Novel Scheme for Packet Forwarding without Header Modifications in Optical Networks

Wessing, Henrik; Christiansen, Henrik Lehrmann; Fjelde, Tina; Dittmann, Lars

Published in:
Journal of Lightwave Technology

Link to article, DOI:
10.1109/JLT.2002.800268

Publication date:
2002

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Novel Scheme for Packet Forwarding Without Header Modifications in Optical Networks

Henrik Wessing, Henrik Christiansen, Tina Fjelde, and Lars Dittmann

Abstract—We present a novel scheme for packet forwarding in optical packet-switched networks and we further demonstrate its good scalability through simulations. The scheme requires neither header modification nor any label distribution protocol, thus reducing component cost while simplifying network management.

Index Terms—Algorithms, communication systems routing, networks, optical communication, packet switching, simulation.

I. INTRODUCTION

The growth of the Internet is continuously setting new demands for flexible transport of high-speed data traffic. It is expected that the bottleneck in the future networks will be the switch nodes, if all traffic is treated as electrical signals. Meanwhile, optical technologies have emerged very rapidly and all-optical networks with optical switch nodes are proposed as a solution to these challenges in the near future. Optical network elements offer only limited functionality and it therefore fits well with Multiprotocol Label Switching (MPLS), where the requirements to the core nodes are reduced [1].

The MPLS concept is characterized by a separation of the network in edge and core nodes. The edge nodes are responsible for resolving a route through the network and the core nodes are responsible for forwarding the data packets using only local information.

Packet forwarding in the MPLS-based core network, which could be an optical network, can basically be subdivided into three groups based on the operation principle. The first is an approach, where each path through the core network is designated by a globally unique label. The distribution of these labels will, however, be very complicated and time consuming. The second approach is the well-known MPLS label swapping [2], where the incoming header is modified by the core node. This leads to much simpler label distribution, but header modification is generally necessary, i.e., the label has to be changed or replaced. This is of minor concern in electrical core nodes, but in optical core nodes this tends to be a major challenge although optical header modification has been demonstrated in, e.g., [1], [3], [4]. The third approach is to carry all the forwarding information solely in the header, thus avoiding lookup tables in the core nodes. This could be done by carrying all the labels in a stack in the header and rotate or pop the stack for each traversed node. In the framework of the P-Internet protocol (PIP) that was suggested in 1993 as a successor for the Internet protocol (IP), the stacking of the forwarding information is proposed as a chain of forwarding table index fields (FTIF) [5]. The index field to be used is indicated by an offset value field, which is updated for each traversed node. It is thus seen that all these label stacking techniques still require some sort of header modification.

Thus, the conventional forwarding schemes are either inflexible or require header modification, which is very difficult to implement in an all-optical switch using currently available technologies.

In this paper, we propose a scheme, the key identification scheme (KIS), that addresses these problems by combining high flexibility while avoiding header modification.

The idea behind the scheme is presented in Section II including a description of the used algorithms and how the scheme can be implemented in a network. As scalability is very important, this is addressed in Section III and evaluated through simulations. A discussion of the scalability issues is carried out and in Section IV a conclusion of this work is given.

II. KEY IDENTIFICATION SCHEME (KIS)

The basic idea in the KIS is sketched in Fig. 1, where a packet is transmitted through a core network. Throughout the paper, the considered networks are characterized by electrical edge nodes and optical core nodes. The scheme is also valid for networks with electrical core nodes, however, the problems with header modification is only present in optical core nodes.

At the ingress edge node the path through the core network is resolved and a label is created and added to the packet. At the first core node an arithmetic operation is performed based on the label and a node-specific key and the result of this operation is used to designate the correct output port. At the subsequent
nodes, the output port is designated using the same function on
the same label but with another node-specific key.

For such a scheme, which is introduced in the following sec-
tion, it is important that the processing in the core nodes can be
done fast and that the scheme scales acceptably as the network
size increases.

A. The Chinese Remainder Theorem (CRT)

The implementation of KIS is done rather straightforwardly
using the Chinese Remainder Theorem (CRT), with which a
label is created based on two arrays. These two arrays contain
all the node-specific keys and all the desired output information
for a given path through the core network. Thus, when a path
through the network is desired, one array is formed by all the
node-specific keys and the other array is formed by the output
ports for the given path. Given these two arrays it is possible to
compute a label with the properties depicted in Fig. 1.

