On the Use of Clustering Algorithms for Message Scheduling in WDM Star Networks

Sophia G. Petridou, *Member, IEEE*, Panagiotis G. Sarigiannidis, *Member, IEEE*, Georgios I. Papadimitriou, *Senior Member, IEEE*, and Andreas S. Pomportsis

***Abstract*—Scheduling algorithms in wavelength division multiplexing (WDM) single-hop networks aim at producing an effective schedule in order to improve the networks' performance. Apart from channel assignment, the message sequencing is an important issue that have to be addressed when designing media access control (MAC) protocols for WDM networks. Up until now, popular approaches have not extensively addressed the order in which the messages are scheduled even though the messages' service order can considerably contribute to the advance of network performance. This paper introduces a new approach to the design of message scheduling algorithms for WDM star networks, which is based on the use of clustering techniques. The proposed clustering oriented—earliest available time scheduling (CO-EATS) creates groups of nodes whose messages are destined to common nodes. The goal of the proposed CO-EATS scheme is to decrease the probability of scheduling messages to the same destination at successive order. The simulation results have shown that the proposed scheme improves channel utilization and as a result it leads to higher network throughput while it keeps mean packet delay at low levels in comparison with conventional scheduling algorithms.**

***Index Terms*—Clustering, optical wavelength division multiplexing (WDM) networks, reservation, scheduling.**

I. Introduction

THE introduction of multigigabit applications has led to a dramatic increase in bandwidth demands of the emerging new generation of fiber optic local area networks (LANs) and wide area networks (WANs). Nowadays, there is a need for LANs supporting 100 Gbps or even higher speeds for applications requiring bandwidth beyond existing capabilities. These include data centers, Internet exchanges, high-performance computing, and video-on-demand delivery [1]. Optical networking and wavelength division multiplexing (WDM) technology can cope with the above demands, since they can provide gigabit-per-second data rates on independent channels (or wavelengths) that simultaneously transmit data streams to a single or multiple users within a single optic fiber [2], [3]. Multiplexing and demultiplexing a number of channels can lead to significant improvement of the optical network performance.

The main classes of WDM optical networks are point-to-point links networks, broadcast-and-select networks, wavelength-routed networks, and passive optical networks [4]. The star topology using a broadcast-and-select architecture has been shown to dominate in LANs connecting computing nodes via two-way fibers to a passive star coupler. This star coupler is located at the center of the network and its role is to combine the incoming optical streams from the multiple transmitting nodes. Subsequently, each node has to use its receiver in order to select the desired wavelength for data reception. In general, network nodes can transmit and receive messages on any of the available channels employing one or more fixed or tunable transmitter(s) (FT or TT) and one or more fixed or tunable receiver(s) (FR or TR) [5].

Even though the passive star coupler may degrade total throughput and security performances of a WDM broadcast-and-select star networks, its usage has some important advantages [2], [6] in comparison to a switch-based approach [7], [8]. The passive property of the optical star coupler is important for network reliability, since no power is needed to operate the coupler. Given that the network hub constitutes a single point of failure, the use of a passive star coupler at this point offers to WDM passive star networks a significant advantage in relation to switch-based networks. Other advantages of the passive star coupler in comparison to a switch-based solution are its natural multicasting capability, simplicity and low cost.

An important issue in such WDM broadcast-and-select networks is to be specified the way that nodes transmit on the available channels [9]. Thus, a media access control (MAC) protocol is needed to coordinate the nodes' data transmission and prevent collisions. In general, collisions can be characterized either as channel collisions, in case that two or more nodes transmit on the same channel simultaneously, or as receiver collisions, in case that two or more nodes destine their data streams to the same node at the same time [10]. A basic distinction among MAC protocols is based on the existence of a channel which is characterized as control channel. More specifically, in the preallocation-based protocols, there is no control channel, while in the pretransmission coordination-based protocols a control channel is used for nodes' coordination before their actual data transmission. As depicted in Fig. 1, the preallocation-based protocols are further divided into fixed-assignment or static access and random access protocols while the pretransmission coordination-based protocols can be characterized either as with receiver collisions or as without receiver collisions according to whether or not prevent receiver collisions [2]. Representatives of MAC protocols that allow receiver collisions can be found in [11]–[14], while [15]–[23] present pretransmission coordination-based protocols without receiver collisions.

Manuscript received October 27, 2007; revised April 4, 2008. Current version published December 19, 2008.

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, Greece (e-mail: spetrido@csd.auth.gr; sarpan@csd.auth.gr; gp@csd.auth.gr; apombo@csd.auth.gr).

Digital Object Identifier 10.1109/JLT.2008.926913

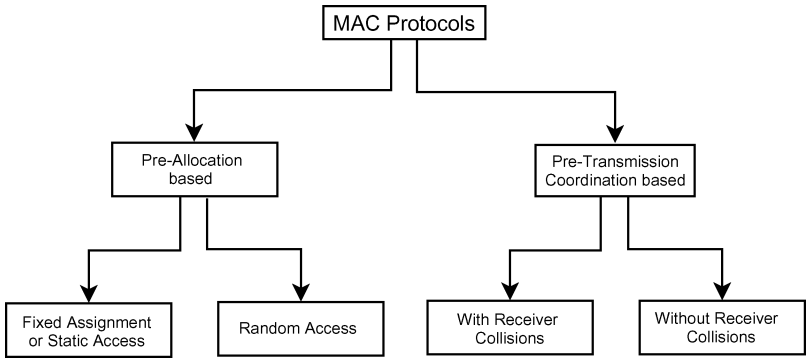


Fig. 1. Classification of MAC protocols according to the existence of control channel.

In this paper, we focus on pretransmission coordination-based protocols without receiver collisions for variable-length message transmission. In such protocols, a way to eliminate receiver collisions is to design a scheduling algorithm which address two crucial issues, namely the message sequencing and channel assignment issues. Among the well-known, efficient scheduling algorithms for local area WDM networks with broadcast-and-select architecture are the earliest available time scheduling (EATS) [15], the receiver oriented-earliest available time scheduling (RO-EATS) [16], and the minimum scheduling latency (MSL) [17]. EATS addresses the channel assignment without, however, handling message sequencing, since it schedules messages according to their arrival order and ignores the fact that the messages' service order may affect the network's performance. RO-EATS comes as an extension of the EATS and its core idea is to prioritize messages that are destined to the least used destination nodes. MSL modifies the actual choice of transmission data channel on the basis of the minimum scheduling latency, i.e., the channel for which the shorter time interval will pass from the time it becomes available. Although both RO-EATS and MSL deal with message sequencing issue, they do not prevent the messages' scheduling to the same destination at successive order which increases the messages' schedule length resulting in limited channel utilization.

A. Contribution

This paper introduces a new algorithm that deals with the message sequencing issue based on the clustering [24] of the network's nodes. The proposed clustering oriented—earliest available time scheduling (CO-EATS) organizes the network's nodes into clusters according to the destination of their messages. In general, clustering aims at creating groups of items, i.e., clusters on the basis that items assigned to the same cluster are "similar" to each other and "dissimilar" to the nodes belonging to other clusters [24]. In our framework, CO-EATS groups together nodes with common message destination. Thus, given that each cluster will consist of nodes which probably destine their messages to common destination, CO-EATS defines the message sequencing choosing for transmission nodes belonging to different clusters. In this way, it decreases the probability of scheduling messages to the same destination at successive order. As a result, the schedule length is reduced and the network performance is upgraded.

More specifically, the proposed algorithm is inspired by the observation that scheduling consecutive messages to the same destination node may not fully use the available channels. Thus, it is necessary to enhance the aforementioned traditional schemes with an efficient message sequencing mechanism which would distinguish consecutive messages destined to the same node. Clustering network nodes on the basis of their destination provide us with the necessary mechanism. Therefore, discovering groups of nodes with common message destination and scheduling their messages properly could lead to higher network performance without aggravating the mean packet delay.

Data clustering is a common technique for data analysis, which is used in many fields, including Web data mining [25], [26], image analysis [27], and bioinformatics [28]. Especially on the Web, many research efforts have focused on grouping users that present similar access behavior [29], [30]. Collecting information about users' behavior and extracting their usage patterns can be quite important for providing dynamic Web content, for effective Web site structuring and management, as well as, for improving specific applications such as e-commerce via pages' caching and prediction.

