

Automatic Configuration in Future Semi-Transparent Distributed Optical Networks

Marco Ruffini, Donal O'Mahony, Linda Doyle.
Centre for Telecommunication Value Chain Research,
University of Dublin,
Trinity College, Dublin 2, Ireland.

Abstract

The recent commercial development of photonic switches has created many new semi-transparent optical architectures with core nodes made up of an all-optical switch controlled by an IP router. The fact that the nodes are transparent to the switched data complicates the implementation of operations such as interface and link configuration.

In this paper we present first an algorithm that allows these semi-transparent nodes to be self-configured. The algorithm optimizes the total configuration time by exploiting the capability of optical switches to execute many connections at the same time. Then we present a solution for the link configuration issue that does not require a pre-established control channel between the nodes. The two solutions represent a first step towards the complete automation of future semi-transparent optical networks.

The benefits brought by our methods are demonstrated by both simulation and testbed results.

1 Introduction

In the last few years the commercial availability of transparent optical switches has paved the way for the development of novel optical network architectures, whose main characteristic is the ability to create and tear down optical paths automatically.

This dynamicity is welcomed by both service providers and their customers: better exploitation of network resources

generates higher revenues for the former and lower costs for the latter [1], allowing a better match of supply and demand.

Currently different standardization bodies (mainly IETF, ITU-T and OIF), are finalizing the definition of the automated optical control plane, which is seen as the first step towards the optical revolution from the current static model to more dynamic future architectures. Their aim is to create a control plane capable of allocating new optical paths, depending on the network utilization and services demanded by the customers.

Their joint efforts have already produced a set of recommendations (ITU-T), implementation agreements (OIF) and request for comments (IETF), which equipment vendors have already started to implement in their products.

Besides the pure standardization activity however, photonic switches have also encouraged the development of new ideas of optical switching.

The Hikari router for example is a GMPLS-based optical router developed by NTT [2], [3]. One of its main characteristics is its ability to allocate bandwidth by creating new end-to-end optical paths when incoming traffic saturates the existing channels.

In [4] the authors propose a hybrid architecture where network nodes are capable of differentiating short-lived applications (e.g. name resolution requests) from long-lived ones (e.g. file transfers). While the former can be routed the latter benefit from a dedicated end-to-end optical path, avoiding routing at the IP layer.

In [5] an architecture for a regional access network is proposed based on a dual-fiber ring. The network allows data coming

from the distribution network to be converted to electrical form and routed at the IP level or to remain optical and be transparently switched.

We also have proposed in [6] and [7] an architecture we call Optical IP Switching (OIS) that combines an IP router and a transparent optical switch.

OIS is based on the creation of a lightpath to bypass the IP layer when an IP flow of significant size is recognised; instead of being created end-to-end, the path is locally generated and its extension proceeds in a distributed fashion.

Even though these architectures are substantially different, they all share a common feature: the core of the network is a semi-transparent device that can be implemented as a photonic switch controlled by an electronic router.

The router is equipped with optical-to-electrical and electrical-to-optical converters used respectively to terminate and generate optical signals. These interfaces are directly connected to the input and output ports of the optical switch, while the remaining ports accommodate the fibers linking to neighbouring nodes (Figure 1).

The router, the intelligent management entity controlling the switch, decides which ports need to be terminated and which ones transparently switched.

We found a similar architecture being adopted also in the GENI project [8], a joint effort to create a large-scale testbed for the research community. The optical nodes constitute a highly reconfigurable core made up of photonic cross connects (PXC) and reconfigurable add-drop multiplexers (ROADM).

These new generation networks all aim at providing optimal exploitation of the network resources, by automatically managing optical links and virtual IP paths.

The first issue to be solved however before automatic management can be implemented is the port discovery of the transparent optical switches. The importance of this automatic discovery is emphasized in [9], where the authors envision a network constituted by optical nodes with plug-and-play features.

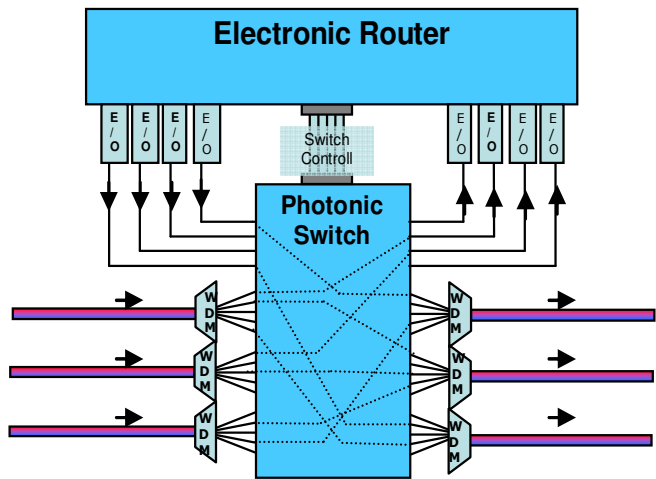


Figure 1 Semi-transparent optical router.

In this paper we discuss the problems associated with automatic discovery, and divide the problem in two subcategories. We refer to the discovery of the switch ports directly connected to the router interfaces as internal discovery, since it concerns the internal structure of the node. The discovery of the ports connecting to other nodes we call external or link discovery, as it concerns the discovery of the ports linking to neighbours.

In current opaque networks (e.g., Sonet/SDH) these tasks are not particularly complex because the switches are able to terminate all the optical signals at the same time and so are perfectly aware of which port is being used to send or receive messages.