These two arrays are in the following denoted by \( \bar{n} \) and \( \bar{n} \)
for the keys and the desired output ports, respectively. Considering
a path through the core network comprising \( k \) nodes, the two
arrays are given as

\[
\bar{a} \leftrightarrow (a_1, a_2, \ldots, a_k) \quad \text{and} \quad \bar{n} \leftrightarrow (n_1, n_2, \ldots, n_k). \quad (1)
\]

The two arrays in (1) are used to create the label, which is
denoted as the scalar \( a \), from which it is possible to restore the
array \( \bar{a} \) using array \( \bar{n} \). Hence, the following necessary operations
and array definitions are dedicated to compute the label \( a \). First,
a new scalar \( n \) is defined as the multiple of all \( (k) \) elements in
the array \( \bar{n} \).

\[
n = n_1 \cdot n_2 \cdots n_k. \quad (2)
\]

Using \( n \) and \( \bar{n} \), a new array \( \bar{a} \) is created so that \(\bar{a} \mod n = 0\)
for all \( j \neq i \), which is satisfied in the following equation. Note
that the modulo operation is the remainder from integers divi-
sion.

\[
m_i = \frac{n}{n_i}, \quad \text{for } i \leq k \quad (3)
\]

Yet another array \( \bar{c} \) is created based on the arrays \( \bar{a} \) and \( \bar{n} \):

\[
c_i = m_i (m_i^{-1} \mod n_i), \quad \text{for } i \leq k \quad (4)
\]

The term \( m_i^{-1} \) denotes the multiplicative inverse of \( m_i \mod n_i \)
defined by \( m_i^{-1} m_i \mod n_i = 1 \). This requires that \( m_i \) and \( n_i \) are relative primes, i.e., they should have no
common divisors larger than one (\( \gcd(m_i, n_i) = 1 \)). Because
\( m_i \) is the multiple of all \( n \)-elements except \( n_i \), all the \( n \)-ele-
ments should be relative primes to satisfy the requirement.
The multiplicative inverse is easily computed, e.g., by using a few
lines of recursive C code [6].

The scalar \( a \) is finally calculated using \( \bar{a} \) and \( \bar{c} \):

\[
a = (a_1 c_1 + a_2 c_2 + \cdots + a_k c_k) \mod n, \quad (5)
\]

Given the scalar \( a \) and the array \( \bar{n} \), it is straightforward to
restore the array \( \bar{a} \) using the following operation:

\[
a_i = a \mod n_i, \quad \text{for } i \leq k \quad (6)
\]

Hence, it is seen that the CRT can be used to construct a scalar
\( a \) from two arrays \( \bar{n} \) and \( \bar{n} \), where all the elements in the latter
array have to be pairwise relative primes. Having \( a \) and the array
\( \bar{n} \), each element of the array \( \bar{a} \) can be restored.

Please refer to [6] for a more stringent proof of the algorithm.

B. Network Implementation

When a path through the core network is resolved, the array \( \bar{n} \)
formed by all the node-specific keys for the nodes involved in
the path, as sketched in Fig. 1. The desired output port for each
of the involved nodes is provided as an element in the array \( \bar{n} \).
The label (corresponding to the scalar \( a \)) is then computed using
the provided algorithm and at each traversed node the operation
given in (6) is carried out

\[
\text{output}_{\text{node}} = \text{label} \mod \text{key}_{\text{node}}, \quad \text{for all nodes}. \quad (7)
\]

It should be stressed that all keys must be unique and they
should be larger than the size of the output information, they
are used to decode. This output information contains the output
port, however, it is possible to encode information as type of
class, priority, or what else is desired for the specific core node.
Furthermore, all the keys should be pairwise relative primes, i.e.,
the greatest common divisor for any pair of keys in the network
should equal one. This constraint requires that the scalability is
investigated, which is done in Section III.

The KIS using the CRT can be implemented in an MPLS
network in the following three steps.

1) When the network is configured, each core node is as-
signed a key that satisfies the requirements previously de-
scribed.

2) Information of the topology of the network and the as-
signed keys are provided to the edge nodes. This can be
done with conventional routing protocols like OSPF, RIP,
etc.