The remainder of this paper is organized as follows. Section II provides the network background, while Section III presents related message scheduling MAC protocols. Clustering background is given in Section IV. Our new scheduling algorithm is described in Section V, while Section VI discusses the simulation results. Conclusions are given in Section VII.

II. Network Background

Consider a single-hop, broadcast-and-select WDM star network consisting of n nodes and $w+1$ channels (wavelengths), where $\Lambda = \{\lambda_1, \ldots, \lambda_w\}$ is the set of the data channels, while one channel λ_0 is used for coordination (i.e., control channel). Given that each of the n network's nodes can either transmit a message or receive more than one messages, the sets $S = \{s_1, \ldots, s_n\}$ and $D = \{d_1, \ldots, d_n\}$ denote the source and destination nodes (e.g., s_i and d_i, where $i = 1, \ldots, n$, refer to the same node, which in the first case behaves as source while in the second case as destination node). Each source node (s_i) is provided with a fixed-tuned transmitter (FT) for the control channel and with a tunable transmitter (TT) for the data channels. These are connected to a 2×1 combiner before reaching

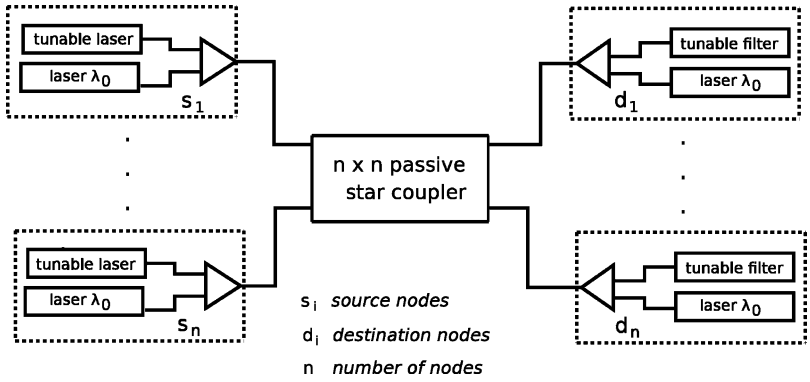


Fig. 2. Network architecture.

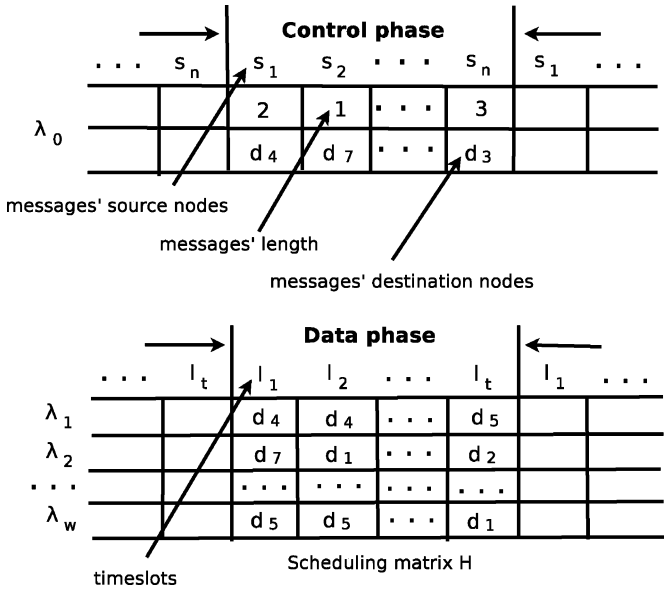


Fig. 3. Control and data phases.

the $n \times n$ passive star coupler via an optical fibre. The n outputs of the star coupler are connected via n separate fibres to the destination nodes (d_i), which are equipped with 1×2 splitters that separate the data from the control channel. The control channel is connected to a receiver that is fixed-tuned to this channel (FR), while the data channel is led to a tunable receiver (TR) capable of tuning over the whole range of available data channels. Hence, the system is CC-FTTT-FRTR as it is depicted in Fig. 2.

In the above CC-FTTT-FRTR implementation, each transmission frame is divided into two phases, namely the control and data phase as illustrated in Fig. 3. During the control phase, a source node s_i sends its control packet to the common control channel in a TDM-fashion, while 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.

Example 1: According to the Fig. 3, the source node identified as s_1 requests during the control phase (λ_0) the transmission of a message whose length is 2 to the destination node d_4, while the node s_2 requests the transmission of a 1-length message to the node d_7. As a result, during the data phase, we can see two packets to be scheduled on the data channel λ_1 occupying the timeslots l_1, l_2 and being destined to the node d_4, while the 1 packet to the node d_7 is scheduled on the channel λ_2 during the timeslot l_1. □

In such a network, it is obvious that two or more source nodes might cause either channel collision, transmitting messages on the same data channel simultaneously, or receiver collision, transmitting messages destined to the same node simultaneously. Thus, in order to avoid collisions two tables are used on each node, namely the receiver available time (RAT) and the channel available time (CAT) tables [31]. The RAT table consists of n elements, where $\text{RAT}(d_i) = x, i = 1, \ldots, n$ implies that destination node d_i will be available after x timeslots. If $\text{RAT}(d_i) = 0$, then node d_i is currently idle and no reception is scheduled for it. The CAT table consists of w elements, where $\text{CAT}(i) = y, i = 1, \ldots, w$, denotes that channel i will be available after y timeslots. If $\text{CAT}(i) = 0$, then data channel i is currently available. RAT and CAT are needed to avoid receiver and channel collisions, respectively.

A MAC protocol handles the above issues and runs a scheduling algorithm at the end of the control phase in each frame. The goal of the scheduling algorithm is to produce a $w \times t$ scheduling matrix H, where t denotes the length on the schedule in timeslots. Each $h(i, j)$ element, $i = 1, \ldots, w$ and $j = 1, \ldots, t$, represents the destination node that receives a message on channel λ_i during the timeslot l_j. Fig. 3 provides an example of such a scheduling matrix.

III. Related Variable-Length Message Scheduling MAC Protocols

A well-known scheduling algorithm for such a network is the earliest available time scheduling (EATS) [15]. The core idea of EATS is to assign a message to the data channel that has the earliest available time among all the network data channels. Once the data channel is assigned, the algorithm proceeds to the message schedule as soon as that channel becomes available. EATS uses the aforementioned RAT and CAT tables in order to keep a record of the channels and receivers state. With this global information in each node, the distributed EATS operates as follows: transmit a control packet on the control channel; select the channel with the earliest available time; define the

transmission schedule based on RAT and CAT; and update these tables according to the last scheduled message. The algorithm produces the $w \times t$ scheduling matrix H operating in time linear on the number of nodes and channels $O(nw)$.

The receiver oriented-earliest available time scheduling (RO-EATS) protocol [16] employs the same system structure and network service as EATS. The difference between the two protocols is that RO-EATS considers not only the transmission channel, i.e., channel assignment, but also the transmission order of the corresponding messages declared during the control frame, i.e., message sequencing. RO-EATS is executed by all nodes when they receive the entire control frame (not only a part of it like EATS). This algorithm also uses the RAT and CAT and more specifically sort the RAT table in ascending order so as to schedule the message which is destined to the least visited destination node. In practice, it first considers the earliest available receiver among all the network nodes and then selects the message which is destined to this receiver from those which are ready and identified by the control frame. Then, a data channel is selected and assigned to the message that is to be scheduled based on the sorted RAT and following the EATS logic. RO-EATS achieves higher throughput and lower average message delay in comparison with EATS. However, its complexity is higher than this of EATS due to the RAT sorting.

The minimum scheduling latency (MSL) protocol [17] selects the channel with the minimum scheduling latency. Similarly to EATS, the MSL is executed by all nodes after the reception of each control packet. In this manner, MSL starts the construction of the transmission schedule after the reception of the first node. The main idea of MSL is that, instead of always selecting the earliest available channel, it selects the available channel optimally by taking into account the destination availability. Hence, the difference with EATS concerns the way that the transmission channel is selected. EATS selects the transmission channel with the minimum CAT, while MSL selects the channel with the minimum scheduling latency. The MSL scheme has higher processing requirements but improves the performance of EATS and RO-EATS in terms of mean packet delay and channel utilization.

