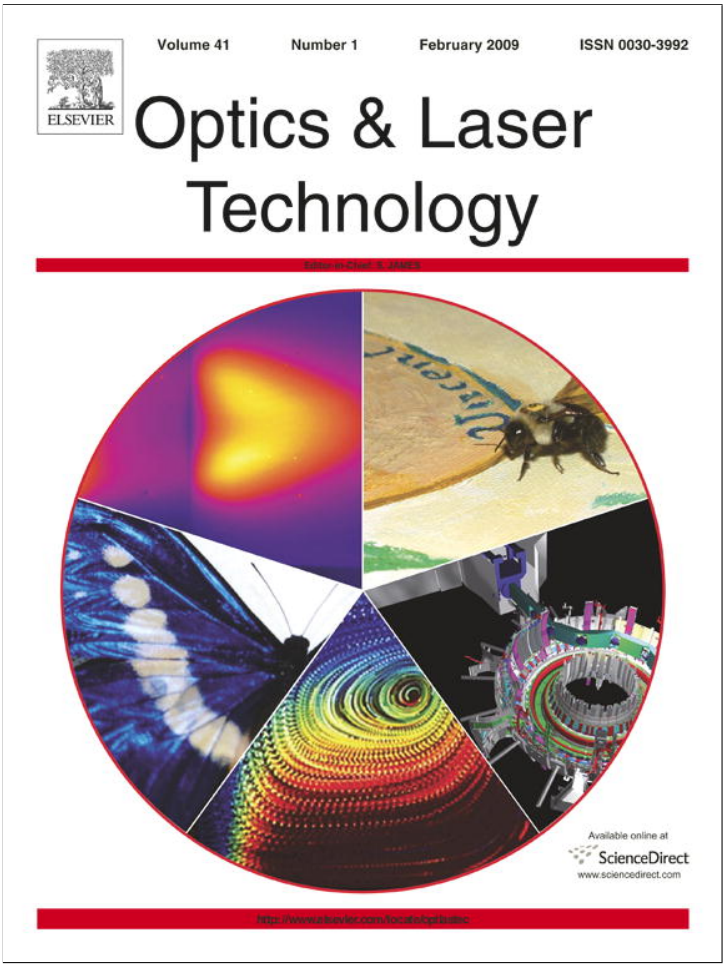

Optics & Laser Technology 41 (2009) 42–52

Contents lists available at ScienceDirect

Optics & Laser Technology

journal homepage: www.elsevier.com/locate/optlastec

A clustering-driven medium access control protocol for WDM star networks

Sophia G. Petridou, Panagiotis G. Sarigiannidis, Georgios I. Papadimitriou *, Andreas S. Pomportsis

Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 15 December 2007
Received in revised form
13 April 2008
Accepted 21 April 2008
Available online 5 June 2008

Keywords:
WDM star networks
Scheduling
Clustering

ABSTRACT

Channel assignment and nodes' service order are two key issues that have to be addressed when designing medium access control (MAC) protocols for WDM star networks. Traditional scheduling techniques consider either channel assignment or nodes' service order issues. Furthermore, they make use of information such as data channels or receivers' availability, without combining it with senders' demands. This paper introduces a novel approach to message scheduling algorithms for WDM star networks, which is driven by clustering techniques. The proposed clustering driven-minimum scheduling latency (CD-MSL) scheme combines all the aforementioned information to create groups of similar source nodes on the basis of the destination nodes of their messages, aiming at rearranging nodes' service order and improving network performance. Extensive simulation results are presented, which indicate that the proposed clustering-driven scheme leads to a significantly higher throughput-delay performance, in comparison to conventional scheduling algorithms.

1. Introduction

Nowadays, given the ever-growing demands for network capacity, it seems that optical networking using wavelength division multiplexing (WDM) offers an excellent way to exploit the huge bandwidth of optical fibers [1,2]. WDM star networks using single-hop architecture seem to dominate in both local and metropolitan area networks [3,4]. Using the star topology, a WDM network can be configured as a broadcast-and-select network in which all of the inputs from the various network nodes will be combined by a passive star coupler via two-way fibers [5–7]. An important issue in such WDM networks is to specify the way that the nodes transmit on the available channels [8,9]. Thus, a media access control (MAC) protocol is necessary in order to unleash the network's capabilities by introducing a scheduling algorithm which will allocate the network resources in an efficient way [10].

In practice, there are two constraints on scheduling a message in WDM star networks: the data channels' availability as well as the receivers' availability. Thus, channel assignment and nodes' service order are two key issues in designing MAC protocols for optical WDM star networks [1]. Up to now, popular scheduling techniques consider either channel assignment or nodes' service order issue but not both of them, and, thus, they suffer from low performance, especially when operating under heavy traffic.

1.1. Research review

A well-known, efficient scheduling algorithm for local area WDM networks with broadcast-and-select star architecture is the earliest available time scheduling (EATS) [11]. EATS addresses the channel assignment without, however, handling nodes' service order, since it considers source nodes in a sequential order, ignoring the fact that rearranging the nodes' service order may affect the network's performance. In practice, EATS always selects for transmitting the earliest available data channel independently of the destination's availability. Thus, if a message is destined for a busy node, its transmission time will be scheduled far later than the data channel's earliest available time, downgrading the channel utilization.

The receiver oriented-earliest available time scheduling (RO-EATS) [12] comes as an extension of the EATS which overcomes the aforementioned drawback by taking into account the receiver's availability. The RO-EATS's core idea is to prioritize messages that are destined for the least used receiver which means, that it rearranges nodes' service order, taking into account the destination nodes. However, it retains the EATS's logic for the channel assignment issue. As a result, RO-EATS is significantly improved in comparison to EATS in terms of mean packet delay without, however, providing remarkable improvements in terms of network throughput.

A different approach which largely advances both network throughput and mean packet delay is adopted by the minimum scheduling latency (MSL) [13]. MSL modifies the choice of the transmission data channel on the basis of the minimum scheduling latency, where the channel-scheduling latency is defined as

* Corresponding author.
E-mail addresses: spetrido@csd.auth.gr (S.G. Petridou), sarpan@csd.auth.gr (P.G. Sarigiannidis), gp@csd.auth.gr (G.I. Papadimitriou), apombo@csd.auth.gr (A.S. Pomportsis).

0030-3992/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.optlastec.2008.04.003

the difference between the time that a channel starts transmission and the time that this channel becomes available. Furthermore, it deals with the channel assignment issue taking into account not only the channel's availability, i.e. the earliest time that a channel will become available for transmitting data, but also the receiver's availability, i.e. the earliest time that a receiver will become available for receiving data. Even though MSL clearly outperforms EATS and RO-EATS, its performance is limited by the fact that it retains the EATS's logic for the nodes' service order. As it has already been mentioned, the sequential service order suffers from not taking into account the specific demands of the network's nodes.

1.2. Contribution

This paper introduces a novel algorithm that deals with both channel assignment and nodes' service order issues, based on the clustering [14,15] of the network's nodes. More specifically, the proposed algorithm handles the channel assignment issue according to the MSL logic, which takes into account both channels' and receivers' availability information. Moreover, it addresses the nodes' service order in an innovative clustering-driven way, which combines information considering both the source and destination nodes.

More specifically, the proposed clustering driven-minimum scheduling latency (CD-MSL) protocol is inspired by the fact that the nodes' service order should be addressed in an effective way, which would take into account both source and destination nodes without imposing complexity to the scheduling algorithm. Thus, the new scheme is driven by the clustering techniques which, in general, aim at creating groups of objects (i.e. clusters) on the basis that objects assigned to the same cluster are "similar" to each other and "dissimilar" to the nodes belonging to other clusters [14]. In practice, the CD-MSL organizes the network's nodes into a table structure whose rows represent the source nodes while its columns correspond to the destination nodes. Each cell of this table indicates the length of message from each source node to each destination node. Then, based on this table, the CD-MSL groups the network's nodes into clusters according to the destination of their messages. In our framework, CD-MSL groups together nodes with common message destination. Thus, given that each cluster will consist of nodes with probably common destination, CD-MSL defines the nodes' service order, choosing for transmission nodes belonging to different clusters.

Rearranging the nodes' service order in this way, CD-MSL manages to decrease the probability of scheduling messages to the same destination at successive order which downgrades the channel utilization. As a result, the schedule length is reduced and the network performance is upgraded. At the same time, clustering runs in time linear with the number on network's nodes without aggravating scheduling algorithm's complexity.

The remainder of this paper is organized as follows. Section 2 provides the network configuration while clustering preliminaries are given in Section 3. The proposed scheduling algorithm is described in Section 4. Section 5 provides details about the traffic model we use, while Section 6 discusses the simulation results. Conclusions and future work insights are given in Section 7.