In the semi-transparent node of Figure 1 instead, things are complicated by the fact that the optical switch is completely transparent to the data passing through its ports, and for economical and practical reasons only a limited number of ports can be terminated at the same time.

The rest of the article is organized as follows. First we provide a background for the discovery issues and some solutions already proposed in the literature. In Sections 3 and 4 we describe in details our solutions to the internal and external discovery problems and provide performance results obtained both through simulations and a real testbed. Finally, we list the main conclusions of our work.

2 State of the art

2.1 Internal discovery

Determining the connectivity between switch and router interfaces is a novel problem that only arises in semi-transparent network, where a transparent optical switch is connected to an electrical router.

Although a trivial solution would see a manual mapping of the interface position on the switch ports in a configuration file, this solution does not seem suitable and scalable for a scenario where a transparent node may have hundreds of ports and tens of interfaces. The manual process would require in fact hours of work, coordination of more people in case router and switch are located in different places, and would be prone to human errors. Automatic configuration is already widely deployed in electronic communication equipment. In optical systems however, since optical signal processing is still many years away, automatic configuration requires that optical signals are terminated and converted into electrical to be analysed by the router. The limiting factor is that both for economical and practical reasons there are only a limited number of router interfaces, and only a subset of ports can be terminated at the same time.

The techniques used by electronic equipment are based on exchanging configuration messages between the devices. If the elements are not directly connected together, different algorithms can be implemented to optimize the message exchange by retransmitting at intermediate nodes [10].

In the optical system of Fig.1 however, since the discovery requires cross-connecting the router interfaces, extensive interaction with the switching matrix becomes an essential part of the procedure.

A simple automatic implementation of the internal discovery process can be realized by checking all the ports in sequence.

The process starts connecting the first input port to the first output port, sending a message on each interface and waiting for the message at the receiving interfaces, as illustrated in

Figure 2. If no message is received a new connection is created: the first input port is connected to the second output port and messages are sent again. The process is replicated connecting input port 1 to each of the remaining ports. If after N iterations no message is received, the system connects the second input port to each output port in turn.

As soon as a message is received the node can identify the switch ports attached to the interfaces involved.

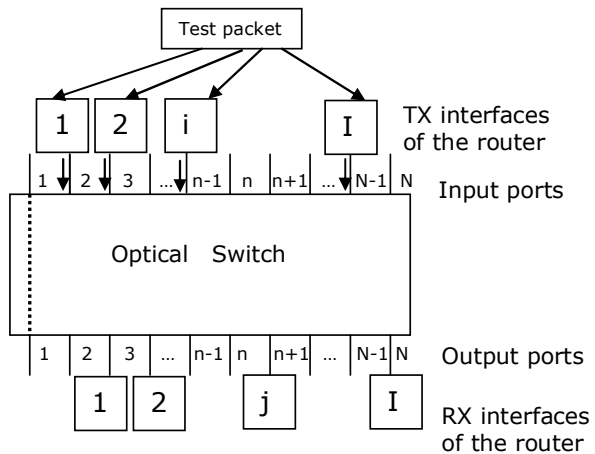


Figure 2 First Iteration of the serial scan.

The following procedure is straightforward: the switch connects the input port associated with the discovered TX interface to each of the still unresolved output ports sending a message each time. Every time a message is received a new association between a RX interface and a switch output port is created. When all the RX interfaces have been discovered, the node can apply the same procedure to the remaining TX interfaces. It simply connects the output port associated with one of the RX interfaces (for example the first discovered) to all the undiscovered input ports (this time sending a message on each unresolved TX interface). Each time a message is received a new association between a TX interface and the correspondent input port is discovered.

The number of iterations needed to accomplish the discovery process depends on how the interfaces are connected to the ports.

In this paper we will present a better solution to this problem which drastically reduces the configuration time, by exploiting important characteristics of typical photonic switches: the ability of executing many connections at the same time.

2.2 External discovery

The external discovery consists of determining which ports of the photonic switch link to neighbouring nodes and which wavelength is used on each port: a problem also known as Link Discovery.

The Common Control and Measurement Plane (CCAMP) of the Internet Engineering Task Force (IETF) has addressed the problem in the Link Management Protocol (LMP) [11], a part of the GMPLS protocol suite [12]. LMP describes the steps required for link discovery both for opaque and transparent optical switches. One of the main assumptions in LMP is that a bi-directional channel is already established between the two nodes on which the configuration procedure can be initiated.

The procedure consists of a node sending test messages and another node terminating each input port in turn until the test message is received. When a message is received a link adjacency is noted and the transmitter starts the same procedure on a different port.

Considering that a node is connected generally to more than one neighbour the procedure has to be repeated for every other neighbour.

In [13] the authors recognise the importance of this problem in semi-transparent optical networks and suggest that the discovery procedure could be initiated in parallel with all the neighbours in order to run the discovery process faster.

The procedure however still requires that a control channel is pre-established and that the nodes are manually informed whenever the discovery procedure needs to be initiated. Moreover the authors do not explicitly suggest an algorithm implementing the idea.

In this paper we propose a solution for the link discovery problem in semi-transparent optical networks that does not require a pre-established control channel. This procedure represents a step forward towards the complete automatic configuration of future optical networks.

In our scheme, the control channels between two peers is in fact selected during the link discovery process and embedded in one of the active wavelengths linking the two nodes.

We compare the performance of the algorithm we propose with an implementation following the idea introduced in [13].

