An Efficient Clustering Oriented Algorithm for Message Scheduling on WDM Star Networks

Sophia G. Petridou, Panagiotis G. Sarigiannidis, Graduate Member, IEEE
Georgios I. Papadimitriou, Senior Member, IEEE, Andreas S. Pomportis
Aristotle University
email:{spetrido, sarpan, gp, apombo}@csd.auth.gr

Abstract — Message sequencing and channel assignment are two important issues that have to be addressed when designing MAC protocols for optical Wavelength Division Multiplexing (WDM) networks. Up to now, popular approaches deal with channel assignment without however addressing the order in which the messages are scheduled. This paper presents a new reservation-based message scheduling algorithm for WDM star networks which is based on clustering techniques. The proposed Clustering Oriented - Earliest Available Time Scheduling (CO-EATS) creates groups of nodes whose messages are destined to common destination nodes. The goal of CO-EATS is to prevent consecutive messages from being destined to the same node. 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.

I. Introduction

Message sequencing and channel assignment are two important issues in designing MAC protocols for optical Wavelength Division Multiplexing (WDM) networks [1]. A well-known, efficient scheduling algorithm for local area WDM networks with broadcast and select architecture is the Earliest Available Time Scheduling (EATS) [2]. 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. This paper introduces a new algorithm that deals with the message sequencing issue based on the clustering [3], [4] 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. Then, given that each cluster will consist of nodes with probably the same destination, the CO-EATS defines the message sequencing choosing for transmission nodes from 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.

The proposed algorithm is inspired by the observation that consecutive messages to the same destination node may not fully use the available channels when the EATS algorithm is employed. Thus, it was necessary to enhance the EATS scheme with an efficient message sequencing mechanism which would distinguish consecutive messages destined to the same node.
RAT handles the above issues and receiver and channel collisions respectively. The earliest available time becomes available. With this global information in each node, the distributed transmission takes place. Each packet is transmitted in time equal to a timeslot. 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. The RAT table consists of elements, where RAT(d$_i$) = $t$, $i$ = 1, $\ldots$, $n$, implies that destination node d$_i$ will be available after $t$ timeslots. The CAT table consists of $w$ elements, where CAT(i) = $t$, $i$ = 1, $\ldots$, $w$, denotes that channel i will be available after $t$ timeslots. 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 [6].

A well-known scheduling algorithm for such a network is the Earliest Available Time Scheduling (EATS) [2]. 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 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 $S$, where $t$ denotes the length on the schedule in timeslots. Each $s(i,j)$ element, $i$ = 1, $\ldots$, $w$ and $j$ = 1, $\ldots$, $t$, represents the destination node that receives a message on channel $d_j$ during the timeslot $l_j$. The time complexity of EATS is $O(nw)$. III. 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, $\ldots$, $n$ indicate 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 $i$th row of the $M$ table will have one non-zero value. On the other hand, the $j$th column of the $M$ table can have more than one non-zero 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 $C_l$ 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 $C_i \cap C_j = \emptyset$ 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.

Thus, the notion of similarity is fundamental in a clustering process, and so far it is quite common to evaluate the dissimilarity between two items (in our case the source nodes) by using a distance measure [3]. To proceed with the clustering of $S$, we employ the Squared Euclidean distance which is a well-known and widely used distance measure in the vector-space model [3], [4]. 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

$$d_E(s_x, s_y)$$

1The 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: $\|x_1 - y_1\|^2$
the Squared Euclidean distance of the nodes’ vectors $M(x,:)$ and $M(y,:)$:

$$d_E(s_k, s_l) = \|M(x,:) - M(y,:))\|^2$$

Consider an arbitrary cluster $C_j$, $j = 1, \cdots, noc$, of the set $S$. The representation of cluster $C_j$ when clustering process 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 $s_k \in C_j$ is represented by $c_j$. Given the vectors of $s_k \in C_j$, the vector of $c_j$ is defined as follows:

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

Since both $M(i,:)$ and $\text{Means}(j,:)$ are vectors, their dissimilarity is measured by their Squared Euclidean distance $d(s_k, c_j)$. 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:

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

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 CI clustering of $S$ into $noc$ clusters such that the $J$ is minimized. A CI that minimizes $J$ groups together nodes from the set $S$ that probably destine their messages to the same nodes of the set $D$.

IV. 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. 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 the same 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, we employ the K-means, a widely used partitioning clustering algorithm [7], in order to produce the CI clustering of $S$. Then, given the CI and the message table $M$, we sort the members on each cluster according to the length of their messages. Similarly, using the Means table, consisting of the clusters representatives’ vectors $\text{Means}(j,:)$, the SortedC is computed in order that we prioritize the clusters with longer messages. The calculated SortedM and SortedC are then used in order that the message sequence will be defined. Once the $\text{MsgSequencing}$ is formed, the algorithm proceeds to the second step called the channel assignment step. The goal of the function $\text{ChannelAssignment}$ is to form the scheduling matrix $S$ 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 $S$.

1: /*Clustering Step*/
2: $\text{CI, Means} = K - \text{means}(M, noc)$
3: SortedM = Quicksort(M, CI)
4: SortedC = Quicksort(Means)
5: $\text{MsgSequencing} = \text{Sequencing}(\text{SortedM, SortedC})$
6: /*Channel Assignment Step*/
7: $S = \text{ChannelAssignment}(\text{MsgSequencing})$

**Theorem 1:** The CO-EATS has time complexity $O(n\log n + mw)$.

**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 [3]. Thus, the CI 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 $O(n\log n + n\log noc + n)$ 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 $O(n\log n + noc\log noc + n)$ which becomes $O(n\log n)$ since $noc$ is relatively small compared to the number of nodes $n$. During the second step, the $\text{ChannelAssignment}$ function (line 6) needs $O(nw)$ time [2] to form the scheduling matrix $S$, where $w$ is the number of channels. As a result, the total complexity of CO-EATS is $O(n\log n + mw)$.