3) When a packet enters the network at the edge node, the
label is computed using the CRT. It is noted that the CRT
computations are performed in the edge nodes, which op-
erate in the electrical domain. Finally, the label is added
to the packet that is transmitted to the core network.

Within each of the core nodes, the output information is cal-
culated using the modulo operation given in (7) based on the
node-specific key and the attached label. Normally, a modulo
operation is calculated using a division, a multiplication, and
proper truncation. However, as the key is rarely changed for
each node, it is advantageous to compute the inverse value of
the key and store this in internal memory. Hence, the modulo
operation is performed with two multiplications and truncation
and, consequently, the output port can be computed very effi-
ciently well within the time duration of a packet even for small
packets with length \( \sim 1 \mu s \). This size is the considered optical
packet length for the IST project DAVID [7] and in the European
ACTS project KEOPS a packet length of 1.646 \( \mu s \) was used [8].

C. Example

As an example, it is desired to transmit a packet through a
core network using a path comprising three core nodes. When
the network was configured, these nodes were assigned the keys
25, 14, and 37 and the output ports 4, 2, and 3 should be used,
Fig. 2. Packet traversing three nodes with three unique keys. The label is 2704 and the following calculations are performed: 2704 mod 25 = 2, 2704 mod 14 = 2, and 2704 mod 37 = 3.

respectively. Note that even though only 37 is an absolute prime, all the keys are valid as no pair of keys has common divisors greater than one. The keys and the outputs ports are illustrated in Fig. 2.

The key and the output array are constructed corresponding to the arrays $\mathbf{n}$ and $\mathbf{\pi}$ in (1)

\[
\mathbf{n} = \text{outputs} = (4, 2, 3) \\
\mathbf{\pi} = \text{keys} = (25, 14, 37).
\]  

(8)

Using these arrays it is possible to calculate $\eta = 25 \cdot 14 \cdot 37 = 12950$, the array $\mathbf{\eta}$, and the multiplicative inverses

\[
\mathbf{\eta} = (518, 925, 350) \\
518^{-1} \mod 25 = 7 \\
925^{-1} \mod 14 = 1 \\
350^{-1} \mod 37 = 24.
\]  

(9)

(10)

The array $\mathbf{\bar{\pi}}$ is then computed based on (4)

\[
\mathbf{\bar{\pi}} = (518 \cdot 7, 925 \cdot 1, 350 \cdot 24) \mod 12950 \\
= (3626, 925, 8400).
\]  

(11)

Finally, the label corresponding to the scalar $\alpha$ is calculated using (5)

\[
\alpha = (4 \cdot 3626 + 2 \cdot 925 + 3 \cdot 8400) \mod 12950 \\
= 41554 \mod 12950 = 2704.
\]  

(12)

The example is depicted in Fig. 2, where the packet with the computed label enters the core network. In the three core nodes, the following operations are carried out:

1) $2704 \mod 25 = 4$

2) $2704 \mod 14 = 2$

3) $2704 \mod 37 = 3$.

It is seen that the KIS offers high flexibility as each path in the network can be uniquely chosen simply by calculating the label.

III. SCALABILITY OF THE SCHEME

As stated in Section II, the scalability of the scheme, when the network size increases is important due to the requirements to the keys. It is not easy to estimate the number of core nodes that will be required in the core MPLS network in the future. However, an investigation of a large number of backbone networks operated by different Internet service providers is given in [9] and it is indicated that 10–30 core nodes are sufficient and a similar number is used in the European ACTS project OPEN [10], which considers an all-optical network based on optical cross connects. Therefore, the focus is on networks with less than 50 nodes in the core network.

A. Simulation of Scalability

In order to give an indication of the scalability of the KIS, a large number of randomly connected core networks have been constructed by computer simulation. The networks are generally characterized by the following properties.

- All interconnections are full duplex, i.e., if a connection exists from node $x$ to node $y$, then a similar connection from node $y$ to node $x$ is present.
- A number between 1 and 6 is randomly chosen for each node. This number defines the number of connections that is created to other core nodes in the network.
- The network connectivity is ensured as possible “single islands” are connected with full duplex connections, i.e., in case the randomly connected network turns out to be two or more separated networks, these are connected using an extra duplex connection.

