Dec. 1996.

[13] S. J. Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Trans. Commun.*, vol. 39, no. 12, pp. 1802–12, Dec. 1991.

[14] E. A. Varvarigos and V. Sharma, "An efficient reservation connection control protocol for gigabit networks," *Comput. Netw.*, vol. 1998, no. 12, pp. 1135–1156, 1998.

[15] E. A. Varvarigos, "Control protocols for multigigabit-per-second networks," *IEICE Trans. Commun.*, vol. 1998, no. 2, pp. 440–448, 1998.

[16] E. A. Varvarigos and J. P. Lang, "A virtual circuit deflection protocol," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 335–349, Jun. 1999.

[17] D. Hunter and D. Smith, "An architecture for frame integrity optical TDM switching," *J. Lightw. Technol.*, vol. 11, no. 5, pp. 914–924, May–Jun. 1993.

[18] F. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. San Mateo, CA: Morgan Kaufmann, 1992.

[19] C. Guillemot *et al.*, "Transparent optical packet switching: The European acts KEOPS project approach," *J. Lightw. Technol.*, vol. 16, no. 12, pp. 2117–2134, Dec. 1998.

[20] D. Chiaroni *et al.*, "First demonstration of an asynchronous optical packet switching matrix prototype for multiterabit class routers/switches," in *Proc. Eur. Conf. Optical Communication (ECOC 2001)*, vol. 6, pp. 60–61.

[21] S. L. Danielsen, B. Mikkelsen, C. Joergensen, T. Durhuus, and K. E. Stubkjaer, "WDM packet switch architectures and analysis of the influence of tunable wavelength converters on the performance," *J. Lightw. Technol.*, vol. 15, no. 2, pp. 219–227, Feb. 1997.

[22] M. Renaud, M. Bachmann, and M. Erman, "Semiconductor optical space switches," *IEEE J. Sel. Topics Quantum Electron.*, vol. 2, no. 2, pp. 277–287, Jun. 1996.

[23] P. Doussière, "Recent advances in conventional and gain clamped semiconductor optical amplifiers," presented at the Conf. Optical Amplifiers Their Applications (OAA), Monterey, CA, Jul. 11–13, 1996.

[24] K. Tajima, S. Nakamura, and Y. Sugimoto, "Ultrafast polarization discriminating Mach-Zehnder all optical switch," *Appl. Phys. Lett.*, vol. 67, no. 25, pp. 3709–3711, 1995.

George Theophilopoulos (M'03) was born in Athens, Greece, in 1976. He received the Electrical and Computer Engineering Diploma from the National Technical University of Athens (NTUA), Athens, Greece, in 1999 and the Ph.D. degree in Electrical and Computer Engineering from the Photonics Communications Research Laboratory (PCRL), NTUA, in 2003.

From 2003 to 2004, he was a Senior Research Associate with PCRL. He is now with the Research Academic Computer Technology Institute (RACTI), Patras, Greece. His research interests include all-optical logic, all-optical switches, and optical networks. He is author or coauthor of 24 publications.

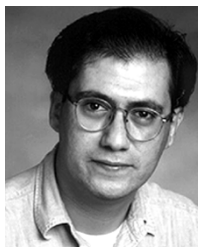
Marios Kalyvas, photograph and biography not available at the time of publication.

Konstantinos Yiannopoulos, photograph and biography not available at the time of publication.

Kyriakos Vlachos (SM'98–M'02) received the Ph.D. degree from the National Technical University of Athens (NTUA), Athens, Greece, in 2001.

From 2001 to 2003, he was a Member of the Technical Staff of Bell Laboratories, Lucent Technologies, The Netherlands, and in 2004 he was appointed an Assistant Professor in the Computer Engineering and Informatics Department of the University of Patras, Patras, Greece.

Dr. Vlachos has participated in various European and National research and development programs.



Emmanouel A. Varvarigos was born in Athens, Greece, in 1965. He received the Electrical and Computer Engineering Diploma from the National Technical University of Athens (NTUA), Athens, Greece, in 1988, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1990 and 1992, respectively.

Since 1999, he has been a Professor with the Department of Computer Engineering and Informatics at the University of Patras, Patras, Greece, where he is currently the Director of the Hardware and Computer Architecture Division, as well as the Director of the Networking Technologies Sector (NTS) of the Research Academic Computer Technology Institute (RACTI), Patras, Greece.

Hercules Avramopoulos (M'91), photograph and biography not available at the time of publication.