3 Internal Discovery

We have developed a smart algorithm that executes the discovery process taking advantage of the capability of photonic switches to execute multiple input-to-output port connections at the same time.

The algorithm can be divided in two parts: the first collects all the information regarding the relative distance (in terms of port number) between each of the interfaces; the second determines the absolute position of a single TX-RX interface pair and uses the information collected during the first phase to determine the position of the remaining interfaces.

Figure 3 shows the state of the connections during the first iteration of the discovery process: the node connects each input port n to the output port $n+k$, where k is a number that is incremented after each iteration. For the first iteration k is equal to '0': input port 1 is connected to output port 1, input port 2 to output port 2, and so forth. After configuring the switch the node sends a message on each TX interface, while the RX interfaces wait for incoming messages. When RX interface j receives a message that was transmitted by TX interface i the value k is stored in the (i,j) position of a matrix of dimensions $I \times J$ (i.e. the total number of TX and RX interfaces).

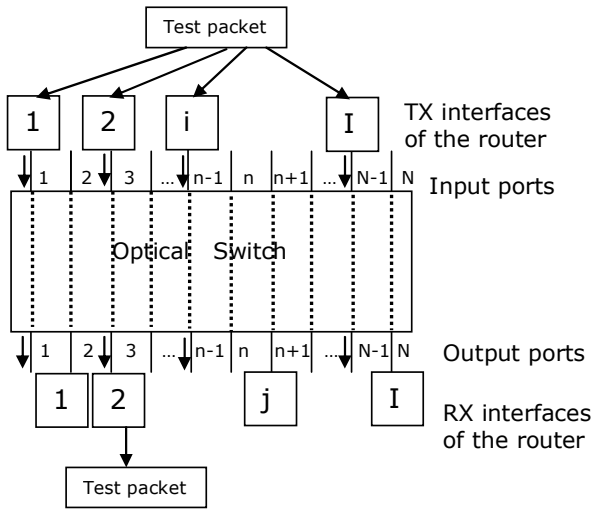


Figure 3 First iteration of the parallel algorithm.

At each new iteration all the connections are re-established with k incremented by 1 unit. During the second iteration for example input port 1 is connected to output port 2, and in general input port n to output port $n+1$. The last input port N is connected to output port 1, as illustrated in Figure 4. We can generalize this rule expressing the output ports as function of the input port n :

$$\begin{cases} \text{outputport}(n) = (n + k) \bmod(N) & \text{if } (n + k \neq N) \\ \text{outputport}(n) = N & \text{if } (n + k = N) \end{cases} \quad (1)$$

where $n = \text{inputport}$

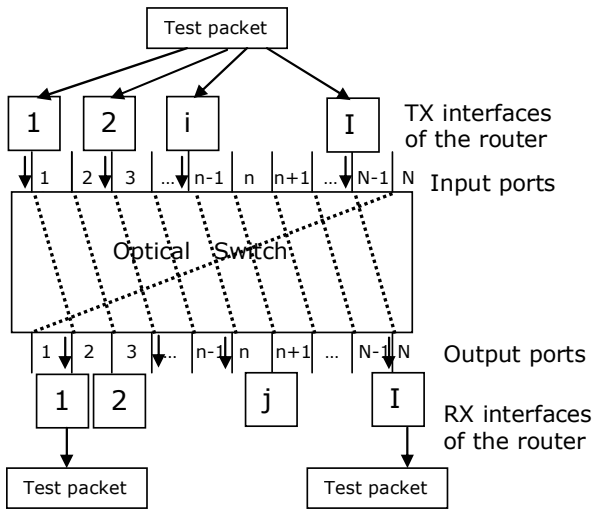


Figure 4 Second iteration of the parallel algorithm.

The process is iterated until $k = N-1$, that is when all the possible connections have been tested. The first phase of the algorithm always terminates in N iterations.

At this point the $I \times J$ matrix is completely filled in with values representing the relative distance between the ports connected to the interfaces.

The aim of the second phase of the algorithm is to determine the absolute location of a (TX,RX) interface pair, and use it together with the values stored in the matrix to calculate the position of the remaining interfaces.

This phase begins by picking a value k^1 from an arbitrary cell of the matrix (even though choosing the value that most often appears would speed up the process, by increasing the chances of receiving a message).

Each new iteration tests the connections one at a time, starting from input port 1. Each input port n is connected in turn to output port $n+k^1$, following rule 1) until a message is delivered. Once one message is received the position of the (i,j) interface pair is uniquely identified and can be used as reference point to calculate the location of the remaining interfaces without further switch activations.

Starting from the matrix cell (i,j) (Figure 5), corresponding to the newly discovered (TX,RX) interfaces pair, the algorithm can use the value stored in the generic cell (i_a, j_b) to determine the position of TX interface i_a or RX interface j_b . The cells (i,j) and (i_a, j_b) however must always lie either on the same row or on the same column, to keep a TX or RX interface as common reference. If (i,j) and (i_a, j_b) lie in the same row ($i = i_a$), the (i_a, j_b) value can be used to discover the position of RX interface j_b :

$$\text{OutPort}_{j_b} = (\text{InPort}_i + k(i, j_b)) \bmod(N) \quad (2)$$

If instead (i,j) and (i_a, j_b) are in the same column ($j = j_b$), the value stored in (i_a, j_b) can be used to determine the position of TX interface i_a :

$$\text{InPort}_{i_a} = (N + \text{OutPort}_j - k(i_a, j)) \bmod(N) \quad (3)$$

At each step the newly discovered position of (i_a, j_b) can be used as reference point for the next step, as shown in Figure 5.