To facilitate the comprehension of the proposed scheme, let us consider a network consisting of the source nodes $s_1, s_2, s_3, s_4, s_5, s_6$, the data channels $(\lambda_1, \lambda_2, \lambda_3)$ and having the upper bound of messages’ length $k = 4$ packets. Then, a $6\times 6$ message table $M$ could be the following:

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

Example 1. In the above message table $M$ the fact that $M(1, 3) = 4$ means that the source node $s_1$ sends a message on length 4 to the destination node $d_3$. Applying the K-means for $noc = 3$ in the above $M$ table
results in $C_{l} = (3, 1, 1, 2, 2, 2)$ which can be represented by the following Members table:

<table>
<thead>
<tr>
<th>C_1</th>
<th>s_2</th>
<th>s_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_2</td>
<td>s_4</td>
<td>s_5</td>
</tr>
<tr>
<td>C_3</td>
<td>s_1</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II**

**The table Members before Members’ sorting**

From Table II, it holds that $s_1 \in C_3$, $s_2, s_3 \in C_1$ while $s_4, s_5, s_6 \in C_2$. It is obvious that $C_{l}$ places to the same cluster similar source nodes in terms of their destination nodes. Then, sorting the members on each cluster according to the length of their message results in swapping the nodes of $C_1$. Therefore, the above table is updated as follows:

<table>
<thead>
<tr>
<th>C_1</th>
<th>s_2</th>
<th>s_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_2</td>
<td>s_4</td>
<td>s_5</td>
</tr>
<tr>
<td>C_3</td>
<td>s_1</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III**

**The table Members after the Members’ sorting**

Given this $C_{l}$ the Means table is:

$$\text{Means} = \begin{pmatrix} 0 & 0 & 0.5 & 1 & 0 & 0 \\ 0 & 1.67 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \end{pmatrix}$$

Sorting Means provides our algorithm with the following service order: $C_3, C_2, C_1$. 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 $s_1, s_4, s_3, s_5, s_2, s_6$ instead of the sequential one $s_1, s_2, s_3, s_4, s_5, s_6$. Tables IV and V depict the scheduling matrix $S$ produced respectively. Based on these tables, the CO-EATS provides 22.2% improvement on channels’ utilization while it reduces the mean packet delay from 3 to 2.2 timeslots.

<table>
<thead>
<tr>
<th>Timeslots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1$</td>
</tr>
<tr>
<td>$w_1$</td>
</tr>
<tr>
<td>$w_2$</td>
</tr>
<tr>
<td>$w_3$</td>
</tr>
</tbody>
</table>

**TABLE IV**

**The scheduling matrix $S$ produced by CO-EATS**

V. EXPERIMENTATION

To evaluate the proposed CO-EATS in comparison with the EATS algorithm we carried out experiments which are based on 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 message transmitted by a node can be destined to every other node with equal probability.

3) Nodes may send messages of 0 to $k$ length on each frame following uniform distribution.

4) The line is defined at 3 Gbps per channel.

5) The outcome results from 10000 transmission frames.

The performance of CO-EATS is compared to that of EATS in terms of the network throughput and mean packet delay. Network throughput represents the average number of bits transmitted per frame on each channel while mean packet delay denotes the mean time that messages are waiting to transmit. Fig. 2(a) and 3(a) depict the network’s throughput as a function of the number of nodes for $n = 10, 20, 30, 40, 50, 60$ and $k = 10$ while the number of channels and clusters is $w = 5, noc = 5$ and $w = 10, noc = 10$ respectively. 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 which they probably have different destinations. It is apparent that
CO-EATS provides steadily higher throughput than EATS both for \( w = 5 \) and \( w = 10 \). Their maximum observed difference is 2.38 Gbps for \( n = 40 \) and \( w = 10 \) while for \( n = 10 \) and \( w = 10 \) the minimum difference is expected since each node destines its message to different channel and thus the contribution of clustering is of no value. In Fig. 2(b) and 3(b) we can observe that the improvement in network’s throughput does not affect the mean packet delay. More specifically, the CO-EATS keeps lower the mean packet delay in comparison with EATS independently of the number of channels while obtains higher throughput. For example, for \( n = 30 \) and \( w = 10 \) CO-EATS offers 19.19 Gbps throughput causing 8 timeslots as mean packet delay while the respective values for EATS are 17.42 Gbps and 8.

VI. Conclusions and Future Work

This paper introduces and evaluates a novel message scheduling algorithm for WDM star networks which address both the message sequencing and channel assignment issues. The proposed Clustering Oriented - Earliest Available Time Scheduling (CO-EATS) deals with the message sequencing using a clustering approach which aims at grouping together network’s nodes that sent 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 traffic for different number of nodes and channels and it has resulted in significantly upgrading the network performance while keeping low the mean packet delay in comparison with the EATS.

Future work aims at studying the simulation results using different values of propagation delay as well as evaluating the proposed scheme under poisson traffic. Furthermore, we will compare the proposed scheme with other message scheduling algorithms which also address the message sequencing issue.

References