2. Network configuration

Let us consider a single-hop, broadcast-and-select WDM star network containing n nodes. Each node is connected to a passive star coupler via a pair of optical fibers, one of which is used for transmission and the other for reception [16–19]. Given that each of the n nodes can either transmit or receive a message, we use the sets $S=\{s_1,\ldots,s_n\}$ and $D=\{d_1,\ldots,d_n\}$ to denote the role of a node as a source or a destination, respectively. The system supports $w+1$ channels (wavelengths), where $\Lambda=\{\lambda_1,\ldots,\lambda_w\}$ is the set of the data channels while one channel λ_0 is dedicated for pre-transmission coordination (i.e. control channel). Thus, according to Table 1, the network consists of n nodes and $w+1$ channels, where $n>w$. Transmissions from all nodes to all channels are combined at the passive star coupler and broadcast to all nodes via receiver fibers [8]. More specifically, each node is equipped with two transmitters and two receivers. The first transmitter and receiver are fixed and tuned to λ_0 (FT-FR), for transmitting or receiving the control packets, while the second transmitter and receiver are tunable (TT-TR) for accessing the data channels. In such CC-FTTT-FRTR implementation, which is depicted in Fig. 1, each node maintains w queues i.e. one for each data channel [20] and two nodes s_i and d_j, $i,j=1,\ldots,n$ and $i\neq j$, are able to communicate when the transmitter of s_i and the receiver of d_j are tuned to the same wavelength.

Table 1
Network symbols' notation

Symbol	Description
n, w	Number of nodes and data channels
$S=\{s_1,\ldots,s_n\}$	The set of source nodes
$D=\{d_1,\ldots,d_n\}$	The set of destination nodes
$\Lambda=\{\lambda_1,\ldots,\lambda_w\}$	The set of data channels
λ_0	The control channel
M	The $n\times n$ message table
k	The upper bound of messages' length
t	Schedule's length in timeslots
$L=\{l_1,\ldots,l_t\}$	The set of timeslots
H	The $w\times t$ scheduling matrix

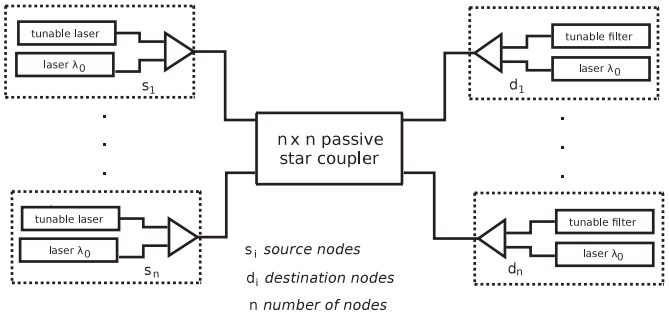

Fig. 1. The network architecture.

In the above CC-FTTT-FRTR system time is slotted and the transmission is synchronous. In particular, there are two different phases, namely the control and data phase [6], as illustrated in Fig. 2. Control and data phases may overlap each other during the transmission frame. Regarding the control phase, the control channel λ_0 is shared using the TDMA technique to avoid collisions of control packets [11]. In channel λ_0, each frame consists of n timeslots where each timeslot is dedicated to one source node i.e. the node s_i, $i=1,\ldots,n$. For example, the node s_i uses the i-th timeslot during the control phase to transmit its control packet. During the data phase, the real message transmission takes place. The nodes are assumed to generate messages of variable lengths which can be divided into several equal-sized packets. Each packet is transmitted in time equal to a timeslot. Thus, the data channels are also time-slotted, but the length of a timeslot is independent of the length of control frame [11]. We define $L=\{l_1,\ldots,l_t\}$ to be the set of the t timeslots which specify the

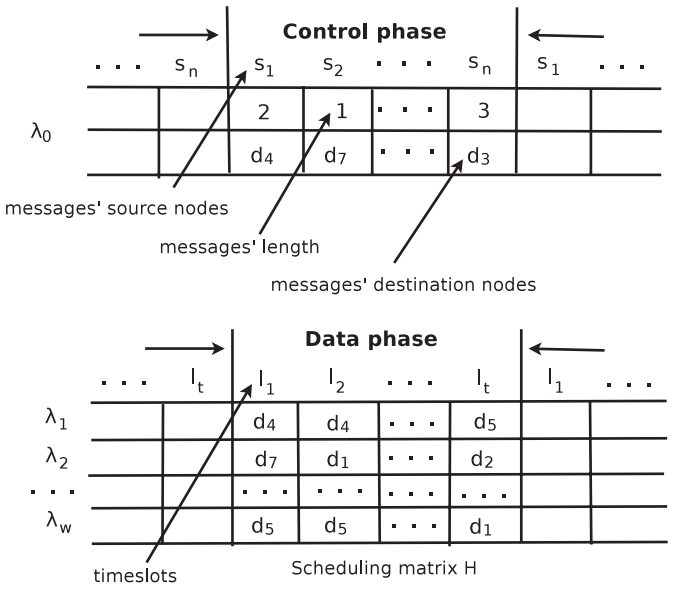

Fig. 2. Control and data phases of each frame.

schedule length of the data phase. All nodes in the network are assumed to be synchronized by using a common clock [21]. The synchronization problem has been studied in [22,23].

Example 1. According to Fig. 2, the source node identified as s_1 requests during the 1st slot of the control phase (λ_0) the transmission of a message whose length is 2 to the destination node d_4, while, node s_2 requests the transmission of a message of length 1 to node d_7 during the 2nd slot. As a result, during the data phase, we observe two packets to be scheduled on data channel λ_1 occupying the timeslots l_1, l_2 and being destined to node d_4, while, the one packet destined for node d_7 is scheduled on channelλ_2, during the timeslot l_1.

In such a network, given the tunability of the transmitters, it is obvious that two or more source nodes might cause channel collision, transmitting messages on the same data channel simultaneously. Furthermore, receiver collisions might occur when two or more nodes transmit messages to the same node simultaneously, since each node is equipped with a single tunable receiver [7]. Thus, the MAC protocol has to coordinate the nodes' data transmission and prevent collisions. In our framework, the MAC protocol handles the above issues running a scheduling algorithm at the end of the control phase in each frame. This scheduling algorithm is distributed and runs simultaneously at each node using two tables, namely the receiver available time (RAT) and the channel available time (CAT) tables [11–13].

The RAT table contains n elements, where $\mathrm{RAT}(d_i) = x$, $i = 1,\ldots,n$, indicates that destination node d_i will be available after x timeslots. If $\mathrm{RAT}(d_i) = 0$, then node d_i is currently idle and no reception is scheduled for it. The CAT table consists of w elements, where $\mathrm{CAT}(i) = y$, $i = 1,\ldots,w$, implies that channel i will be available after y timeslots. If $\mathrm{CAT}(i) = 0$, then data channel i is currently available. Tables RAT and CAT are used to avoid receiver and channel collisions respectively. Based on these tables, the scheduling algorithm produces a $w \times t$ scheduling matrix H, where t denotes the length of the schedule in timeslots. Each element $h(i,j)$, $i = 1,\ldots,w$ and $j = 1,\ldots,t$, represents the destination node that receives a message on channel λ_i, during the timeslot l_j. Given that the scheduling algorithm is distributed, it holds that all nodes build the same scheduling matrix H before the data transmission. Fig. 2 provides an example of such a scheduling matrix.

3. Clustering preliminaries

Informally, clustering is defined as the problem of partitioning data objects into groups (i.e. clusters), such that objects in the same group are "similar", while objects in different groups are "dissimilar" [24]. In our framework, the clustering process aims at partitioning the source nodes of set S. The source nodes are considered to be similar and, thus, they are grouped together if they present common message destination. Therefore, we organize the sets S and D into an $n \times n$ message table M, whose $m(i,j)$ element, $i,j = 1,\ldots,n$ and $i \neq j$, indicates the length of the message from the source node s_i to the destination node d_j. Given that each node s_i can transmit a message per frame, it is obvious that the i-th row of the M table will have one nonzero value. On the other hand, the j-th column of the M table can have more than one nonzero values indicating that each d_j node can receive more than one messages during a frame. Under this notation, each node s_i is considered to be a multivariate vector consisting of n values. We call this vector as demand pattern and we define it as follows:

$$M(i,:) = (m(i,1),\ldots,m(i,n))$$

According to Table 2, a clustering Cl of S is a partition of S into noc disjoints sets i.e. clusters $C_1,\ldots,C_{\mathrm{noc}}$, that is, $\bigcup_{i=1}^{\mathrm{noc}} C_i = S$ and $C_i \cap C_j = \emptyset$ for all $i \neq j$. The noc clusters $C_1,\ldots,C_{\mathrm{noc}}$ consist of $|C_1|,\ldots,|C_{\mathrm{noc}}|$ members (i.e. source nodes), respectively. Nodes assigned to the same cluster are "similar" to each other and "dissimilar" to the nodes belonging to other clusters in terms of the destination of their messages.