IV. Clustering Background

For the clustering process, the sets S and D are organized into an $n \times n$ message table M, whose $m(i,j)$ element, $i = 1, \ldots, n$ and $j = 1, \ldots, n$, indicates the length of the message from the source node s_i to the destination node d_j. Given that each s_i node can transmit a message per frame, it is obvious that the ith row of the M table will have one nonzero value. On the other hand, the jth column of the M table can have more than one nonzero values indicating that each d_j node can receive more than one messages. Under this notation, each node s_i is considered to be a multivariate vector consisting of n values and could be denoted as follows:

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

A clustering Cl of S is a partition of S into noc disjoints sets, i.e., clusters $C_1, \ldots, C_{noc}$, that is, $\bigcup_{i=1}^{noc} C_i = S$ and $C_i \bigcap C_j = \emptyset$ for all $i \neq j$. The noc clusters $C_1, \ldots, C_{noc}$ consist of $|C_1|, \ldots, |C_{noc}|$ 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 membership of a node s_i, where $i = 1, \ldots, n$, to a cluster C_j, where $j = 1, \ldots, noc$, is defined by the function f as follows:

$$f(s_i, C_j) = \begin{cases} 1, & \text{if } s_i \in C_j \\ 0, & \text{otherwise.} \end{cases}$$

Based on the above, it is apparent that similarity is fundamental to the definition of a cluster. Thus, a measure of the similarity between two patterns (in our case source nodes' patterns) is essential to most clustering approaches. In practice, it is most common to calculate the dissimilarity between two patterns using a distance measure. However, because of the variety of distance measures, patterns' representation plays a major role to the selection of distance measure [24]. Conventionally, patterns are represented as vectors whose values can be either quantitative (continuous values, e.g., weight, discrete values, e.g., the number of visits of a Web user or interval values, e.g., the duration of an event) or qualitative (nominal, e.g., "red" or ordinal e.g "cool").

Since our nodes' patterns are represented as vectors with discrete values, we will focus on the Squared Euclidean distance,[1] which is a well-known and widely used distance measure in the vector-space model [24], [26]. Therefore, the evaluation of the dissimilarity between two source nodes, e.g., $s_x, s_y \in S$ can be expressed by the distance of their vectors. Therefore, $d_E(s_x, s_y)$ denotes the Squared Euclidean distance of the nodes' vectors $M(x,:)$ and $M(y,:)$

$$d_E(s_x, s_y) = \|M(x,:) - M(y,:)\|^2.$$

Let us consider an arbitrary cluster $C_j, j = 1, \ldots, noc$, 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:

$$MeansD(j,:) = \frac{1}{|C_j|} \sum_{i=1}^{n} f(s_i, C_j) * M(i,:), \quad j = 1, \ldots, noc.$$

Since both $M(i,:)$ and $Means(j,:)$ are vectors, their dissimilarity is measured by their squared Euclidean distance $d_E(s_i, c_j)$. 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

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

[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: $\|x_i - y_i\|^2$.

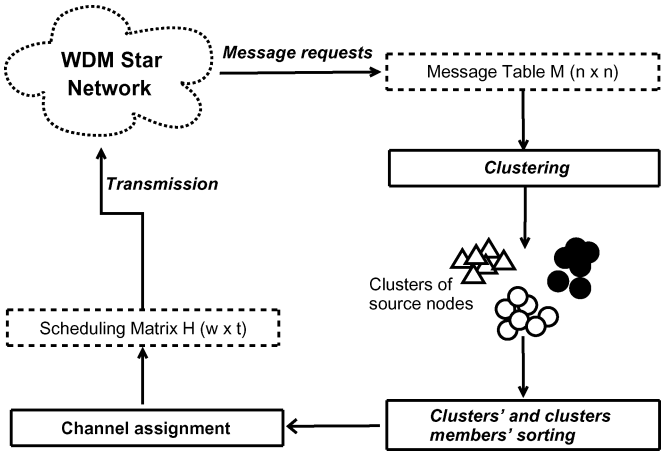


Fig. 4. CO-EATS overview.

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. A Cl that minimizes J groups together nodes from the set S that probably destine their messages to the same nodes of the set D.

For the purpose of our clustering, we employed the well-known and widely used K-means partitional clustering algorithm [32]. K-means classifies a given dataset 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 in summary is: 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.

V. The Proposed Algorithm

The proposed CO-EATS is a two-step process which firstly handles the message sequencing and then deals with channel assignment based on the EATS algorithm as depicted in Fig. 4. The core idea is that message sequencing should take into account the messages' destination. The proposed algorithm aims at grouping together nodes from S with common destination. The goal is that messages to the same destination should not be scheduled in a successive order. Thus, CO-EATS schedules in sequence messages from nodes belonging to different clusters. Furthermore, CO-EATS prioritizes clusters as well as the members on each clusters according to the length of their messages.

More specifically, during the first step, i.e., clustering step, we employ the K-means in order to produce the Cl clustering of S. Then, given the Cl and the message table M, we sort the members on each cluster according to the length of their messages. More specifically, given that each node s_i, where $i = 1, \ldots, n$, transmits a message per frame, clusters' members sorting gives priority to the nodes with long-length messages. Similarly, using the $Means$ table which consists of the clusters representatives' vectors $Means(j,:)$, where $j = 1, \ldots, noc$, the $SortedC$ is computed in order that we prioritize the clusters with longer messages. 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 CO-EATS needs, i.e., the vectors with high values. For example, given the vectors (1,1,1) and (3,0,0), their elements sum is 3, while their lengths are $\sqrt{3}$ and 3 respectively which means that the second vector will have priority over the first one in the service order. Then, the calculated $SortedM$ and $SortedC$ are used in order that the message sequencing will be defined. Once the $MsgSequencing$ is formed, the algorithm proceeds to the second step called the channel assignment step. The goal of the function $ChannelAssignment$ is to form the scheduling matrix H using the EATS algorithm.

Algorithm 1 The CO-EATS 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.

1: $/^*Clustering\ Step^*/$.
2: $(Cl, Means) = K - means(M, noc)$.
3: $SortedM = Quicksort(M, Cl)$.
4: $SortedC = Quicksort(Means)$.
5: $MsgSequencing = Sequencing(SortedM, SortedC)$.
6: $/^*Channel\ Assignment\ Step^*/$.
7: $H = ChannelAssignment(MsgSequencing)$.

Theorem 1: The CO-EATS has time complexity $O(nlogn + nw)$.

Proof: During the clustering step we employ the K-means algorithm (line 2) whose time complexity is $O(n\,noc\,r)$, 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 [24]. Thus, the Cl clustering is computed in time linear on the number of nodes: $O(n)$. The $Quicksort$ functions (lines 3 and 4) sorts the nodes and clusters' representatives in $O(nlogn + noc\,log(noc))$ time. The $Sequencing$ function (line 5) takes time $O(n\ noc)$ to arrange the messages from the n nodes according to the $SortedM$ and $SortedC$. The total time complexity of the clustering step is, thus, $O(n + nlogn + noc\,log(noc) + n\,noc)$ which becomes $O(nlogn)$, since noc is relatively small compared to the number of nodes n. During the second step, the $ChannelAssignment$ function (line 6) needs $O(nw)$ time [15] to form the scheduling matrix H, where w is the number of channels. As a result, the total complexity of CO-EATS is $O(nlogn + nw)$. ■

To facilitate the comprehension of the proposed scheme, let us consider a 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 $(\lambda_1, \lambda_2, \lambda_3)$

TABLE I
BASIC 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 x 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 x t scheduling matrix
Cl	Clustering process
noc	Number of clusters
C_j	Cluster, $j = 1, \ldots, noc$
c_j	Cluster representative
Means	The noc x n clusters representatives' message table
d_E	Nodes distance over their messages' destination
J	Objective function

TABLE II
TABLE $Members$ BEFORE MEMBERS' SORTING

C_1	s_3	s_6		
C_2	s_7	s_8		
C_3	s_1	s_2	s_4	s_5

and having the upper bound of nodes' messages length $k = 4$ packets. Then, a 8×8 message table M could be the following:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Example 2: In the above message table M, the fact that $M(3,7) = 4$ means that the source node s_3 sends a message of length 4 to the destination node d_7. Node d_3, as well as node d_7, will receive two messages, while s_1 and s_4 have no messages to transmit. □

Applying the K-means algorithm for $noc = 3$ in the above message table M results in $Cl = (3,3,1,3,3,1,2,2)$ which can be represented by the following $Members$ table.