As an example, in Fig. 3 are given two randomly connected networks generated by the model. Both networks comprise 10 core nodes, but the topologies are very different as (B) is considerably more densely connected than (A). This illustrates the variety in the simulated networks. It should be noted that all the links between adjacent nodes have equal weight, i.e., the length of a path equals the number of intermediate hops.

It is notable that the size of information that is extracted for each node can be larger than the number of neighboring core nodes. This extra space can be allocated to outputs to legacy networks, reserved for type of class (TOS) information and other purposes. The data field including all these properties and information is in the following denoted the forwarding information field (FIF) and it is thus obvious that this may differ from the number of connections to neighboring core nodes.

The keys were applied to the core nodes in each of the generated networks. For each node, the lowest possible key was used restricted by the following.

Fig. 3. Examples of randomly connected networks generated by the simulation model.
In Fig. 4, two different node distribution scenarios are given, which are used in the following simulations. The uniform distribution in the figure refers to a network where values up to 8 or 16 are required in the FIF and the nonuniform distribution refers to networks with a wide range of node sizes with FIF ranging from 4 to 256. These two node size distributions are chosen to give an indication of the robustness of the scheme against variations in the node sizes.

The simulation results are depicted in Fig. 5 that shows the required number of bytes in the label field as a function of the size of the network for the two previously described node distribution scenarios. The “average” are calculated as an average over the average label size for each network and the “maximum” values are calculated as an average over the largest label size for each network. It is seen that the node distribution scenarios exhibit almost identical results, especially when the network size increases. This is seen as an indication that the scheme is very robust against changes in the node size distribution and with a label size about 7 bytes it is possible to support networks comprising ~ 50 nodes even when the maximum values are chosen.

Until now the nodes have been assigned keys randomly. However, a simple optimization can be performed by prioritizing the nodes before applying the keys. The core nodes, which are used most intensively, are prioritized highest and thus assigned the lowest possible keys. This is implemented by initially resolving all shortest path through the network to determine which core nodes are most intensively used. The results using prioritized nodes are shown in Fig. 6, where the same simulation scenarios from Fig. 4 are used. It is seen that half a byte is gained for the uniform network and a few bits are gained for the nonuniform network.

1) The key should be larger than the size of the FIF.
2) The key should be unique and have no common divisors (larger than 1) with any previously assigned keys in the network.

The shortest paths connecting each pair of nodes were found by using Dijkstra’s algorithm [6] and the labels were computed. Both the average label size and the maximum label size for each network were stored and the result was found by averaging over all the generated networks.

B. Analysis of Very Large Networks

Even though focus is on networks with less than 50 core nodes, it is interesting to analyze the situation for very large networks. The increase in the label size is dependent on two factors. First, larger randomly connected networks are more likely to require longer routes, which affect the label size, as more information has to be encoded with the same label. Secondly, the available primes increase nonlinearly with the network size making the average valid key larger.
The first issue is analyzed by counting the total number of paths of different lengths in the simulated networks with the assumptions given previously. The results are shown in Fig. 7 for networks comprising 50, 100, 150, and 200 core nodes. As an example it is seen that in a network with 100 core nodes about 3% of the routes traverse eight core nodes.

The figure is used to indicate the impact of reducing the maximum allowed route length to, e.g., eight core nodes. For a very large network of 200 nodes, less than 2.5% of the routes are affected, as illustrated by the “area” below the graph for a route lengths of 9 to infinite. It should be stressed that these values are obtained for randomly connected networks and the results can thus be considered as based on conservative assumptions.

It is interesting to compare the topologies generated here to real topologies as seen in the Internet. It turns out that the topology of the Internet tends to be less random than the ones used as a basis for the simulations presented in this paper [11], [12]. Thus, the proposed limitation in route lengths to eight hops is not a severe restriction in real networks.

The few longer routes that might anyway exist can be dealt with by dividing the path into two or more shorter path, which is further discussed in the next section.

The second scalability issue arises because the average size of a key increases with an increase in the network size. That is, for a given network size, the average key is calculated as an accumulation of all assigned keys in the network divided by the network size. The size of a label for a given path will in average reflect the calculated average key times the length of the path. For a number of route lengths this estimation is depicted in Fig. 8, where the required label size is sketched as a function of the network size for different route lengths.