The clustering definition assumes that there is a quality measure that captures intra-cluster similarity and/or inter-cluster dissimilarity, and then clustering becomes the problem of grouping together data objects so that the quality measure is optimized. A common approach is to evaluate the dissimilarity between two objects (in our case the source nodes) by using a distance measure [14]. In our case, we proceed to the clustering of S using the Squared Euclidean distance[1]which is a well-known and widely used distance measure in the vector-space model [14,15]. Therefore, the dissimilarity between two source nodes e.g. $s_x, s_y \in S$ can be evaluated by the distance of their vectors. Thus, we use the expression $d_{\mathrm{E}}(s_x, s_y)$ to denote the Squared Euclidean distance of the nodes' vectors $M(x,:)$ and $M(y,:)$:

$$d_{\mathrm{E}}(s_x, s_y) = \sum_{i=1}^{n} (M(x,i) - M(y,i))^2$$

Once the clusters are obtained, we consider an arbitrary cluster $C_j, j = 1,\ldots,\mathrm{noc}$, of the set S. The representation of cluster C_j when clustering process Cl is applied to it, collapses the nodes belonging to C_j into a single point (e.g. the mean value which does not correspond to an existing node). This point is called cluster's representative c_j (also known as centroid) since each node $s_i \in C_j$ is represented by c_j. Given the vectors of $s_i \in C_j$, the vector of c_j is defined as follows:

$$\mathrm{Means}(j,:) = \frac{1}{|C_j|} \sum_{s_i \in C_j} M(i,:), \quad j = 1,\ldots,\mathrm{noc}$$

Since both $M(i,:)$ and $\mathrm{Means}(j,:)$ are vectors, their dissimilarity is measured by their Squared Euclidean distance $d_{\mathrm{E}}(s_i, c_j)$.

[1] The Squared Euclidean distance uses the same equation as the Euclidean distance, but does not take the square root. For two points $X = (x_1,\ldots,x_n)$ and $Y = (y_1,\ldots,y_n)$ in n-space their Squared Euclidean distance is defined as

$$\sum_{i=1}^{n} (x_i - y_i)^2$$

Table 2
Clustering symbols' notation

Symbol	Description
Cl	Clustering process
noc	Number of clusters
C_j	Cluster, $j = 1, \ldots, \text{noc}$
c_j	Cluster representative
Means	The noc $\times$ n clusters representatives' message table
d_E	Nodes distance over their messages' destination
J	Objective function

Considering all clusters, the clustering process is guided by the *objective function J* which is defined to be the sum of distances between each source node and the representative of the cluster that the node is assigned to:

$$J = \sum_{j=1}^{\text{noc}} \sum_{s_i \in C_j} d_E(s_i, c_j)$$

Based on the above we can define the network nodes clustering as follows: Given a network with a set S of n source nodes whose messages to n destination nodes (set D) are organized in an $n \times n$ message table M, the integers noc and k, and the objective function J, find a Cl clustering of S into noc clusters such that the J is minimized.

For the purpose of our clustering, we employed the well-known and widely used K-means partitional clustering algorithm [25]. K-means classifies a given data set to a certain number of clusters e.g. noc fixed a priori. Although K-means does not provide approximation guarantees it is widely used because it is efficient and it works very well in practice. K-means algorithm can be briefly described as follows: given n points to be clustered, a distance measure d_E to capture their dissimilarity and the number of clusters noc to be created, the algorithm initially selects noc random points as clusters' centers and assigns the rest of the n—noc points to the closest cluster center (according to d_E). Then, within each of these noc clusters the cluster representative (also known as centroid or mean) is computed and the process continues iteratively with these representatives as the new clusters' centers, until convergence.

4. The proposed clustering-driven scheduling

The proposed CD-MSL scheme is a two-step approach which firstly handles the nodes' service order driven by the clustering process and then, it deals with the channel assignment issue following the MSL spirit, as depicted in Fig. 3. The core idea is that both the source and destination nodes should be taken into account in determining the nodes' service order. Thus, the proposed algorithm groups together source nodes with the same message destination. The goal is that messages destined for the same destination should not be scheduled in a successive order. Therefore, CD-MSL schedules in sequence messages from nodes belonging to different clusters. Furthermore, CD-MSL prioritizes clusters as well as the members of each cluster according to the length of their messages.

More specifically, during the first step, i.e. the clustering step, we employ the K-means in order to produce the Cl clustering of S. Then, given the Cl and the $n \times n$ message table M, we sort the members of each cluster according to the length of their messages and the result is recorded on Sorted$M$. In practice, given that each node s_i, where $i = 1, \ldots, n$, transmits a message per frame, sorting these clusters' members gives priority to the nodes with long-length messages. Similarly, using the Means table, consisting of the clusters representatives' vectors Means$(j,:)$, we compute the Sorted$C$ which contains the sorted clusters. In this case, given that Means$(j,:)$ vectors may contain more than one nonzero values, we sort them according to their length i.e. |Means$(j,:)$|. A vector's length is more indicative than the sum of its values for revealing the information that CD-MSL needs, i.e. the clusters with heavy load. For example, given the vectors $(2,2,2)$ and $(6,0,0)$, their elements sum is 6, while their lengths are $\sqrt{12}$ and 6, respectively, which means that the second vector will have priority over the first one in the service order. The calculated Sorted$M$ and Sorted$C$ are then used in order to define the nodes' service order (NSO).

Once the NSO is formed, the algorithm proceeds to the second step, called the channel assignment step. The goal of the function *ChannelAssignment* is to address the channel assignment issue, providing collision-free communication. Thus, it builds the scheduling matrix H, based on the MSL algorithm's logic, which considers both RAT (i.e. receivers' availability) and CAT table (i.e. channels' availability).

Algorithm 1 The CD-MSL flow control

Input:	A set S of n nodes organized in an $n \times n$ message table M, the upper bound on nodes' requests k and the number of clusters noc.
Output:	The scheduling matrix H.
	/ * *Clustering Step* * /
1:	(Cl, Means) $= K-$means$(M,$ noc$)$
2:	Sorted$M$ $=$ *Quicksort*$(M,$ Cl$)$
3:	Sorted$C$ $=$ *Quicksort*(Means)
4:	NSO $=$ *ServiceOrder*(Sorted$M$, Sorted$C$)
	/ * *Channel Assignment Step* * /
5:	$H =$ *ChannelAssignment* (NSO)

Theorem 1. *The* CD-MSL *has time complexity* $O(nw^2)$.

Proof. During the clustering step we employ the K-means algorithm (Algorithm 1, line 1) whose time complexity is $O(n \text{ noc } r)$, where n is the number of nodes, noc the number of clusters to be created and r the number of iterations that takes the algorithm to converge. However, both noc and r are relatively small compared to the number of nodes n and thus their contribution to the algorithm's complexity can be ignored [14]. Thus, the Cl clustering is computed in time linear on the number of nodes: $O(n)$. The *Quicksort* functions (lines 2 and 3) sorts the nodes and clusters' representatives in $O(n \log n + \text{noc} \log(\text{noc}))$ time. The *ServiceOrder* function (line 5) takes time $O(n \text{ noc})$ to arrange the messages from the n nodes according to Sorted$M$ and Sorted$C$. The total time complexity of the clustering step is thus $O(n + n \log n + \text{noc} \log(\text{noc}) + n \text{ noc})$ which becomes $O(n \log n)$ since noc is relatively small compared to the number of nodes n. During the second step, the *ChannelAssignment* function (line 5) needs $O(nw^2)$ time [13] to form the scheduling matrix H, where w is the number of channels. Given the $O(n \log n)$ complexity of the clustering step, the $O(nw^2)$ complexity of the channel assignment step and the fact that $\log n$ is significantly smaller compared to w^2, it holds that the total complexity of CD-MSL is $O(nw^2)$. □

Thus, we can claim that the proposed scheme succeeds in combining information from different network sources i.e. the channels and receivers' availability, as well as the specific demands of the network's nodes, without aggravating the scheduling algorithm, since the complexity of CD-MSL is equal to the one of MSL, i.e. $O(nw^2)$.

Two new protocols using clustering techniques have been recently introduced in [18,19]. However, these protocols cannot be fairly compared to the proposed CD-MSL, because the latter uses a

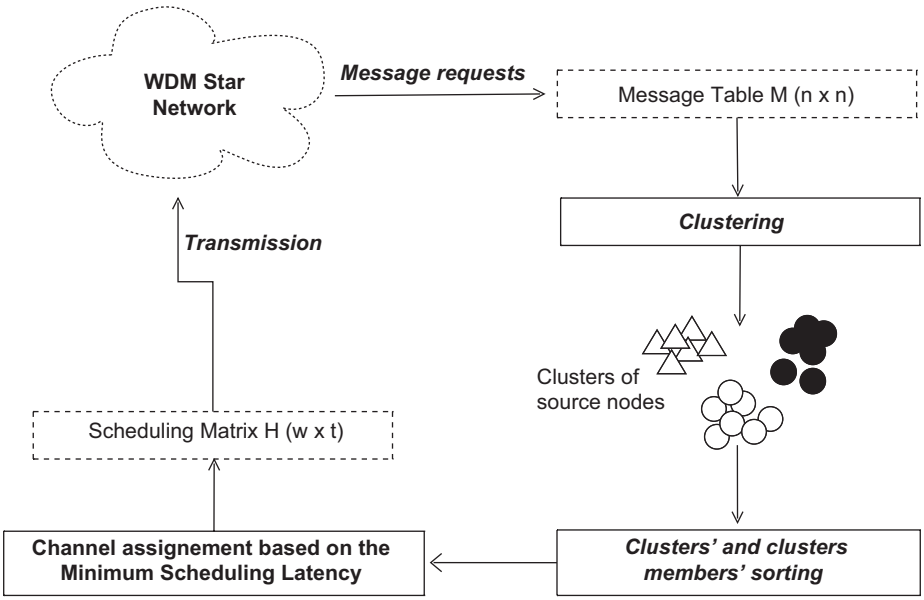

Fig. 3. The CD-MSL overview.