From Table II, it holds that $s_3, s_6 \in C_1$, $s_7, s_8 \in C_2$ while $s_1, s_2, s_4, s_5 \in C_3$. As we discussed in Section IV, it is obvious that Cl places to the same cluster source nodes which are similar in terms of their destination nodes, e.g., the nodes s_3 and s_6 destine their messages to the node d_7, while s_7 and s_8 to the node d_3. Given that we chose $noc = 3$ and C_1, C_2 consist of actually similar nodes, the rest of the nodes, i.e., s_1, s_2, s_4, and s_5 are forced to be placed to the same remaining cluster, i.e., C_3. Then, sorting the members on each cluster according to the length of their message results in swapping the nodes of C_3. Therefore, the Table II is updated as follows.

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

$$Means = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0.5 & 0 & 0 \end{pmatrix}.$$

Sorting $Means$ provides our algorithm with the following service order: C_1, C_2, C_3 since $|Means(1,:)| = 3$, $|Means(2,:)| = 3$ and $|Means(3,:)| = \sqrt{0.3125}$. To this point, given that each cluster consists of nodes with probably the same destination, our scheme should separate them taking, at the same time, into account the result of $Means$ sorting. Therefore, the message sequencing is defined as $s_3, s_7, s_5, s_6, s_8, s_2, s_1, s_4$ instead of the sequential one $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$.

TABLE III
TABLE $Members$ AFTER MEMBERS' SORTING

C_1	s_3	s_6		
C_2	s_7	s_8		
C_3	s_5	s_2	s_1	s_4

TABLE IV
SCHEDULING MATRIX H PRODUCED BY CO-EATS

	Timeslots						
	l_1	l_2	l_3	l_4	l_5	l_6	l_7
λ_1	d_7	d_7	d_7	d_7	d_4		
λ_2	d_3	d_3	d_3		d_3	d_3	d_3
λ_3	d_6	d_6				d_7	d_7

TABLE V
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
λ_1	d_4					d_7	d_7		
λ_2	d_7	d_7	d_7	d_7			d_3	d_3	d_3
λ_3	d_6	d_6	d_3	d_3	d_3				

Table IV depicts the scheduling matrix H produced by the proposed CO-EATS 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 V–VII 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 the CO-EATS is 71.4%, which is significantly improved in comparison with EATS whose channel utilization is 55.6%, as well as with that of RO-EATS and MSL which both provide 62.5% utilization. In terms of the mean packet delay the observed values are 2.6 (CO-EATS), 3.2 (EATS), 2.8 (RO-EATS), and 2.8 (MSL) timeslots, which means that the proposed scheme is presented to have similar performance with the RO-EATS and MSL while it clearly outperforms the EATS.

VI. SIMULATION RESULTS

To evaluate the proposed algorithm, we carried out experimentation where we compare CO-EATS with EATS, RO-EATS, and MSL. More specifically, the experiments are conducted using a discrete-event simulator implemented in C environment. We experimented with different number of nodes n, channels w, and clusters noc, while we also evaluated the algorithms'

TABLE VI
SCHEDULING MATRIX H PRODUCED BY RO-EATS

	Timeslots							
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
λ_1	d_3	d_3	d_3				d_7	d_7
λ_2	d_4	d_7	d_7	d_7	d_7			
λ_3	d_6	d_6			d_3	d_3	d_3	

TABLE VII
SCHEDULING MATRIX H PRODUCED BY MSL

	Timeslots							
	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
λ_1	d_4	d_3	d_3	d_3		d_3	d_3	d_3
λ_2	d_7	d_7	d_7	d_7				
λ_3	d_6	d_6						

performance under different traffic load k, where k expresses the maximum message length requested per node per frame. The performance of the compared algorithms is measured in terms of network throughput and mean packet delay.

Let us suppose 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 and M the message $n \times n$ table, the network throughput is defined as follows:

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

On the other hand, the mean packet delay is defined as the time required for a head-of-line packet in the waiting queue to gain access to the medium (MAC delay) [33].

The simulation results are produced according to the following approximations.

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 $r = 10$ Gbps [34].
3) The outcome results from 10000 transmission frames.

A. Data Generation

For the purpose of our experimentation, we have generated data based on three distinct models. According to the first model, namely the model A, it is assumed that the packet arrival process at each of the queues follows Uniform distribution. In practice, the source nodes s_i, where $i = 1, \ldots, n$, may send messages of 0 to k length (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, it is assumed that the packet arrival process follows a Poisson process. In general, the Poisson distribution of the number of packets arriving at a specific queue per frame is defined as

$$p(X;\lambda) = \frac{e^{-\lambda}\lambda^X}{X!} \tag{2}$$

where $p(X;\lambda)$ is the probability of X packets being assigned to a specific queue during a specific frame whereas λ 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 λ for these three classes are defined as $k/4$, $k/2$, and $3k/4$, respectively [18].

The third model, namely the model C, assumes self-similar traffic, since recent studies have revealed that LAN traffic is statistically self-similar or long range dependent in nature (i.e., bursty over a wide range of time scales) [35]. Among several methods used for generating self-similar synthetic traces [36], [37], we employ the well-known ON/OFF model, which assumes ON periods during which packets are sending at fixed rate, whereas OFF periods are idle [38], [39]. The length of ON and OFF periods is Pareto distributed. According to [40] suggestion, we use multiple aggregated Pareto sources, in our case 16 [41], with a Hurst parameter equal to 0.9 [35].

B. Experimentation Under Uniform and Poisson Traffic

Fig. 5(a) and (b) depicts 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 $k = 30$ following the model A and B, respectively. The number of channels is set to $w = 20$ and the number of clusters is set to $noc = 20$. Defining $noc = w$ we succeed in not scheduling consecutive messages to the same destination, since we choose to transmit messages from nodes belonging to different clusters. These messages probably have different destinations. The throughput improvement in case of the CO-EATS algorithm proves that its schedule length is reduced. It is apparent that for all values of n the CO-EATS provides steadily higher throughput compared to EATS, RO-EATS, and MSL.

Indicatively, under model A and for $n = 80$ nodes, the network throughput provided by CO-EATS is improved as much as 13.4% over EATS, 15.3% over RO-EATS and 7.1% over MSL. Under model B and for $n = 70$ nodes, the improvement of CO-EATS over EATS, RO-EATS and MSL are of the level of 11.4%, 13.4%, and 5.4%, respectively. It should be noticed that these are the maximum observed improvements, which were expected to be noticed for a high value of n, since the ratio between noc and n significantly contributes to the performance of CO-EATS. This is confirmed by the minimum observed differences, which occur for $n = 20$, where the contribution of the clustering is of no value, i.e., each node constitutes a cluster. Overall, we can claim that independent of the number of nodes and traffic pattern the proposed approach is superior since it creates a shorter schedule which advances the network's throughput.

The second system metric which is important to be evaluated is the mean packet delay. To this point, we should make clear that all algorithms have almost the same performance in terms of mean packet delay, while the proposed scheme is clearly and steadily improved in the context of network throughput. More specifically, on the basis of mean packet delay, CO-EATS behaves better in comparison with the EATS without, however, overcoming the performance of RO-EATS and MSL. This is expected since the significant throughput improvement results in

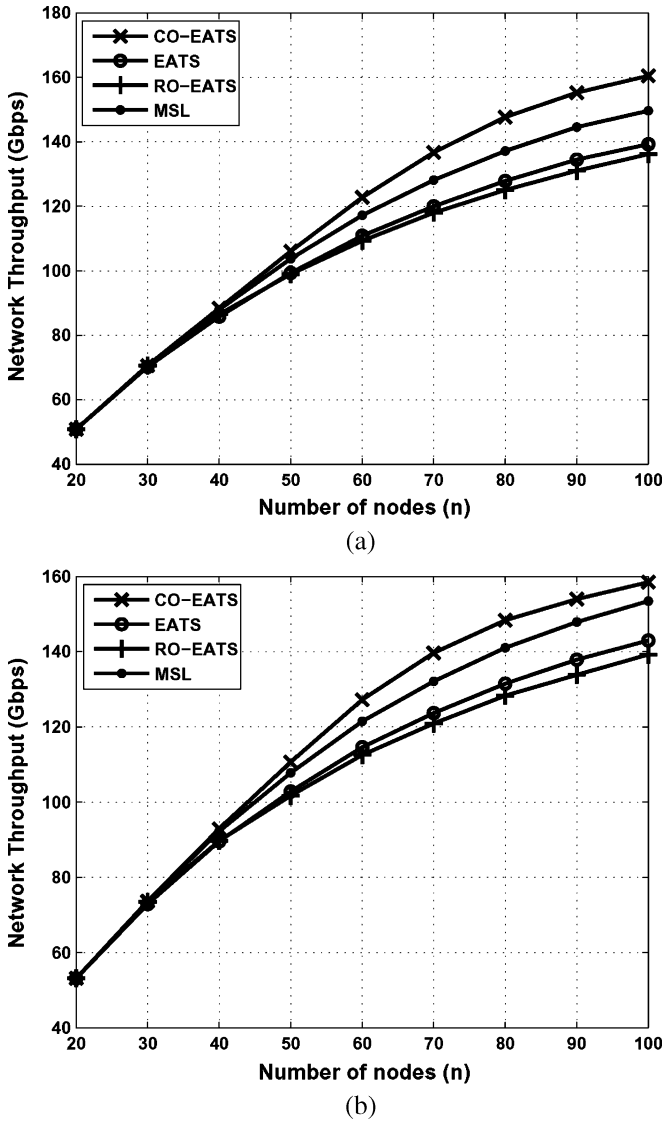


Fig. 5. Network throughput as a function of the number of nodes for $w = 20$ channels, traffic load $k = 30$, and $noc = 20$ clusters. (a) Uniform traffic; (b) Poisson traffic.