$k(1,1)$	$k(1,\dots)$	$k(1,j)$	$k(1,\dots)$	$k(1,J)$
$k(\dots,1)$	$k(\dots,\dots)$	$k(\dots,j)$	$k(\dots,\dots)$	$k(\dots,J)$
$k(i,1)$	$k(i,\dots)$	$k(i,j)$	$k(i,\dots)$	$k(i,J)$
$k(\dots,1)$	$k(\dots,\dots)$	$k(\dots,j)$	$k(\dots,\dots)$	$k(\dots,J)$
$k(I,1)$	$k(I,\dots)$	$k(I,j)$	$k(I,\dots)$	$k(I,J)$

Figure 5 Matrix that stores the values of mutual distances between the TX and RX interfaces.

All the interfaces are discovered when at least one cell per row and one cell per column of the matrix have been exploited. The total number of switch activations required by the parallel algorithm is between $N+1$ and $2N$.

3.1 Redundancy of information

We can easily notice that part of the information stored in the matrix was not used during the discovery process. It is possible to eliminate this redundancy and save some iterations by rearranging the discovery process: we could execute the first part of the algorithm until a message is received, and use the k value found to determine the interface-to-port relation of a pair (i,j) of TX-RX interfaces in the second part of the algorithm. Then, returning to the first phase, we could stop the iterations as soon as just enough information is collected to complete the discovery process. However this operation is only safe if none of the ports is connected to a neighbouring switch.

An example is shown in Figure 6, where the left node sends a test message through TX interface 2 and receives it through RX interface j , even if the two interfaces are not directly connected together. The loop accidentally created by switch 2 in fact misleads the first node that erroneously inserts a k value of '1' in the position $(2,j)$.

If the node had followed the procedure described in § 3 instead, it would have realized the mistake and avoided using the cell $(2,j)$ for its calculations.

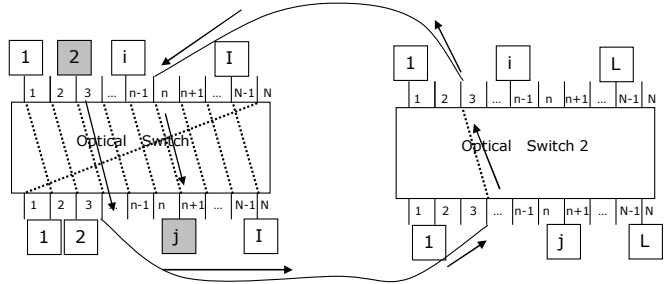


Figure 6 Example of configuration that may confuse the discovery process.

3.2 Cases with $N \times M$ switches and $I \neq J$

A more general implementation, valid for $N \times M$ switches with I TX interfaces and J RX interfaces can be easily deduced.

Having $I \neq J$ simply implies that the matrix we use to collect data in the first phase is rectangular with dimensions $I \times J$ instead of being square. This has no further implications in the use of the algorithm.

If our switch has N input ports and M output ports instead, with $N \neq M$, we have to distinguish the case $N \leq M$ from the case $N > M$. In the first case we should arrange the connections in the first phase of the algorithm following the rules:

$$\begin{cases} \text{outputport}(n) = (n + k) \bmod(M) & \text{if } (n + k \neq M) \\ \text{outputport}(n) = M & \text{if } (n + k = M) \end{cases} \quad (4)$$

where $n = \text{inputport}$

Here at each iteration we shift the output ports with respect to the input ports, exactly as in the case of a $N \times N$ switch. The only difference in the formulas is that we calculate the output ports modulo M (the total number of output ports).

In the case $N > M$ instead at each iteration we need to shift the input ports respect to the output ports:

$$\begin{cases} \text{inputport}(n) = (n+k) \bmod(N) & \text{if } (n+k \neq N) \\ \text{inputport}(n) = N & \text{if } (n+k = N) \end{cases} \quad (5)$$

where $n = \text{outputport}$

Similar modifications apply to formulas 2) and 3) used in the second phase of the algorithm.

3.3 Algorithms comparison

We have simulated and compared the performances of our parallel algorithm with the serial solution presented in 2.1, using $N \times N$ switches connected to routers with same number of TX and RX interfaces. We use the number of switch activations as a performance metric, since switching time is the main source of delay in practical implementations. We envisage that transparent optical devices will vary in terms of the ratio of switch ports to local terminations. For this reason we have expressed the number of TX-RX interfaces relatively to the number of ports, using values of $N/8$ and $N/4$, where we expect typical values to be closer to $N/8$, so that most of the ports can be allocated to transparently switch traffic between adjacent nodes.

For each configuration we have simulated 1000 trials using the Monte Carlo method to randomly connect the router interfaces to the switch ports.

The graphs in

Figure 7 report the average number of switch activations required to complete the discovery process, as a function of the switch dimensions (N).

Figure 7.A reports the results when the number of TX and RX interfaces is $N/8$. We can see that the parallel algorithm performs much better than the serial scan, and the difference increases with the number of ports: we have a 6-fold improvement when considering a 512 ports switch. The improvement tends to decrease in

Figure 7.B, where the number of interfaces is $N/4$. The reason is that while the performance of the parallel algorithm does not depend on the number of interfaces used, the serial algorithm

improves with a higher number of interfaces; in fact in those cases it is statistically easier for the serial scan to find the first interface to use as reference for the discovery of the others.