Table 3
The table *Members* before members' sorting

C_1	s_3	s_5	s_7	s_8
C_2	s_1	s_4		
C_3	s_2	s_6		

Table 4
The table *Members* after members' sorting

C_1	s_8	s_5	s_3	s_7
C_2	s_4	s_1		
C_3	s_2	s_6		

completely different hardware configuration in relation to CBSA [19] and has a higher computational complexity in relation to CO-EATS [18].

4.1. A numerical example

To facilitate the comprehension of the proposed scheme, let us consider a WDM star network consisting of the source nodes $(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8)$, the data channels $(\lambda_1, \lambda_2, \lambda_3)$ and having the upper bound of nodes' messages length $k = 10$ packets. Then, a 8×8 message table M could be the following:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Example 2. In the above message table M the fact that $M(8,2) = 5$ means that the source node s_8 sends a message of length 5 to the destination node d_2 while nodes d_4 and d_7 will receive two messages.

Applying the K-means algorithm for $\text{noc} = 3$ in the above message table M results in $\text{Cl} = (2,3,1,2,1,3,1,1)$ which can be represented by Table 3 i.e. the table *Members* before member's sorting. From Table 3, it holds that the nodes $s_3, s_5, s_7, s_8 \in C_1$, the nodes $s_1, s_4 \in C_2$, while the nodes $s_2, s_6 \in C_3$. As discussed in Section 3, Cl places in the same cluster, source nodes which are similar in terms of their destination nodes, e.g. the nodes s_1 and s_4 destine their messages for node d_7, the nodes s_2 and s_6 for node d_4, while the nodes s_3 and s_8 for node d_2. The rest nodes s_5 and s_7 are forced to be placed in the same cluster (i.e. C_1). Then, sorting the members on each cluster according to the length of their message results in swapping the nodes of C_1 and C_2. Therefore, Table 3 is updated as Table 4.

Given the above Cl clustering that K-means algorithm produces, the clusters representatives' message table Means is formed according to the vectors of clusters' members:

$$\text{Means} = \begin{pmatrix} 1 & 1.75 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.5 & 0 \\ 0 & 3.5 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Sorting Means provides our algorithm with the following service order: C_2, C_3, C_1, since $|\text{Means}(1,:)| = 2.1$, $|\text{Means}(2,:)| = 3.5$ and $|\text{Means}(3,:)| = 3.5$. At this point, given that each cluster consists of nodes with probably the same destination, our scheme should separate them taking into account the result of the Means sorting. Therefore, the nodes' service order is defined as $s_4, s_2, s_8, s_1, s_6, s_5, s_3, s_7$ instead of the sequential one $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$.

Table 5 depicts the scheduling matrix H produced by the proposed CD-MSL algorithm when the transmitters/receivers tuning time is set to 1 and the propagation delay of messages is set to 2. On the other hand, Tables 6–8 represent the scheduling matrix H in case that the EATS, RO-EATS and MSL algorithms are employed, respectively. Based on these tables, the channel utilization providing by CD-MSL is 90% which is significantly improved in comparison to EATS whose channel utilization is 69% as well as with that of RO-EATS and MSL which both provide 75% utilization. In terms of the mean packet delay, the observed values are 4.2 (CD-MSL), 4.9 (EATS), 4.4 (RO-EATS) and 4.4 (MSL) timeslots. Thus, the proposed CD-MSL scheme clearly outperforms the EATS, RO-EATS and MSL approaches, since the schedule length produced by CD-MSL is clearly reduced compared to that of the other approaches.

Table 5
The scheduling matrix H produced by CD-MSL

	Timeslots									
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
λ_1	d_7	d_7	d_7	d_7	d_1	d_1	d_1	d_1	d_2	d_2
λ_2	d_4	d_4	d_4	d_4	d_4	d_7	d_7	d_7	d_6	d_6
λ_3	d_2	d_2	d_2	d_2	d_2		d_4	d_4		

Table 6
The scheduling matrix H produced by EATS

	Timeslots												
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}	l_{13}
λ_1	d_7	d_7	d_7	d_1	d_1	d_1	d_1	d_6	d_6				
λ_2	d_4	d_4	d_4	d_4	d_4		d_4	d_4	d_2	d_2	d_2	d_2	d_2
λ_3	d_2	d_2			d_7	d_7	d_7	d_7					

Table 7
The scheduling matrix H produced by RO-EATS

	Timeslots											
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}
λ_1	d_1	d_1	d_1	d_1	d_7	d_7	d_7		d_7	d_7	d_7	d_7
λ_2	d_2	d_2	d_6	d_6	d_2	d_2	d_2	d_2	d_2			
λ_3	d_4	d_4	d_4	d_4	d_4		d_4	d_4				

Table 8
The scheduling matrix H produced by MSL

	Timeslots											
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}
λ_1	d_7	d_7	d_7		d_7	d_7	d_7	d_7				
λ_2	d_4	d_4	d_4	d_4	d_4	d_6	d_6	d_2	d_2	d_2	d_2	d_2
λ_3	d_2	d_2	d_1	d_1	d_1	d_1	d_4	d_4				

The CD-MSL scheme is compared to EATS, RO-EATS and MSL protocols in terms of network throughput and mean packet delay. It does not addresses the fairness issue, since none of these protocols does so. However, CD-MSL is considered to be fair, since it does not favour any node among the others, on the basis that each node produces both short- and long-length messages. The CD-MSL protocol could face fairness problems only in the extreme case, where one node produces only short-length messages.

The proposed scheme considers that all packets are of equal priority. However, since real-time traffic represents 25–30% of the Internet traffic, the proposed CD-MSL can be easily extended to handle real-time traffic, such as audio and video. In the literature, real-time traffic is treated as high-priority packets, while nonreal-time traffic is treated as low-priority packets [13]. Thus, during each frame the nodes include in their control packets the priority information of their data packets i.e. $p_r = 1$ for high-priority packets and $p_r = 0$ for low-priority packets. Then, two separate clustering processes take place. One for the nodes with high-priority packets and a second for the nodes with low-priority ones. However, since each node handles both high- and low-priority packets, it holds that it may be assigned to clusters obtained from both processes. Thus, the proposed scheme succeeds in obtaining significant improvements for real-time traffic, without sacrificing the performance on nonreal-time traffic, such as text, e-mail or file transfer.

5. Traffic modeling

Two distinct traffic models are employed for the purpose of our experimentation. According to the first model, namely model A or uniform model, it is assumed that the packet arrival process on each of the w queues follows Uniform distribution. In practice, the source nodes s_i, where $i = 1, \ldots, n$, may send messages of 0 to k both included on each frame with equal probability. Moreover, the traffic pattern is uniform i.e. a message is destined to every other node d_j, where $j = 1, \ldots, n$, with equal probability.

According to the second model, i.e. model B or poisson model, the packet arrival process is assumed to follow Poisson distribution. In general, the Poisson distribution of the number of packets arriving at a specific queue per frame is defined as

$$p(X;\theta) = \frac{e^{-\theta}\theta^X}{X!} \quad (1)$$

where $p(X;\theta)$ is the probability of X packets being assigned to a specific queue during a specific frame whereas θ is the expected number of packets being assigned to this queue during this frame.

Based on the above, we proceed to the nodes load patterns' generation defining three classes of nodes, in order to simulate a more realistic environment. More specifically, each node is assigned to a class with equal probability and characterized as light, medium or heavy according to its traffic load. The values of θ for these three classes are defined as $k/4$, $k/2$ and $3k/4$, respectively [26], where k expresses the upper bound of message length requested per node per frame.

Even though the traffic streams in real world are often characterized as bursty, we do not experimented with bursty traffic, since the variable-length messages can be actually considered as bursts. Given that each message consists of consecutive arriving fixed size packets having the same destination node and it is scheduled as a whole which is transmitted continuously, it can be safely assumed that both models A and B approximate bursty network traffic [27].

6. Experimentation

To evaluate the proposed algorithm we carried out experimentation, where we compare CD-MSL with the well-known EATS, RO-EATS and MSL protocols. We experimented with different network parameters, including different number of nodes n, channels w and clusters noc, as well as, different traffic load k, where k expresses the upper bound of message length requested per node per frame. The performance of the compared algorithms is evaluated in terms of network throughput and mean packet delay.

Consider that Γ denotes the network throughput, while r represents the line transmission rate per channel in Gbps. Then, given that t represents the schedule's length, w denotes the number of channels and M is the message table of dimension $n \times n$, the network throughput is defined as follows:

$$\Gamma = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} m(i,j)}{w * t} * r \quad (2)$$

On the other hand, the mean packet delay is defined as the average time the packets spend in the system, waiting to be transmitted and it is composed of packet transmission time, queuing delay, tuning transmission delay and propagation delay in timeslots.