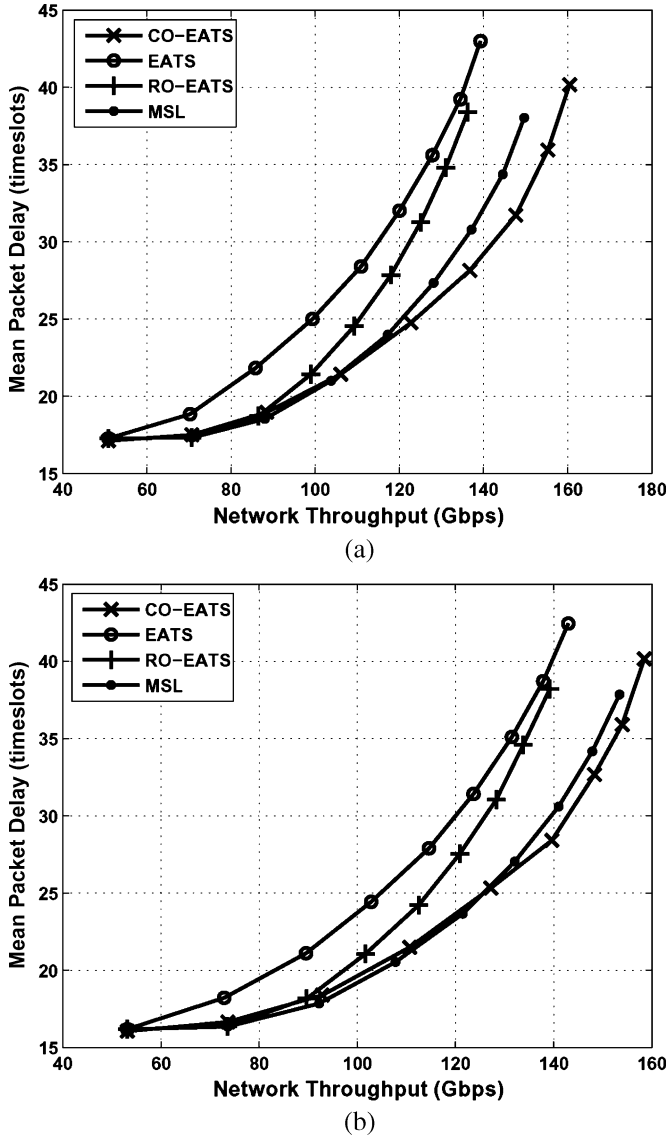


Fig. 6. Mean packet delay as a function of the network throughput for $w = 20$ channels, traffic load $k = 30$, and $noc = 20$ clusters. (a) Uniform traffic; (b) Poisson traffic.

a minimal mean packet delay cost when the number of nodes are increasing. For this part of experiments, we keep the same values of traffic load, i.e., $k = 30$ and number of clusters and channels, i.e., $noc = w = 20$ as previously, while we vary the number of nodes setting $n = 20, 30, \ldots, 100$.

Thus, according to Fig. 6(a) and (b), which represents the mean packet delay as a function of the network's throughput in case of models A and B, respectively, it is obvious that for $n = 20$, 30, 40, all schemes provide almost the same levels of throughput-delay performance. However, for $n \geq 50$, CO-EATS succeeds in obtaining higher throughput while causing either lower or slightly higher delay compared to the rest schemes. Indicatively, in Fig. 6(a), for $n = 80$ nodes, CO-EATS offers 147.7 Gbps throughput causing 32.7 timeslots as mean packet delay, while the respective values for EATS are 127.9 Gbps and 35.6 timeslots, for RO-EATS 125.1 Gbps and 31.3 timeslots and for MSL 137.2 Gbps and 30.8 timeslots. On the other hand, in Fig. 6(b), for $n = 70$ nodes, the proposed CO-EATS with 139.6 Gbps significantly outperforms the EATS with 123.7 Gbps, as well as the RO-EATS with 120.9 Gbps and MSL with 132.1 Gbps. In this case the observed delay values are 29.4, 31.4, 27.5, and 27.0 timeslots for CO-EATS, EATS, RO-EATS and MSL, respectively. In all cases, it is apparent that there is a trade-off between the improved throughput and the minimal delay cost.

Given that the proposed scheme achieves high throughput levels, we evaluated network throughput under different number of channels and the results are illustrated in Fig. 7(a) and (b). For this experimentation, we fixed the number of nodes at $n = 70$, while we set the traffic load to $k = 30$. The number of channels are set to $w = 5$, 10, 15, 20, while the number of clusters are varied accordingly in order that $noc = w$. More specifically, Fig. 7(a) shows the network throughput versus the number of channels under uniform traffic, i.e., model A, while Fig. 7(b) depicts the network throughput versus the number of channels under poisson traffic, i.e., model B. Based on these figures, we

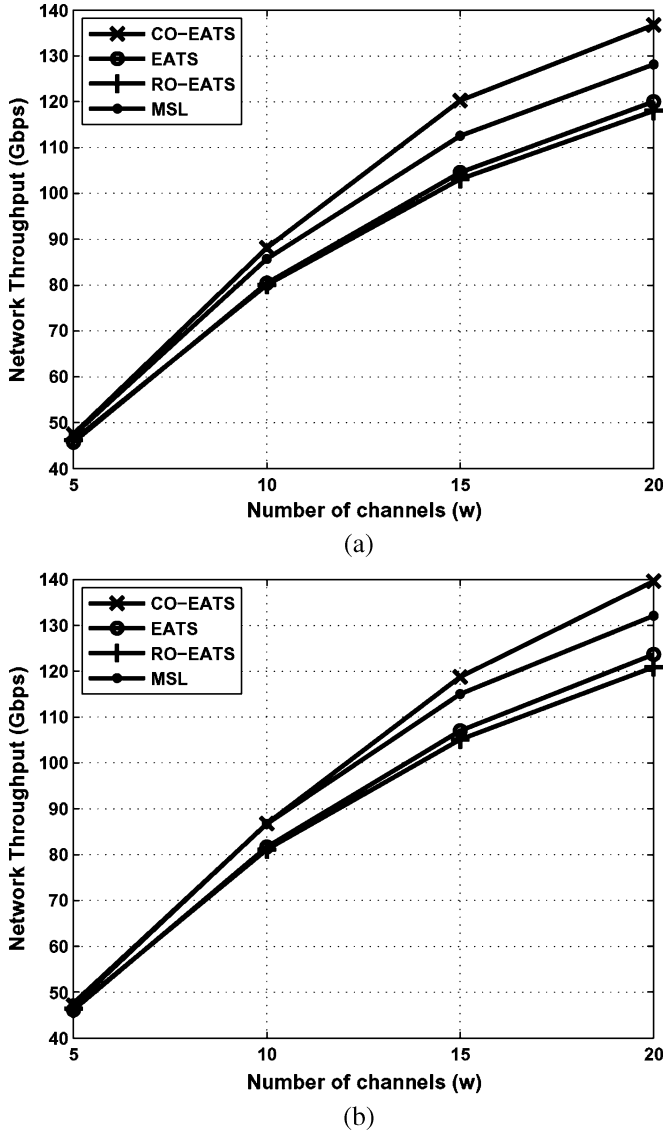


Fig. 7. Network throughput as a function of the number of channels for $n = 70$ nodes, traffic load $k = 30$, and $noc = w$ clusters. (a) Uniform traffic; (b) Poisson traffic.