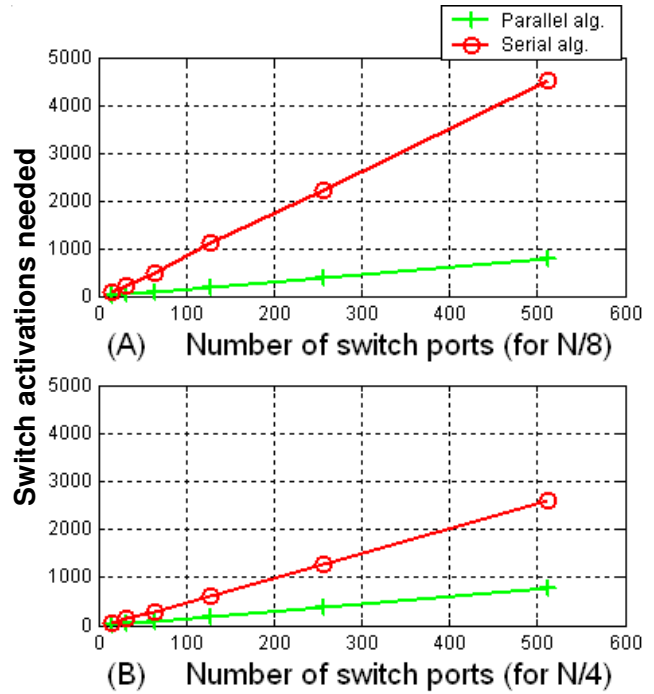


Figure 7 Average number of switch activations required to complete the discovery process.

In addition to their average values it is interesting to examine the distribution of the simulation results. For this reason we have collected the results of the trials simulated into histogram graphs. Figure 8 shows the distribution obtained with a 512-port switch (other cases present similar behaviour). We can see that for the parallel algorithm the distribution is generally uniform and confined to a narrow interval of values (513 to 1024 switch activation precisely, as envisaged in Section 3). The serial scan instead presents a decreasing exponential envelope and the results span over a much wider interval, which decreases when the number of interfaces increases.

From the above results we can conclude that the parallel algorithm outperforms the serial scan: the number of switch activations required is lower in the average and is distributed over a smaller interval. From a practical point of view a system implementing the parallel algorithm would have shorter and less variable configuration times.

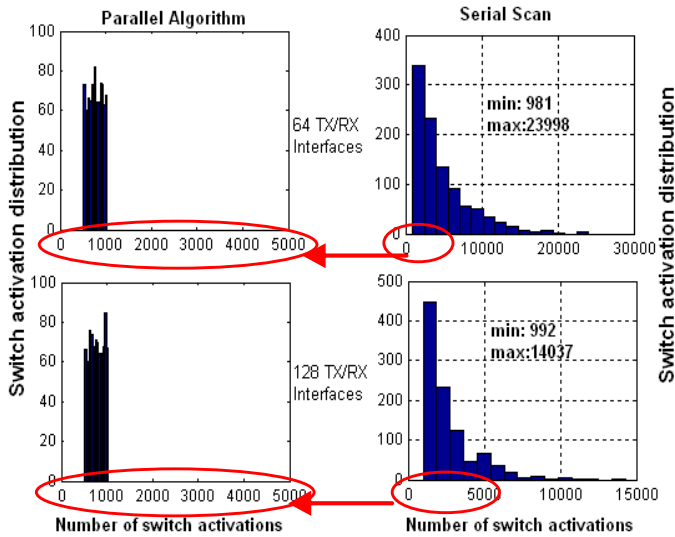


Figure 8 Distribution of the number of switch activations required.

We have also tested the self-configuration algorithm in a real testbed [6]. The optical router on which the algorithm is implemented is made up of a blade server connected to a photonic switch through a dedicate Ethernet connection. The server has three Ethernet ports which connect to three Gigabit Ethernet optical transceivers, while the optical switch is a 16 x 16 ports MEMs-based Glimmerglass System-300. The server controls the switch connections through a direct TCP link using the TL1 control protocol.

The messages used to test the ports are delivered through our protocol stack in form of UDP packets and are physically transmitted and received by the Gigabit Ethernet transceivers.

Because the MEMs-based switch elements need time to settle after being activated a deliberate delay must be introduced before sending a test message on the link. A further delay is introduced by the time needed by the switch to process two consecutive TL1 commands and send an acknowledgment back to the server through the TCP connection.

We have achieved an average value of 47 milliseconds per iteration in our system, which includes the time needed to create and delete the connections, plus some guard intervals to avoid the overlapping of certain operations.

If we apply this value to the previous simulated results we can infer typical configuration times of larger systems. Considering

a 512 x 512 ports switch with $N/8$ number of interfaces for example we can see that our algorithm would terminate in 36 seconds in the average, where the serial algorithm would need 3 minutes 32 seconds. Moreover while the parallel algorithm has also a very low maximum configuration time (48 seconds) the serial algorithm may require up to 18 minutes 48 seconds to configure the system, due to the long tail of the iteration distribution.

4 Link discovery

After completing the internal discovery, the node starts the discovery of its external links.

Here we propose a solution to the link discovery problem in transparent switches, that does not need the pre-provision of a control channel. The control channel is in fact provisioned in-band during the link discovery process and allocated in a wavelength that can also be used to transport data traffic (a prioritized virtual link could be established within the wavelength to favor the control traffic over the data traffic).