The simulation results are produced according to the following assumptions:

(1) The transmitters/receivers tuning time is set to 1 and the propagation delay of messages is set to 2.

(2) The line transmission rate per channel is set to 3 Gbps.
(3) The outcome results from 10 000 transmission frames.

6.1. Throughput versus number of nodes

Fig. 4(a) and (b) depict the network's throughput as a function of the number of nodes for $n = 20, 30, \ldots, 100$, while the traffic load is set to $k = 30$ according to model A and B, respectively. The number of channels is set to $w = 20$, while we also fixed the number of clusters at noc $= 20$. Defining noc $= w$, the nodes' service order is formed by selecting source nodes belonging to different clusters. In this way, consecutive messages are not scheduled to the same destination, since messages from different clusters probably have different destinations. The throughput improvement in case of the CD-MSL algorithm is due to the fact that the length of the scheduling matrix is reduced.

It is apparent that for any number of nodes n, the proposed CD-MSL scheme provides steadily higher throughput compared to EATS, RO-EATS and MSL under both uniform and poisson traffic. Indicatively, in Fig. 4(a), we observe that, under uniform traffic, for $n = 100$ the network throughput provided by CD-MSL is improved as much as 17.6% over EATS, 19.6% over RO-EATS and 11.5% over MSL. In case that model B is employed, Fig. 4(b) shows that CD-MSL presents similar improvements over EATS, RO-EATS and MSL, which, for $n = 100$ are 16.9%, 18.9% and 10.6%, respectively.

It has to be noticed, that the minimum observed improvements are for $n = 20$ nodes, where the contribution of the clustering is of low value, since each node constitutes a cluster. In this case, nodes with similar destination will be assigned to different clusters and, as a consequence, they may be scheduled at successive order downgrading the network's throughput. On the other hand, remarkable improvements are observed for $n > 40$ nodes, since the ratio between noc and n significantly contributes to the performance of CD-MSL. This can be explained by the fact that, on the average, each cluster consists of two or more members i.e. the clustering captures nodes with similar destination and, thus, prioritizing them according to the Algorithm 1, we manage to provide a short length scheduling matrix.

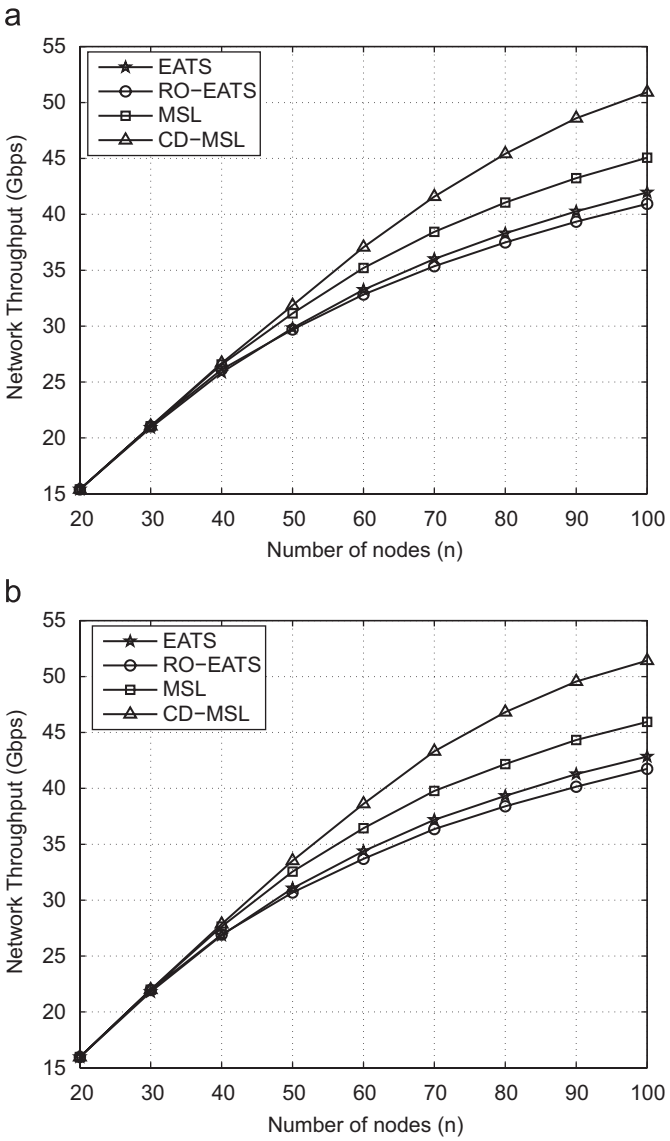

Fig. 4. Network throughput as a function of the number of nodes for $w = 20$ channels, traffic load $k = 30$ and noc $= 20$ clusters: (a) uniform traffic and (b) Poisson traffic.

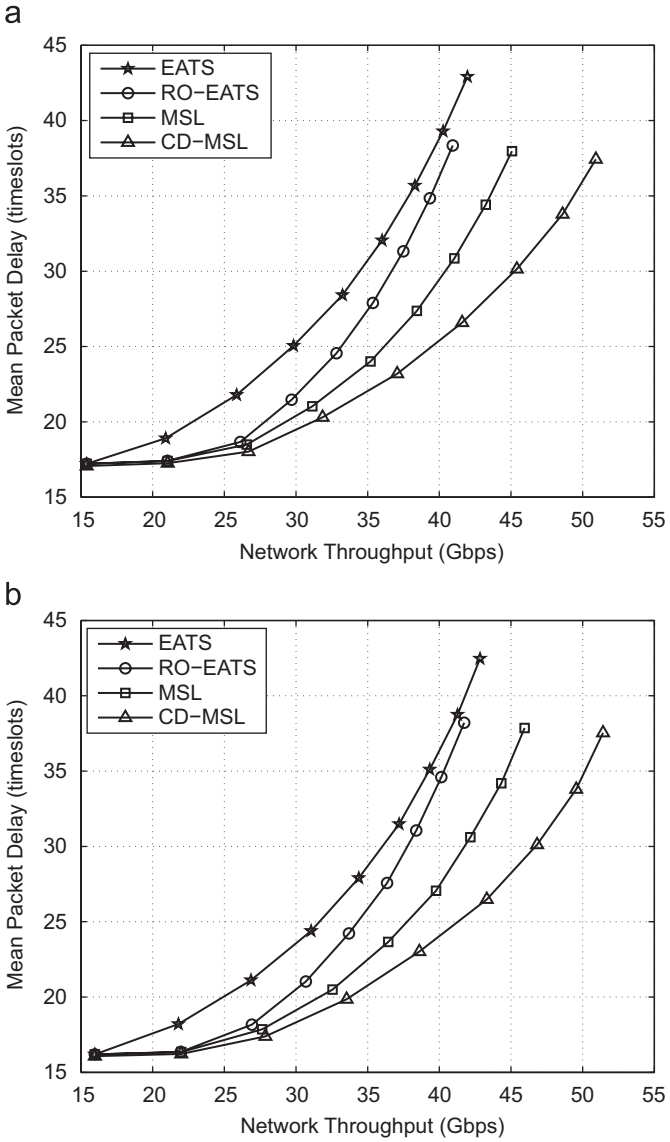

Fig. 5. Mean packet delay as a function of the network throughput for $w = 20$ channels, traffic load $k = 30$ and noc $= 20$ clusters: (a) uniform traffic and (b) Poisson traffic.

6.2. Throughput versus mean packet delay

For this part of experimentation, we evaluate the mean packet delay as a function of the network throughput. We keep the traffic load at $k = 30$ and define the same number of clusters and channels i.e. noc $= w = 20$, as previously. The number of nodes is set to $n = 20, 30, \ldots, 100$.

According to Fig. 5(a) and (b), it is apparent that the improvement in network's throughput does not affect the mean packet delay. We observe that CD-MSL keeps the mean packet delay lower in comparison to the other protocols, independently of the number of nodes, while it obtains a higher throughput. Indicatively, for $n = 100$, under uniform traffic, CD-MSL achieves a throughput of 50.9 Gbps and a mean packet delay of 37.4 timeslots, while the respective values for EATS are 41.9 Gbps and 42.9 timeslots, for RO-EATS 40.9 Gbps and 38.4 timeslots and for MSL 45.1 Gbps and 37.9 timeslots. The proposed CD-MSL scheme exhibits similar performance under poisson traffic. For example, for $n = 100$, CD-MSL obtains 51.4 Gbps throughput while the corresponding values of EATS, RO-EATS and MSL are 42.8, 41.7 and 45.9 Gbps, respectively. In terms of mean packet delay the values are 37.5 timeslots for CD-MSL and 42.5, 38.2 and 37.9 timeslots for EATS, RO-EATS and MSL, respectively.

The accuracy of the above simulation results can be verified by their confidence intervals. For each point of the curves, the simulation time was 10 000 frames during which several millions of packets were generated. The 95% confidence intervals of mean packet delay in case of CD-MSL under Uniform and Poisson traffic are shown in Table 9.