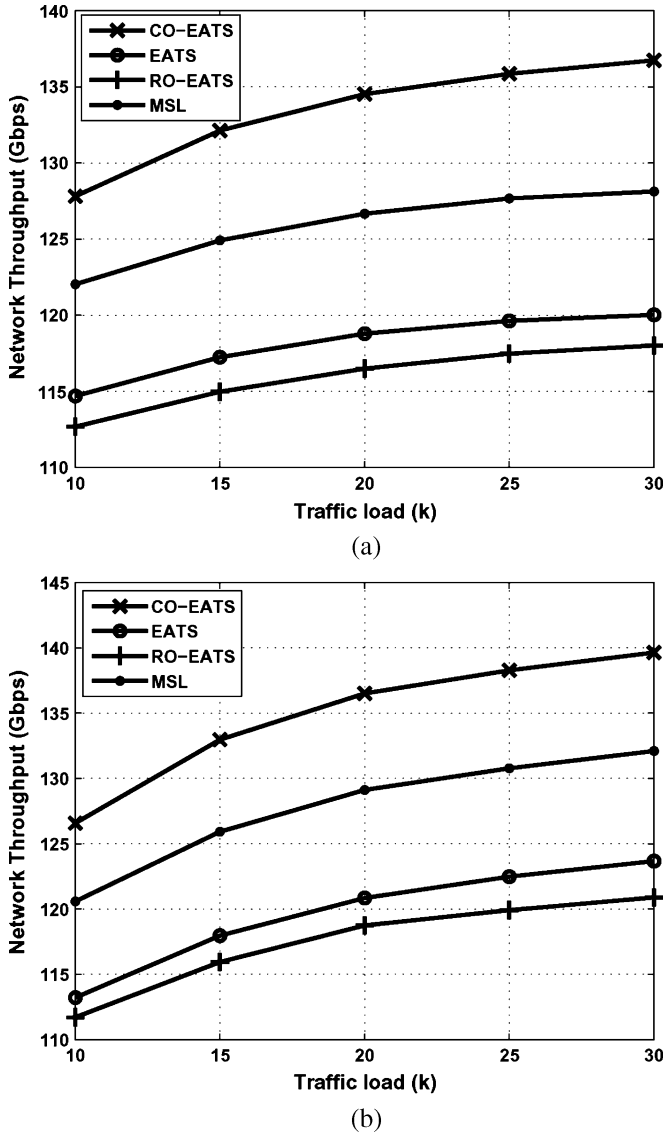


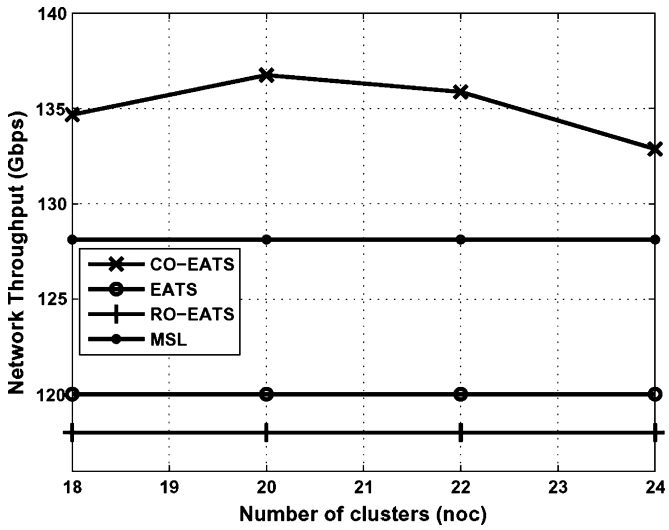
Fig. 8. Network throughput as a function of the traffic load for $n = 70$ nodes, $w = 20$ channels, and $noc = 20$ clusters. (a) Uniform traffic; (b) Poisson traffic.

can see that in case of all algorithms the network throughput is increasing as the number of channels increases and this is natural, since the more network channels the shorter schedule length. In practice, there is more time space for nodes messages to be scheduled. It is remarkable that CO-EATS offers higher throughput in comparison with the rest algorithms for both uniform and poisson traffic, while its performance is getting better as the network channels increase.

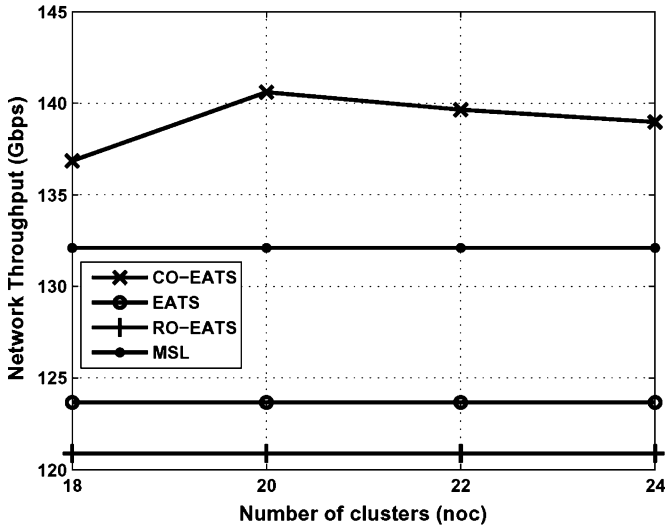
In Fig. 8(a) and (b), the network throughput is presented as a function of the traffic load. As discussed, the traffic load is expressed by the parameter k, which indicates the upper bound of messages' length in packets (or equivalently in timeslots), under both model A and B. In this group of simulation, we carried out experiments with $k = 10$, 15, 20, 25, 30, while we fixed the number of nodes at $n = 70$ and the number of channels and clusters at $noc = w = 20$. The proposed algorithm is presented to be steadily superior in comparison with EATS, RO-EATS, and MSL for both uniform and poisson traffic, while it is important to notice that its behavior is improved as the load is increasing.

Given that the proposed algorithm aims at grouping together nodes with the same message destinations it was challenging to study its performance under different number of clusters. Thus, in the last section of simulation, we evaluated network throughput under different values of noc. In the previous experiments, we define $noc = w$ and we explain that this assumption prevents CO-EATS of scheduling consecutive messages to the same destination. Thus, in Fig. 9(a) and (b), the number of nodes is fixed at $n = 70$, the network load is $k = 30$, while the number of channels was set to $w = 20$. Under these parameters, we varied the number of clusters setting $noc = 18$, 20, 22, 24.

What we can first comment based on Fig. 9(a) and (b) is that the performance of EATS, RO-EATS and MSL is independent of the number of clusters as it is expected. MSL outperforms EATS and RO-EATS, however, all three algorithms present low performance in comparison with CO-EATS both under uniform [Fig. 9(a)] and poisson traffic [Fig. 9(b)]. On the other hand, it is clear that CO-EATS reach its maximum performance for



(a)



(b)

Fig. 9. 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; (b) Poisson traffic.

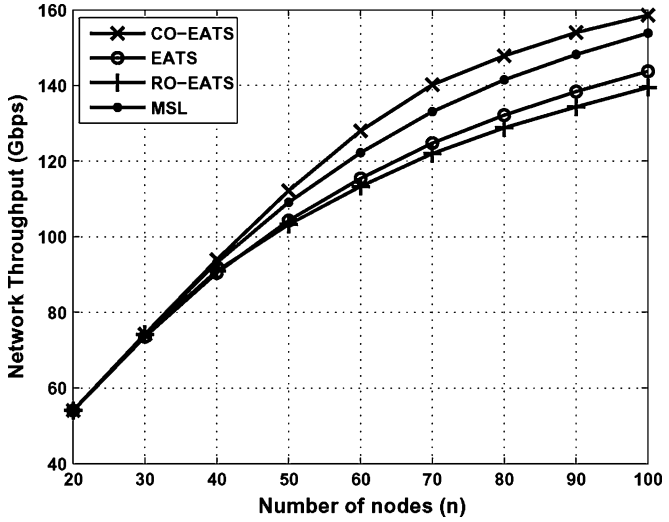


Fig. 10. Self-similar traffic: Network throughput as a function of the number of nodes for $w = 20$ channels and $noc = 20$ clusters.