It is seen that a label supporting a route length of six core nodes in a network comprising 100 nodes will require about 6 bytes. The values corresponding to the circles denote the percentages of the routes that require to traverse 9, 10, or 11 core nodes. These values are derived from Fig. 7 as a summation of the percentages of routes longer than eight nodes as described previously. It is obvious that these values can be further reduced in case proper network design is implemented. The procedure for dealing with path length longer than eight nodes and the impact on the network performance is discussed in the following section. However, this indicates that in case the path length is restricted to eight transit core nodes up to 200 nodes can be supported with a label size of only 9 bytes in the header.

C. Scaling Impact on Network Performance

In case a label becomes too large to be contained in the header, the path through the network has to be divided in two or more shorter paths. This is illustrated in Fig. 9, where a path from A to B is resolved illustrated with the dotted line comprising six core nodes. If the label for this path becomes too large the path AB is divided in AC and CB comprising three and four core nodes, respectively. The new path is depicted with a solid line.

When the routing protocol in A determines a route that requires a too large label, it simply defines a shorter and valid path to a node C, where C is closer to the final destination B than A. At C, the route to B is found using normal routing protocols and the packet is labeled and forwarded to B.

The example illustrates that dividing a path in two smaller will induce extra processing at the intermediate edge node C, where the packet is handled as a network layer packet. Furthermore, the number of traversed core nodes might be a little
higher, thus, a minor increase in the total network traffic might be induced.

Compared to traditional label swapping, where 20 bits are used for the label [13] the required label size in the KIS may seem rather high. It is, however, relevant to illustrate how the label size affects the total performance. In Fig. 10 it is assumed that the MPLS packet is used to encapsulate transport layer TCP/IP packets, which all have overhead larger than or equal to 40 bytes [14]. It is thus obvious, with these assumptions, that the impact of 6 or 10 bytes in the label is rather limited. This impact is even lesser if IPv6 should be used in the future as 40 bytes is required only for the IP header [14].

Hence it is indicated that the scheme scales acceptably with an increased network size as it supports network sizes of up to 50 nodes with only a label of 7 bytes and up to 200 nodes with 9 bytes using a minor restriction, which gives a negligible increase in overall traffic and processing. These label sizes should be compared to the large TCP/IP overhead required if the network is transporting IP traffic.

IV. CONCLUSION

In this paper, a novel scheme for packet forwarding in optical packet switched networks is presented. The scheme is conceptually compliant with the MPLS networking concept as the forwarding and the routing operations are fully separated. Furthermore, the need for header modification in the core nodes is avoided, as a label is created at the edge of the network and at each core node the required information is extracted from the label by a simple expression. The scheme requires no distribution of forwarding information throughout the network and header modification is avoided, which is very advantageous in optical packet switches as optical header modification is complicated and require costly components.

The scalability of the scheme is evaluated showing an acceptable scalability for MPLS networks with up to 50 core nodes, as only 7 bytes are required for the label in the header, which is reduced half a byte by prioritizing the most used nodes. By setting a minor restriction on the route length to eight nodes in the core network the scheme can support up to 200 nodes with a label size about 9 bytes. If the required label size exceeds this limit the path through the network is divided in two smaller paths inducing only a negligible extra traffic. These requirements to the label size are somewhat higher than for traditional label swapping, however, the difference is very small compared to the overhead in, e.g., IP traffic that the network might be used to transport.

Hence, due to the significant advantages that are associated with the scheme, we believe it has a great potential for use in future optical networks.

**References**


Tina Fjelde received the M.Sc.E. and Ph.D. degrees from the Technical University of Denmark, Lyngby, in 1998 and 2002, respectively. The Ph.D. study was done at the COM Research Center in optical communication.

Her main field of interests are optical signal processing and traffic theory in packet-switched networks. She is now with Mintera Corporation, Lowell, MA.

Lars Dittmann was born in 1962. He received the M.Sc.E.E. and Ph.D. degrees from the Technical University of Denmark, Lyngby, in 1988 and 1994, respectively.

He is currently an Associate Professor at the Technical University of Denmark teaching and doing research in the area of network and network node design. He is heading the network competence area within the COM Research Center at the same university. He has been involved in a number of EU and national projects working on packet switching. He is Project Coordinator of the IST-DAVID project.