At this point, we can claim that, independent of the traffic model and the number of nodes the proposed clustering-driven approach clearly outperforms classic scheduling schemes. This is due to the construction of shorter schedules, which leads to significant throughput-delay improvements.

6.3. Throughput versus load

In Fig. 6(a) and (b) the network throughput is presented as a function of the traffic load. As it has already been discussed, the traffic load is expressed by the parameter k, which indicates the upper bound of message length in packets (or equivalently in timeslots) requested per node per frame. In this group of simulation, we experimented with $k = 10, 15, \ldots, 30$, while we fixed the number of nodes at $n = 70$ and the number of channels and clusters at noc $= w = 20$.

Table 9
Confidence intervals of mean packet delay

Network throughput (Gbps)	Confidence interval
Uniform traffic	
15.39	17.07 ± 0.29
21.05	17.25 ± 0.29
26.69	18.02 ± 0.29
31.85	20.30 ± 0.29
37.06	23.19 ± 0.30
41.59	26.59 ± 0.33
45.41	30.14 ± 0.36
48.60	33.78 ± 0.40
50.92	37.43 ± 0.44
Poisson traffic	
15.97	16.08 ± 0.27
21.99	16.22 ± 0.27
27.86	17.39 ± 0.27
33.53	19.85 ± 0.27
38.61	23.02 ± 0.29
43.31	26.48 ± 0.32
46.82	30.11 ± 0.36
49.55	33.79 ± 0.39
51.42	37.54 ± 0.44

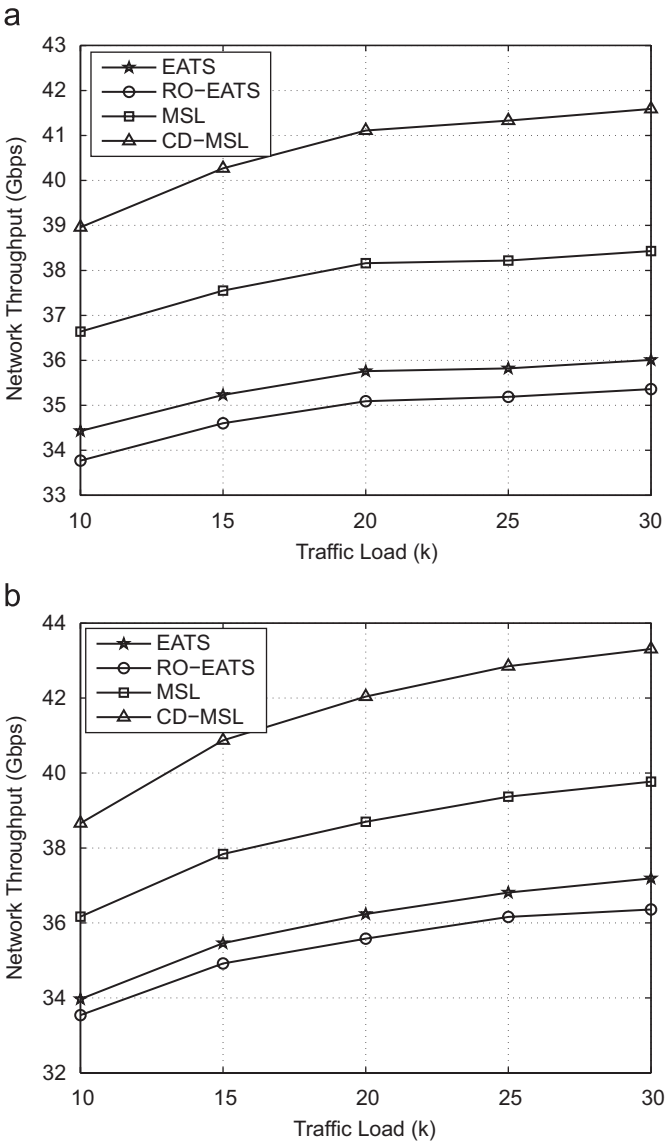

Fig. 6. Network throughput as a function of the traffic load for $n = 70$ nodes, $w = 20$ channels and noc $= 20$ clusters: (a) uniform traffic and (b) Poisson traffic.

The proposed CD-MSL scheme is presented to be significantly superior in comparison to EATS, RO-EATS and MSL, under both uniform and poisson traffic. It is important to notice that its behavior is improved as the network load is increasing. This is due to the fact that, as the k increases, the traffic becomes more asymmetric and, thus, the clustering process obtains well separated clusters. The better the quality of clusters, the higher the performance advantage of CD-MSL over the classic scheduling schemes.

For example, in Fig. 6(a), for $k = 30$, CD-MSL achieves a network throughput of 41.6 Gbps, while EATS, RO-EATS and MSL offer 36.0, 35.4 and 38.6 Gbps, respectively. A similar performance advantage can been seen in Fig. 6(b) for poisson traffic, where the corresponding values are 43.3, 37.2, 36.4 and 39.8 Gbps, respectively.

6.4. Throughput versus channels

Given that the above plots are for $w = 20$ channels, we have to confirm the superiority of the clustering-driven approach in terms

of network throughput, under different number of channels. Thus, we fixed the number of nodes at $n = 70$, we set the traffic load to $k = 30$ and vary the number of channels setting $w = 5, 10, 15$ and 20. The number of clusters was set to be equal to the number of channels (i.e. noc $= w$). The results are illustrated in Fig. 7(a) and (b).

More specifically, Fig. 7(a) shows the network throughput versus the number of channels under uniform traffic, while Fig. 7(b) depicts the network throughput versus the number of channels under poisson traffic. Based on these figures, it is clear that for all algorithms the network throughput is increasing as the number of channels increases and this is expected, since the more network channels, the shorter schedule length. In practice, there is more time space for nodes packets to be scheduled. It is remarkable that CD-MSL offers higher throughput in comparison to the rest algorithms for both uniform and poisson traffic, while its performance is getting better as the number of channels increase.

Indicatively, for $w = 20$ and under uniform traffic, according to Fig. 7(a) the CD-MSL obtains 41.6 Gbps as network throughput while the throughput of EATS, RO-EATS and MSL reach at 36.0, 35.4 and 38.4 Gbps, respectively. In Fig. 7(b), we can see that under poisson traffic, CD-MSL achieves a throughput of 43.3 Gbps, while the throughput of EATS, RO-EATS and MSL is 37.2, 36.4 and 39.8 Gbps, respectively.

6.5. Throughput versus clusters

Given that the proposed algorithm aims at grouping together nodes with similar message destinations, it was challenging to study its performance under different number of clusters. Thus, in the last part of experimentation, we evaluated network throughput under different values of noc. In Fig. 8(a) and (b) we fixed the number of nodes at $n = 70$, the network load at $k = 30$, while we set the number of channels to $w = 20$. Under these parameters, we experimented with noc $= 14, 16, \ldots, 26$.

As expected, the performance of EATS, RO-EATS and MSL is independent of the number of clusters. On the other hand, the performance of CD-MSL is varied under different number of

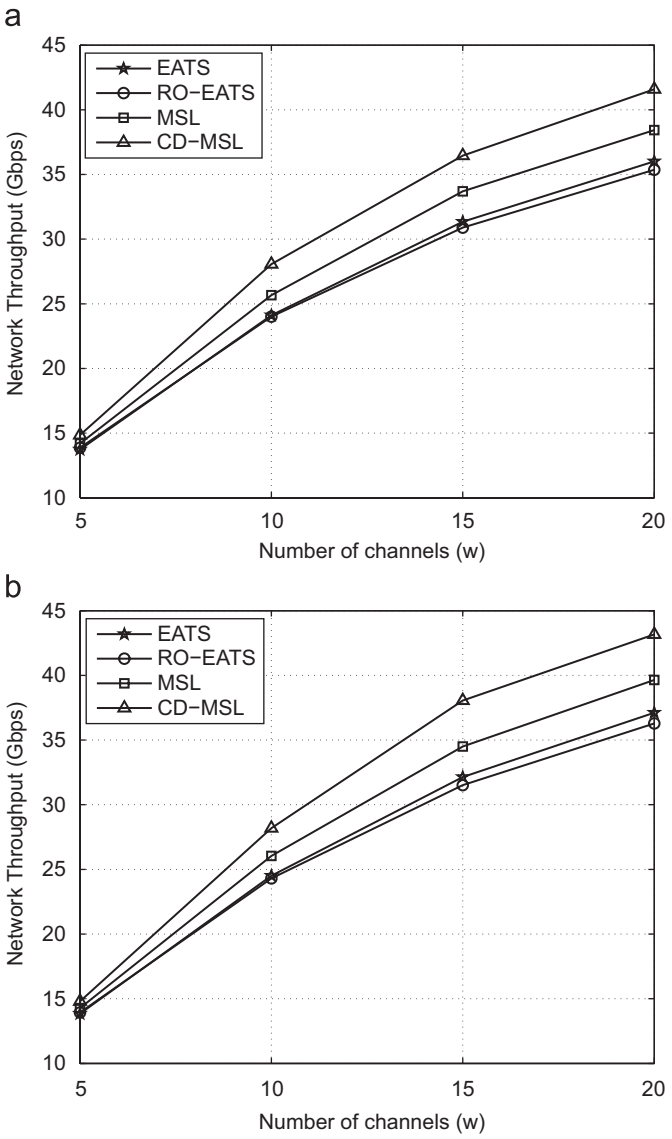

Fig. 7. Network throughput as a function of the number of channels for $n = 70$ nodes, traffic load $k = 30$ and noc $= w$ clusters: (a) uniform traffic and (b) Poisson traffic.