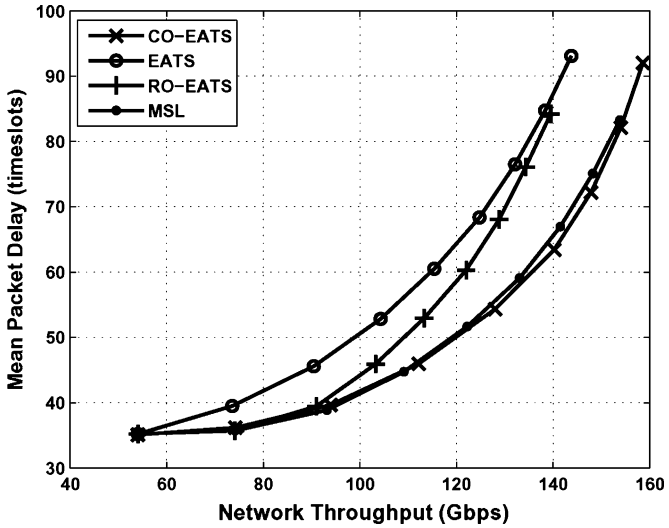


Fig. 11. Self-similar traffic: Mean packet delay as a function of the network throughput for $w = 20$ channels and $noc = 20$ clusters.

$noc = 20$, i.e., for $noc = w$. This observation confirms our intuition about the appropriate value of noc. However, we should notice that even though CO-EATS achieves higher throughput for $noc = w$ it is remarkable that it is superior in comparison with the rest schemes for any value of noc.

C. Experimentation Under Self-Similar Traffic

The performance of the proposed CO-EATS scheme under the self-similar model,i.e., model C, is depicted in Figs. 10–13. As it has been already mentioned, according to model C, the aggregate traffic is generated by multiplexing 16 independent on-off Pareto sources on each network node. Each such source alternates at certain time intervals between an ON state, when a burst of packets is transmitted, and an OFF or idle state. All simulations results obtained defining the Hurst parameter to be equal to 0.9, since it has been shown from statistical tests that real LAN traffic is self-similar with about 0.9 value of Hurst parameter [35].

The results obtained using self-similar model are very similar to the ones obtained under uniform and poisson models. Figs. 10 and 11 confirm the superiority of the proposed scheme, since it is apparent that for any number of network nodes CO-EATS achieves a higher throughput-delay performance compared to EATS and RO-EATS. Although the superiority over MSL is marginal, it should be noticed that MSL runs in time $O(nw^2)$, which is significantly higher compared to $O(nlogn + nw)$ of the CO-EATS. The results of Figs. 12 and 13 stand for different number of channels and clusters, respectively. The performance trends indicate that CO-EATS succeeds in providing higher throughput under self-similar traffic. Especially in case of Fig. 13, it is crucial to pinpoint that CO-EATS reaches its maximum performance for $noc = w$.

The observations about the number of clusters, obtained from Figs. 9(a) and (b) and 13, justifies the choice of the clustering

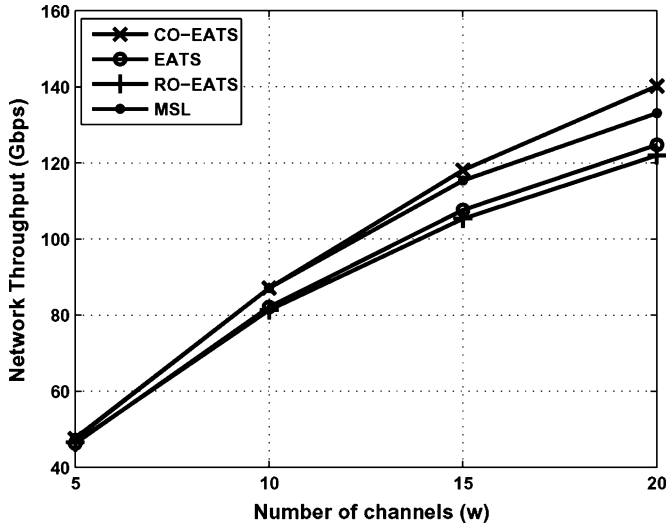


Fig. 12. Self-similar traffic: Network throughput as a function of the number of channels for $n = 70$ nodes and $noc = w$ clusters.

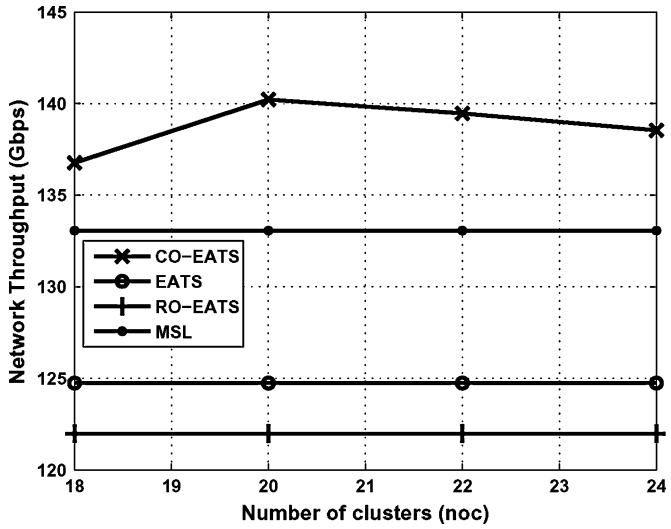


Fig. 13. Self-similar traffic:Network throughput as a function of the number of clusters for $n = 70$ nodes and $w = 20$ channels.

scheme instead of an approach which would simply sort nodes on the basis of their destination. The sorting solution would lead to unknown number of groups, in the worst case to n groups, which would degrade the performance of the proposed scheme, while employing the clustering solution provides the protocol the advantage of fixing *a priori* the desired number of clusters. Overall, in our framework, a clustering scheme provides more information compared to a sorting scheme, since using a combination of clustering and sorting provides groups of sorted nodes. This is achieved with no extra computational cost, since the clustering step has the same complexity with sort procedure, i.e., $O(nlogn)$.

VII. Conclusion

This paper introduces a new class of message scheduling algorithm for WDM star networks, which addresses both the message sequencing and channel assignment issues by making use of clustering techniques. The proposed CO-EATS deals with the message sequencing using a clustering approach which aims at grouping together network's nodes that send their messages to common destination nodes. Based on the produced clusters, the CO-EATS manages to avoid scheduling consecutive messages to the same destination which harms the channels' utilization. The proposed algorithm has been evaluated under uniform, poisson and self-similar traffic for different number of nodes, channels and clusters as well as for different network load. The simulation results clearly show that the proposed scheme considerably upgrades the network performance in comparison with the EATS, RO-EATS, and MSL algorithm. In terms of the mean packet delay, CO-EATS behaves better compared to the EATS without, however, overcoming the performance of RO-EATS and MSL. This is expected since there is a tradeoff between the improved throughput and the minimal delay cost.

The use of clustering algorithms can be the base of a new generation of high performance message scheduling algorithms for optical networks. We are currently working in this direction.