In the GMPLS architecture, where each node may be connected to a central management entity, an out of band control channel may be preferred. However we believe that the trend of semi-transparent optical networks, with the IP layer directly on top of the WDM layer, will follow a more distributed approach, similar to current IP networks.

In distributed architectures, where control channels are only allocated between peering neighbours, the in-band approach may be preferred instead because it eliminates the need to set up a separate control channel and allows easier automation of discovery procedures.

The protocol we propose is made up of two distinct processes: one scans the receiving ports for incoming messages while the other sends TEST messages through the transmitting ports.

The process scanning the input ports runs permanently in background on every active node and terminates one after the other all the undiscovered optical ports using receiving router interfaces. All the receiving interfaces in idle mode (i.e. those not being used for data traffic) can be used contemporarily.

After a pre-configured listening time, the receiving interfaces are connected to the next input ports: the process goes on scanning continuously the undiscovered input ports.

The process beaconing the output ports of the switch instead initiates when a node is started up and terminates once the discovery is completed. This procedure sees the router's optical transmitters beaconing TEST messages through the switch output ports. The parameters needed in the message are described in Table 1.

Table 1 Contents of the configuration messages.

TEST	ACK	DOUBLE_ACK
Message_type	Message_type	Message_type
Message_ID	Message_ID	Message_REF ID
Source_ID	Message_REF ID	Source_ID
Output_port	Source_ID	Destination_ID
Wavelength	Destination_ID	

The aim of the algorithm is to establish a control channel with each peering neighbour and to discover the suitable wavelength on each output port (an essential information when the switch ports are connected to WDM mux/demux).

Wavelength tunability is an important feature of optical transmitters, as it allows testing each port on all the wavelengths using the same transmitter (i.e. within a single switching time).

If each transmitter can only beacon at one fixed wavelength instead, one switch activation is required every time a port is tested on a different wavelength.

As soon as a TEST message is received, the node will link the input port to the sender node and to the wavelength value encoded in the message, entering this information in its link table (as illustrated in Figure 9). If the link is the first discovered from that node, it is designated as the control channel in the downstream direction, and the router permanently connects a receiving interface to that input port. The message also serves as trigger for the receiving node (B in the figure) that starts beaconing on all the undiscovered outgoing ports.

Node B should reply with an ACK, making A aware that the port over which that particular TEST message was sent is connected to node B.

However since there is no control channel discovered yet from node B to A, B will put the ACK on an outbox queue, whose content is relayed every time an output port is beaconed, before each TEST message.

When it finally receives the ACK message, node A links it to the previously sent TEST message, discovering a new output port. Being the first output port linking to node B, A uses this port as outgoing control channel to B.

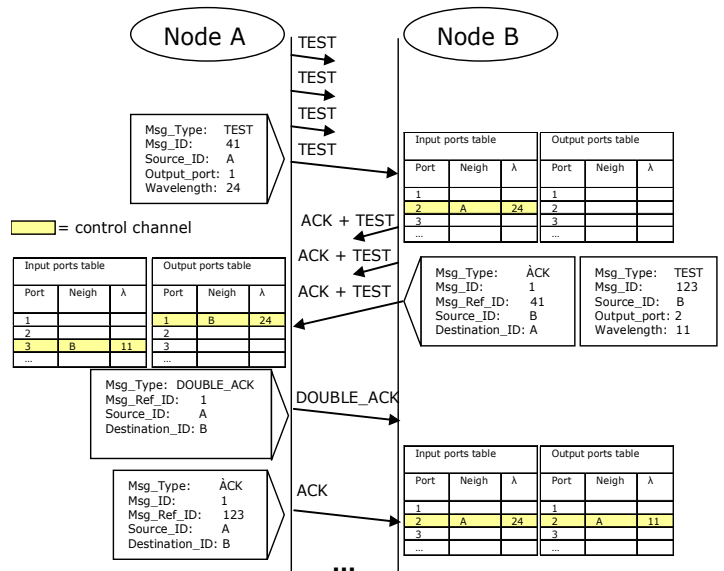


Figure 9 Example of messages exchanged by the link discovery protocol.

The next step consists of A sending a DOUBLE_ACK to B using the new established control channel, following a 3-way handshake approach. The DOUBLE_ACK informs B to delete the related ACK message from its outbox list.

Once the control channel is established in both directions, the two peers can exchange all the pending ACK messages.

The procedure continues in the same way in parallel with all the neighbours, operating asynchronously at all the nodes. A proper timer will terminate the beaconing process when no new link is discovered over a pre-defined time interval.

The parameter that most influences the algorithm performances is the time interval between the test of adjacent ports. Output ports should ideally be beacons one after the other as fast as possible. Our implementation uses MEMs-based devices, with switching times of around 20 ms.

For the input port scanning instead the best interval time (RX_SCAN_TIME) varies depending on the number of ports and interfaces. Intuitively each node should test the same input ports until all the neighbours have beacons all their output ports over the complete range of wavelength. Such time increases with the number of switch ports and wavelengths and decreases with the number of interfaces available for beaconing the output ports.

In operational environments however nodes use switches of different sizes and different number of interfaces, making the optimization of the RX_SCAN_TIME impossible. In Section 4.1 we show, the dependency of the configuration time on this parameter.

The link discovery procedure can also be used when new links are added to already operating nodes. In this case however the output port scanning process needs to be triggered manually on at least one side of the link. Since the process only uses idle interfaces, the discovery can be executed without interfering with regular router operations. Other implementations however could see the output port discovery process triggered automatically after a link interruption or at regular time intervals, to keep the link table up-to-date.