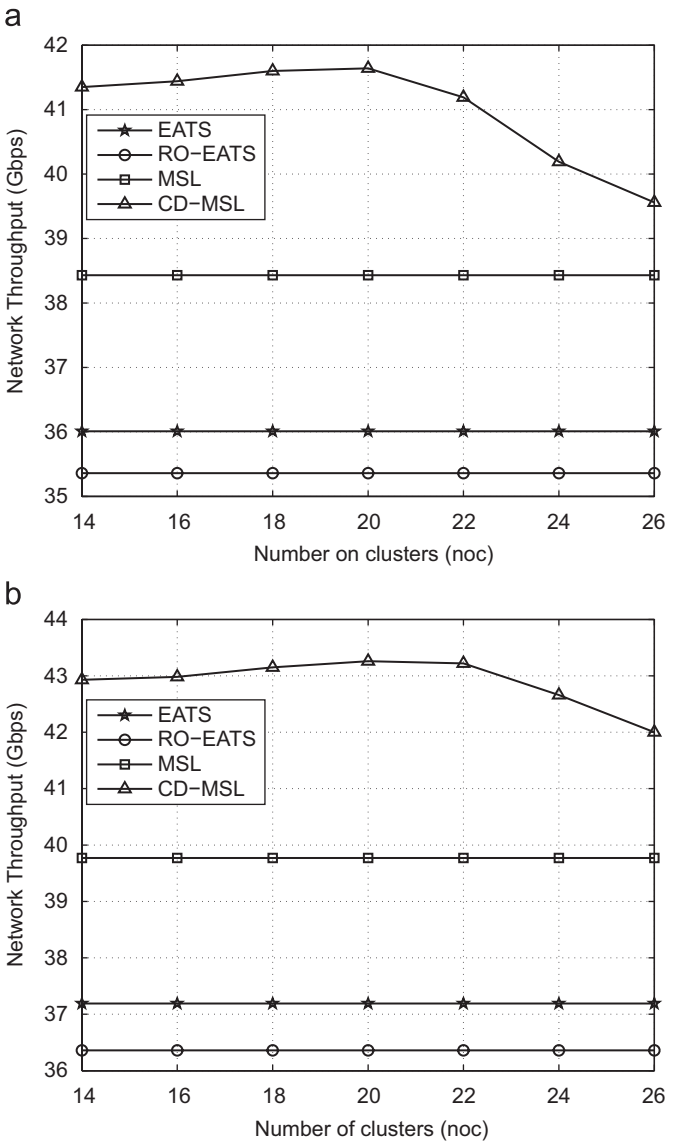

Fig. 8. Network throughput as a function of the number of clusters for $n = 70$ nodes, $w = 20$ channels and traffic load $k = 30$: (a) uniform traffic and (b) Poisson traffic.

clusters, but, in all cases, it clearly outperforms the other three classic scheduling schemes. Especially for noc = 20, CD-MSL reaches its maximum performance confirming the fact that for noc = w the clustering succeeds in creating the shortest schedule.

Indicatively, under uniform traffic, CD-MSL offers up to 41.6 Gbps as network throughput (for noc = 20), which clearly surpass EATS with 36.0 Gbps, RO-EATS with 35.4 Gbps and MSL with 38.4 Gbps. Similar conclusions are obtained for poisson traffic, where the performance of CD-MSL reaches at 43.3 Gbps for noc = 20, while the throughput of EATS, RO-EATS and MSL is clearly lower at 37.2, 36.4 and 39.8 Gbps, respectively.

6.6. Real-time traffic experimentation

On the above simulation results, all packets are considered to be of equal priority. However, since real-time traffic (high-priority packets) represents 25–30% of the Internet traffic, in this subsection, we have conducted experiments where CD-MSL has been extended in order to handle prioritized traffic. In the following experimentation, the shares of high- and low-priority packets is 25% and 75%, respectively. Thus, during each frame, nodes with high- and low-priority packets are clustered separately, as explained at the end of Section 4, and their packets scheduled appropriately. More specifically, the high-priority packets have the privilege of being scheduled prior to low-priority ones. In this way, the proposed scheme succeeds in obtaining significant improvements for real-time traffic, without sacrificing the performance for nonreal-time traffic.

More specifically, Fig. 9(a) and (b) represent mean packet delay as a function of the network throughput in case of total traffic, high-priority and low-priority packets when models A and B are employed, respectively. In these figures, the number of nodes is varied by setting $n = 20, 30, \ldots, 100$, while the number of channels is fixed at $w = 20$. The traffic load is set to $k = 30$ and noc is taken to be equal to 20 clusters. Based on the above figures, it is observed that the curves depicting the mean packet delay of CD-MSL for high- and low-priority packets differ significantly, which means that CD-MSL handles successfully real-time traffic. For example, as the network's throughput increases, CD-MSL achieves a mean delay from 4 to 9 timeslots for real-time traffic, whereas for nonreal-time traffic the delay is from 22 to 47 timeslots. This significantly lower delay for high-priority packets is due to the fact that such packets have the privilege of being

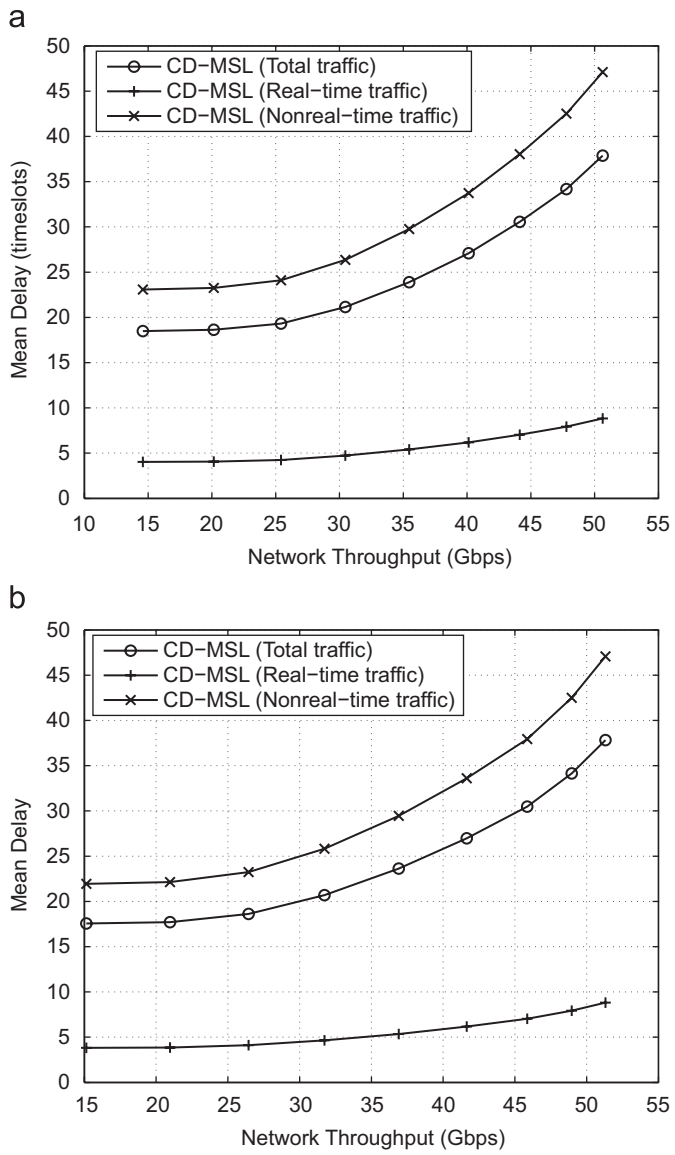

Fig. 9. Mean packet delay of total traffic, real- and nonreal-time traffic as a function of the network throughput: (a) uniform traffic and (b) Poisson traffic.