References

[1] J. McDonough, "Moving standards to 100 GbE and beyond," *IEEE Commun. Mag.*, vol. 45, no. 11, pp. 6–9, Nov. 2007.
[2] B. Mukherjee, *Optical WDM Networks*. New York: Springer, 2006.
[3] P. Green, "Progress in optical networking," *IEEE Commun. Mag.*, vol. 39, no. 1, pp. 54–61, Jan. 2001.
[4] G. Papadimitriou, P. Tsimoulas, M. Obaidat, and A. Pomportsis, *Multiwavelength OPTICAL LANs*. New York: Wiley, 2003.
[5] B. Wang, J. C. Hou, and C. C. Han, "Design and analysis of a channel access protocol for WDM-based single-hop optical networks," *Photon. Netw. Commun.*, vol. 6, no. 3, pp. 289–308, 2003.
[6] C. S. R. Murthy and M. Gurusamy, *WDM Optical Networks: Concepts, Design, and Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[7] E. Shekel, A. Feingold, Z. Fradkin, A. Geron, J. Levy, G. Matmon, D. Majer, E. Rafaely, M. Rudman, G. Tidhar, J. Vecht, and S. Ruschin, "64 × 64 fast optical switching module," presented at the Optical Fiber Communication Conf.
[8] G. Papadimitriou, C. Papazoglou, and A. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," *J. Lightw. Technol.*, vol. 21, no. 2, pp. 384–405, Feb. 2003.
[9] G. Bernstein, B. Rajagopalan, and D. Saha, *Optical Network Control: Architecture, Protocols, and Standards*. Reading, MA: Addison Wesley, 2003.
[10] C. S. R. Murthy and M. Gurusamy, *WDM Optical Networks: Concepts, Design, and Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[11] M. Chen, N. Dono, and R. Ramaswami, "A media-access protocol for packet-switched wavelength division multiaccess metropolitan area networks," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 6, pp. 1048–1057, Jun. 1990.
[12] I. Chlamtac and A. Fumagalli, "Quadro-Star: A high performance optical WDM star network," *IEEE Trans. Commun.*, vol. 42, no. 8, pp. 2582–2591, Aug. 1994.
[13] I. Habbab, M. Kavehrad, and C. Sundberg, "Protocols for very high-speed optical fiber local area networks using a passive star topology," *J. Lightw. Technol.*, vol. 5, no. 12, pp. 1782–1794, Dec. 1987.
[14] N. Mehravari, "Performance and protocol improvements for very high speed optical fiber local area networks using a passive star topology," *J. Lightw. Technol.*, vol. 8, no. 4, pp. 520–530, Apr. 1990.
[15] F. Jia, B. Mukherjee, and J. Iness, "Scheduling variable-length messages in a single-hop multichannel local lightwave network," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 477–488, Apr. 1995.
[16] M. Ma, B. Hamidzadeh, and M. Hamdi, "An efficient message scheduling algorithm for WDM lightwave networks," *Comput. Netw.*, vol. 31, no. 20, pp. 2139–2152, 1999.
[17] J. Diao and P. L. Chu, "Packet rescheduling in WDM star networks with real-time service differentiation," *J. Lightw. Technol.*, vol. 19, no. 12, pp. 1818–1828, Dec. 2001.
[18] E. Johnson, M. Mishra, and K. Sivalingam, "Scheduling in optical WDM networks using hidden Markov chain based traffic prediction," *J. Photon. Netw. Commun.*, vol. 3, no. 3, pp. 271–286, 2001.

[19] K. Sivalingam, J. Wang, X. Wu, and M. Mishra, "An Interval-based scheduling algorithm for optical WDM star networks," *J. Lightw. Technol.*, vol. 4, no. 1, pp. 73–87, Jan. 2002.
[20] P. Sarigiannidis, G. Papadimitriou, and A. Pomportsis, "A high throughput scheduling technique, with idle timeslot elimination mechanism," *J. Lightw. Technol.*, vol. 24, no. 12, pp. 4811–4827, Dec. 2006.
[21] P. Sarigiannidis, G. Papadimitriou, and A. Pomportsis, "CS-POSA: A high performance scheduling aalgorithm for WDM star networks," *Photon. Netw. Commun.*, vol. 11, no. 2, pp. 211–227, 2006.
[22] P. Sarigiannidis, G. Papadimitriou, and A. Pomportsis, "LENA: An efficient channel eclectic algorithm for WDM optical networks," *Opt. Laser Technol.*, vol. 40, no. 1, pp. 39–51, 2008.
[23] P. Sarigiannidis, G. Papadimitriou, and A. Pomportsis, "A high performance scheduling priority scheme for WDM star networks," *IEEE Commun. Lett.*, vol. 11, no. 1, pp. 76–78, Nov. 2007.
[24] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
[25] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, "Web usage mining: Discovery and applications of usage patterns from web data," *SIGKDD Explorations*, vol. 1, no. 2, pp. 12–23, 2000.
[26] S. Petridou, V. Koutsonikola, A. Vakali, and G. Papadimitriou, "Time aware web users clustering," *IEEE Trans. Knowl. Data Eng.*, to be published.
[27] J. Goldberger, S. Gordon, and H. Greenspan, "Unsupervised image-set clustering using an information theoretic framework," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 449–458, Feb. 2006.
[28] J. Kasturi, R. Acharya, and M. Ramanathan, "An information theoretic approach for analyzing temporal patterns of gene expression," *Bioinformatics*, vol. 19, no. 4, pp. 449–458, 2003.
[29] M. Albanese, A. Picariello, C. Sansone, and L. Sansone, "Web personalization based on static information and dynamic user behavior," in *Proc. 6th Annu. ACM Int. Workshop WIDM*, Washington DC, Nov. 2004, pp. 80–87.
[30] A. Bianco, G. Mardente, M. Mellia, M. Munafo, and L. Muscariello, "Web user session characterization via clustering techniques," in *Proc. IEEE GLOBECOM*, Dec. 2005, p. 6.
[31] B. Li, M. Ma, and M. Hamdi, *MAC Protocols for WDM Networks: Survey and Summary. Optical Wdm Networks: Principles and Practice*. Boston, MA: Kluwer, 2000.
[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2001.
[33] G. I. Papadimitriou and A. S. Pomportsis, "Adaptive MAC protocols for broadcast networks with bursty traffic," *IEEE Trans. Commun.*, vol. 51, no. 4, pp. 553–557, Apr. 2003.
[34] S. Han, "Architectural and economic impact of the integration of SONET and DWDM platforms," presented at the Optical Fiber Communication Conf., 2006.
[35] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, Jan. 1994.
[36] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet LAN traffic at the source level," *SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 4, pp. 100–113, 1995.
[37] P. Kokoszka and M. Taqqu, "Parameter estimation for infinite variance fractional ARIMA," *Ann. Statist.*, vol. 24, pp. 1880–1913, 1996.
[38] B. B. Mandelbrot, "Long-run linearity, locally Gaussian process, H-spectra and infinite variances," *Int. Econ. Rev.*, vol. 10, no. 1, pp. 82–111, 1969.
[39] B. B. Mandelbrot, *The Fractal Geometry of Nature*. New York: Freedman, 1983.
[40] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *ACMCCR: Comput. Commun. Rev.*, vol. 27, pp. 5–23, 1997.
[41] R. Kapoor, L.-J. Chen, M. Y. Sanadidi, and M. Gerla, "Accuracy of link capacity estimates using passive and active approaches with CapProbe," presented at the 9th Int. Symp. Computers and Communications.

Sophia G. Petridou (M'08) received the B.S. degree in computer science from the Aristotle University of Thessaloniki, Greece, in 2000, where she is currently pursuing the Ph.D. degree in communication networks.

Her research interests include clustering, optical, and wireless networks.

Panagiotis G. Sarigiannidis (S'05–M'07) received the diploma and Ph.D. degrees in computer science from the Aristotle University of Thessaloniki, Greece, in 2001 and 2007, respectively.

He is currently an adjunct Lecturer at the University of Ioannina, Greece. His research interests include optical networks and optical switching.

Georgios I. Papadimitriou (M'89–SM'02) received the diploma and Ph.D. degrees in computer engineering from the University of Patras, Greece, in 1989 and 1994, respectively.

From 1989 to 1994, he was a Teaching Assistant at the Department of Computer Engineering, University of Patras, and a Research Scientist at the Computer Technology Institute, Patras. From 1994 to 1996, he was a Postdoctorate Research Associate at the Computer Technology Institute. In 1997, he joined the faculty of the Department of Informatics, Aristotle University of Thessaloniki, Greece, where he currently serves as an Associate Professor. His main research interests include optical networks and wireless networks. He is the coauthor of the books *Optical Switching* (Wiley, 2007), *Multiwavelength Optical LANs* (Wiley, 2003), and *Wireless Networks* (Wiley, 2003), and co-editor of the book *Applied System Simulation* (Kluwer, 2003). He is the author or coauthor of 150 journal and conference papers.

Prof. Papadimitriou is Associate Editor of the IEEE NETWORK, the *IEEE Communications Magazine*, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: CYBERNETICS, the IEEE TRANSACTIONS ON BROADCASTING, and the IEEE SENSORS JOURNAL.

Andreas S. Pomportsis received the B.S. degree in physics and the M.S. degree in electronics and communications from the University of Thessaloniki, Thessaloniki, Greece, the Diploma in electrical engineering from the Technical University of Thessaloniki, and the Ph.D. degree in computer science from the University of Thessaloniki in 1987.

Currently, he is a Professor at the Department of Informatics, Aristotle University, Thessaloniki. He is the coauthor of the books *Optical Switching* (Wiley, 2007), *Multiwavelength Optical LANs* (Wiley, 2003), and *Optical Networks* (Wiley, 2003). His research interests include computer networks, computer architecture, parallel and distributed computer systems, and multimedia systems.