4.1 Algorithm performances

We have conducted simulations of the proposed link discovery protocol, considering switches of different sizes, different number of peering neighbours, and varying the number of router interfaces and the RX_SCAN_TIME parameter.

The test topology considers M nodes connected in a full mesh using W wavelengths per link. Each node is constituted by a router with T TX and R RX interfaces and a photonic switch of $N \times N$ ports. We have chosen values of 16, 64 and 256 for N

and respectively values of 3, 10 and 30 for W . We have considered the case of non-tunable transmitter (the worst case scenario): the number T of transmitters should be enough to cover the M control channels and the complete range of wavelengths W . For convenience reasons, all the nodes use transmitters operating at the same wavelength to establish the control channels. The number of TX interfaces at each node is $T = W + M - 1$, $M - 1$ is the number of peering neighbours. We also assume $T = R$.

We have considered for comparison the idea proposed in [13]. Even though the authors do not go into the details of the discovery algorithm, we can infer an approximate performance calculation based on an implementation of the idea they describe. On startup, a node signals the start of a discovery session to all the peering neighbours: the same node gives to all the neighbours the proper timing so that the scanning process can proceed in a synchronous and orderly manner. The approximate number of switch reconfiguration needed by this algorithm is:

$$Reconf = \frac{(N - W) \cdot (N - R)}{2 \cdot R} \quad (6)$$

In fact the node has $N - T$ undiscovered ports that are tested using W of the T interfaces at the same time (here we consider $T = W$, since all the control channels are out-of-band): this requires $(N - T)/T = (N - W)/W$ switch reconfigurations to cover all the output ports. Considering then that we use non-tunable transmitters, each port needs to be tested on each wavelength using a different TX interface, so the number of reconfiguration increases by a factor of W . For each of these reconfigurations the neighbouring nodes needs to scan all their incoming ports: considering broadband receivers each operation can be done in $(N - R - Discovered_Ports)/R$ steps. At the beginning, when many ports are undiscovered, this value will be close to $(N - R)/R$, while towards the end it will be close to '0' (considering that most of the ports are linked to neighbours): we average this behaviour by using a value of $(N - R)/2R$.

Considering that the process needs to be repeated similarly for the discovery of the input ports we need to add a further multiplication factor of 2:

$$Reconf = 2 \cdot \frac{(N - W) \cdot (N - R)}{2 \cdot W \cdot R} \cdot W \quad (7)$$

Assuming the same number of TX and RX interfaces ($T = R$), we get:

$$Reconf = \frac{(N - W) \cdot (N - R)}{R} \quad (8)$$

This calculation however considers a worst-case scenario, where a complete scan of all the ports is required each time. On average, only half the ports need to be scanned, so we can again divide by 2, which brings back to formula (6).

Figure 10 reports the results of the link discovery simulations, run using the topology parameters shown in Table 2.

Table 2 Parameters used for the simulations.

	Sim A	Sim B	Sim C
Neighbours number	3	5	7
Switch size	16 x 16	64 x 64	256 x 256
Wavelengths per link ¹	2	9	29
Interfaces number	5	14	36
Switching time ² (ms)	60	60	60

On the x axis we report the sequential number of the input and output ports and in the y-axis the time of the discovery expressed in milliseconds. The graphs on the left side are averaged over different simulation runs and each line corresponds to a different value of RX_SCAN_TIME. In the right side instead we report the best and worst results (representing the lower and upper limits), obtained using the most performing RX_SCAN_TIME value (e.g., 500 ms for Figure 10.A). Figure B and C report the results for different network topologies.

¹ This number does not include the wavelength used as control channel.

As we can see the choice of the RX_SCAN_TIME parameter is moderately relevant as far as the value selected is not excessively small (red line in the left graph of B and C). Moreover, by comparing the best values for the RX_SCAN_TIME with the simulation parameters in Table 2 we could confirm that, as already anticipated, the best value is directly proportional to the number of ports of neighbour switches and to the link wavelengths, while inversely proportional to the number of transmitting interfaces used by the neighbours:

$$RX_SCAN_TIME \propto \frac{N_{neigh} \cdot W_{towards-neigh}}{T_{neigh}} \quad (9)$$

For a practical implementation, considering that a different optimum value can be estimated for each neighbour and that choosing a too low value has much worst effects than choosing a too high value, the best tactic is to choose the largest of all the estimated optimum RX_SCAN_TIME values.