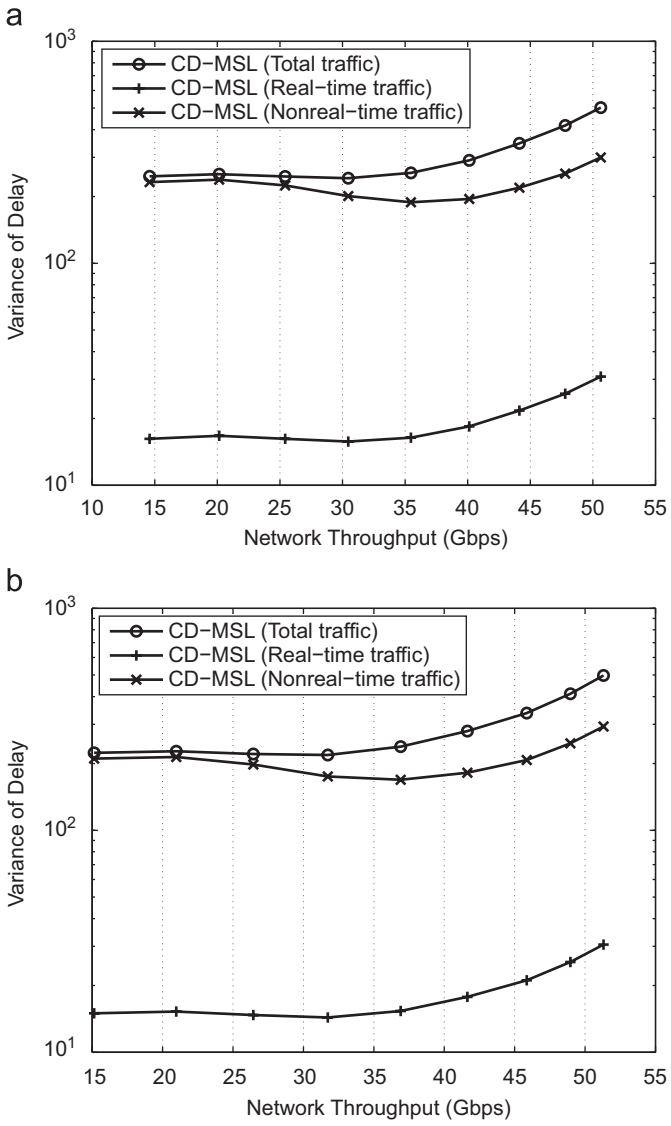

Fig. 10. Variance of delay of total traffic, real- and nonreal-time traffic as a function of the network throughput: (a) uniform traffic and (b) Poisson traffic.

scheduled prior to low-priority ones. However, the above observed improvement is not made in the cost of a high delay for low-priority packets, since as depicted in both Fig. 9(a) and (b), the low-priority packets' curves are very close to the total traffic curves. In practice, as the network's throughput increases, CD-MSL achieves from 19 to 38 timeslots as mean delay for total traffic.

The variance of delay is another important performance metric, especially in case of real-time traffic [13]. In the presence of real-time traffic, it is crucial to keep the variance of delay of this traffic low, in order to avoid long delays. Graphs, depicting the variance of delay for the CD-MSL scheme under prioritized real-time traffic are given in Fig. 10(a) and (b). More specifically, Fig. 10(a) and (b) depict the variance of delay of total traffic, high- and low-priority packets, as a function of the network throughput in case of uniform and poisson model, respectively, for the same values of network's parameters (i.e. n, w, k and noc). Both figures indicate that CD-MSL significantly reduces the variance of delay for real-time traffic in comparison to both total and nonreal-time traffic. For example, as the network's throughput increases, the variance of delay for real-time traffic is, on the average, 90% lower than the one of nonreal-time traffic, and 93% lower than the one of total traffic for both uniform and asymmetric models. Furthermore, it is noticeable that the variance of delay for low-priority packets is lower than that of the total traffic and this is due to the fact that the proposed algorithm closely schedules packets of the same priority.

6.7. Major observations

The following conclusions can be extracted from the simulation results presented in Figs. 4–10:

(1) For any number of nodes, channels and clusters as well as for any traffic model, the proposed CD-MSL scheme is superior to the conventional scheduling schemes EATS, RO-EATS and MSL, since it achieves to create a shorter schedule, which significantly improves the network's throughput and reduces the mean packet delay. This is due to the fact that CD-MSL takes into account the specific nodes' demands driven by the clustering of the network's source nodes.
(2) The proposed scheme can be easily extended to handle real-time traffic (high-priority packets) such as audio or video. This extension provides a superior performance for high-priority packets in terms of mean delay and variance of delay, without sacrificing the performance of low-priority packets such as text, e-mail or file transfer.

7. Conclusions and future work

A novel scheduling scheme which is driven by clustering techniques is introduced. The proposed CD-MSL algorithm handles the channel assignment issue taking into account both channels and receivers' availability information. Furthermore, it addresses the nodes' service order issue in an innovative clustering-driven way which considers both the source and destination nodes. In this way, the individual traffic pattern of each source node is taken into account and the source nodes are prioritized on the basis of not scheduling consecutive messages to the same destination node. As a result, the network performance is significantly upgraded.

The idea of using clustering algorithms for traffic scheduling is applicable to a broad range of networks, including optical and wireless LANs, and wireless push systems. We are currently working in this direction.

References

[1] Mukherjee B. Optical WDM networks. Berlin: Springer; 2006.
[2] Papadimitriou G, Papazoglou C, Pomportsis A. Optical switching: switch fabrics, techniques, and architectures. IEEE/OSA J Lightwave Technol 2003;21(2):384–405.
[3] Mukherjee B. WDM optical communication networks: progress and challenge. IEEE J Sel Areas Commun 2000;18(10):1810–24.
[4] Green P. Progress in optical networking. IEEE Commun Mag 2001;39(1):54–61.
[5] Yao S, Mukherjee B, Dixit S. Advances photonic packet switching: an overview. IEEE Commun Mag 2000;38:84–94.
[6] Lin H, Liu P. Reducing packet delay in single-hop WDM networks using fixed transceiver array and adaptive channel allocation. IEEE/OSA J Lightwave Technol 2006;24(12):4925–36.
[7] Sarigiannidis P, Papadimitriou G, Pomportsis A. A high throughput scheduling technique, with idle timeslot elimination mechanism. IEEE/OSA J Lightwave Technol 2006;24(12):4811–27.
[8] Sivalingam K, SS, editors. Optical WDM networks—principles and practice, Berlin: Springer; 2000.
[9] Bernstein G, Rajagopalan B, Saha D. Optical network control: architecture, protocols, and standards. Reading, MA: Addison-Wesley; 2003.
[10] Papadimitriou G, Tsimoulas P, Obaidat M, Pomportsis A. Multiwavelength optical LANs. New York: Wiley; 2003.
[11] Jia F, Mukherjee B, Iness J. Scheduling variable-length messages in a single-hop multichannel local lightwave network. IEEE/ACM Trans Networking 1995;3(4):477–88.
[12] Ma M, Hamidzadeh B, Hamdi M. An efficient message scheduling algorithm for WDM lightwave networks. Comput Networks 1999;31(20):2139–52.
[13] Diao J, Chu P. Packet rescheduling in WDM star networks with real-time service differentiation. IEEE/OSA J Lightwave Technol 2001;19(12):1818–28.
[14] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surveys 1999;31(3):264–323.
[15] Petridou S, Koutsonikola V, Vakali A, Papadimitriou G. Time aware web users clustering, IEEE Trans Knowl Data Eng 2008; to appear.
[16] Sarigiannidis P, Papadimitriou G, Pomportsis A. Lena: an efficient channel eclectic algorithm for WDM optical networks. Opt Laser Technol 2008;40(1):39–51.
[17] Sarigiannidis P, Papadimitriou G, Pomportsis A. A novel medium access control protocol for optical local networks based on data request ordering. Opt Laser Technol 2008;40(1):175–93.
[18] Petridou S, Sarigiannidis P, Papadimitriou G, Pomportsis A. An efficient clustering oriented algorithm for message scheduling on WDM star networks. In: Proceedings of the 14th IEEE/CVT annual symposium on communications and vehicular technology in the Benelux (IEEE SCVT '07), Delft, The Netherlands, 2007.
[19] Petridou S, Sarigiannidis P, Papadimitriou G, Pomportsis A. Clustering based scheduling: A new approach to the design of scheduling algorithms for WDM star networks. In: Proceedings of the 14th IEEE/CVT annual symposium on communications and vehicular technology in the Benelux (IEEE SCVT '07), Delft, The Netherlands, 2007.
[20] Shang-Tse C, Goel A, McKeown N, Prabhakar B. Matching output queueing with a combined input/output-queued switch. IEEE J Sel Areas Commun 1999;17(6):1030–9.
[21] Akyildiz IF, Levine DA. A collision-free mac protocol for optical star LANS. Comput Networks 1996;28(3):371–90.
[22] Ofek Y, Sidi M. Design and analysis of a hybrid access control to an optical star using WDM. J Parallel Distributed Comput 1993;17(3):259–65.
[23] Semann G, Humblet P. Timing and dispersion in WDM optical star networks. In: Proceedings of IEEE INFOCOM'93.
[24] Gionis A, Mannila H, Tsaparas P. Clustering aggregation. In: Proceedings of international conference on data engineering (ICDE '05), Tokyo, 2005. p. 341–52.
[25] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. Berlin: Springer; 2001.
[26] Johnson E, Mishra M, Sivalingam K. Scheduling in optical WDM networks using hidden Markov chain based traffic prediction. J Photonic Network Commun 2001;3(3):271–86.
[27] Li B, Ma M, Hamdi M. MAC protocols for WDM networks: survey and summary. Optical WDM networks: principles and practice. Boston, MA, USA: Kluwer Academic Publishers; 2000.