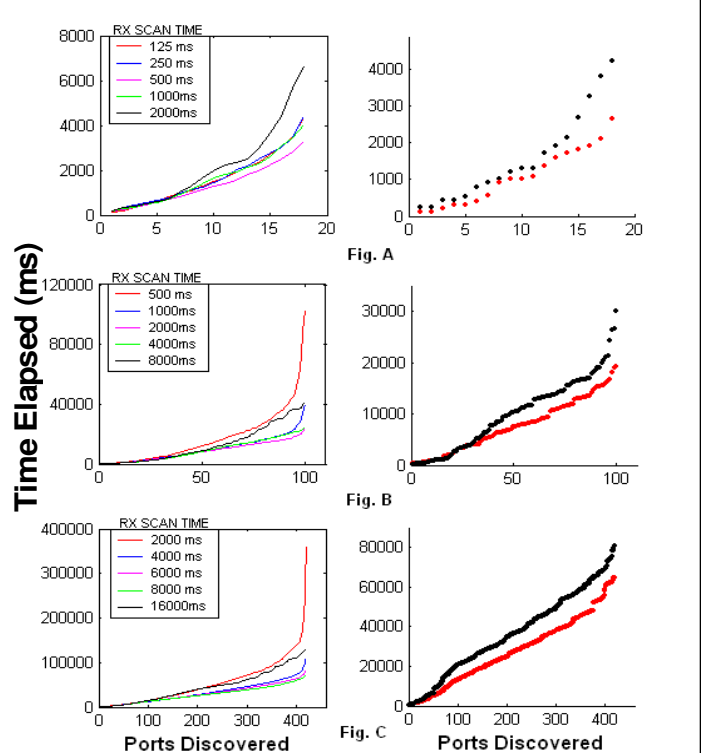


Figure 10 Simulation results of our synchronous algorithm.

By applying formula (6) to the simulated topologies we can compare the average time that could be obtained with the

algorithm described in [13]. Table 3 reports the comparison. The synchronous algorithm completes the task 4 times faster than our algorithm in the A and B cases, and less than 2 times faster in case C. However if we consider that the algorithm we propose also establishes the control channel within the same time (an operation that may require several minutes if manually operated) we can state that the algorithm we propose could reduce the link discovery time from 1 to 2 orders of magnitude.

Table 3 Comparison between the synchronous and asynchronous algorithms (time values are displayed in seconds).

Simulations	Synch Algorithm ²	Synch Algorithm + manual control channel ³	Async Algorithm
Sim A	0.858	900.858	3.264
Sim B	5.785	1,505.785	23.753
Sim C	41.433	2,141.433	72.467

5 Conclusions

In this paper, we have targeted the auto-configuration issue for routers operating in semi-transparent optical architectures, where part of the traffic is converted from optical to electrical and part is switched optically. We have presented first an algorithm to determine the connections between router and a transparent optical switch within the same node. In conjunction we have proposed a method for link discovery that does not require a pre-established control channel.

Our simulation results for the internal discovery process show that the algorithm we propose, by exploiting the photonic switch capabilities of executing connections in parallel,

² We have considered that each iteration requires 60 ms between creating a connection, deleting it, plus some guard intervals.

³ We considered, based on our experience with the testbed that approximately 5 minutes may be required to manually create a control channel to each neighbour.

achieves much lower configuration times compared with a simpler method that only avails of one connection at a time.

In the case of link discovery, the algorithm we propose seems slower than a reference algorithm presented. However if we include in the calculation the time needed to configure a control channel for each neighbour, we see that our solution achieves improvements from 1 to 2 orders of magnitude.

The implementation of the algorithm in a real testbed moreover proved the practical functionality of our method and allowed more accurate measurement of the configuration times, permitting the deduction of typical configuration times of larger optical systems.

References

- [1] A. Kirstädter, et al.: Business models for next generation transport networks, Springer Photonic Network Communications, vol. 10, no. 3, (November 2005), pp. 283-296.
- [2] K. Sato, et al.: GMPLS-based photonic multilayer router (Hikari router) architecture: an overview of traffic engineering and signaling technology, IEEE Communications Magazine, vol. 40, no. 3, (March 2002), pp. 96-101.
- [3] E. Oki, et al.: Dynamic multilayer routing schemes in GMPLS-based IP+optical networks, IEEE Communications Magazine, vol. 43, no. 1, (January 2005), pp. 108-114.
- [4] M. Veeraraghavan, et al.: Architectures and protocols that enable new applications on optical networks, IEEE Communications Magazine, vol. 39, no. 3, (March 2001), pp. 118-127.
- [5] M. Kuznetsov, et al.: A next-generation optical regional access network, IEEE Communications Magazine, vol. 38, no. 1, (January 2000), pp. 66-72.
- [6] M. Ruffini, D. O'Mahony, L. Doyle: A testbed demonstrating optical IP switching (OIS) in disaggregated network architectures, Proc. of IEEE Tridentcom '06, (Barcelona, Spain, March 2006), vol. 1, pp. 156-161.
- [7] G. Mulvihill, M. Ruffini, F. Smith, L. Barry, L. Doyle, D. O'Mahony: Optical IP Switching: a solution to dynamic lightpath establishment in disaggregated network architectures, Proc. of ICTON 06, (Nottingham, UK, February 2006), vol. 3, pp. 78-81.
- [8] T. Anderson, et al.: GENI: Global environment for network investigations, Conceptual design project execution plan, available at <http://www.geni.net/GDD/GDD-06-07.pdf>, (January 2006).
- [9] I. Cerutti, A. Fumagalli, R. Hui, A. Paradisi, M. Tacca: Plug and Play networking with optical nodes, Proc. of

- ICTON 06, (Nottingham, UK, February 2006), vol. 3, pp. 133-138.
- [10]I. Chlamtac, et al.: Topologies for high-speed optical networks, IEEE/OSA Journal of Lightwave Technology, vol. 11, no. 5/6, (May/June 1993), pp. 951-961.
- [11]J. Lang, et al.: Link Management Protocol (LMP), IETF RFC 4204, (October 2005).
- [12]E. Mannie et al.: Generalized Multi-Protocol Label Switching (GMPLS) architecture, IETF RFC 3945, (October 2004).
- [13]G. Ellinas, et al.: Network control and management challenges in opaque networks utilizing transparent optical switches, IEEE Communications Magazine, vol. 42, no. 2, (February 2004), pp. S16-S24.