Dynamic Wavelength Assignment for WDM All-Optical Tree Networks

Poompat Saengudomlert, *Member, IEEE*, Eytan H. Modiano, *Senior Member, IEEE*, and Robert G. Gallager, *Life Fellow, IEEE*

***Abstract*—We develop an on-line wavelength assignment (WA) algorithm for a wavelength-routed WDM tree network. The algorithm dynamically supports all k-port traffic matrices among N end nodes, where k denotes an integer vector $[k_1 \ldots, k_N]$ and end node i, $1 \leq i \leq N$, can transmit at most k_i wavelengths and receive at most k_i wavelengths. Our algorithm is rearrangeably nonblocking, uses the minimum number of wavelengths, and requires at most $d^* - 1$ lightpath rearrangements per new session request, where d^* is the degree of the most heavily used node. We observe that the number of lightpath rearrangements per new session request does not increase as the amount of traffic k scales up by an integer factor. In addition, wavelength converters cannot reduce the number of wavelengths required to support k-port traffic in a tree network. We show how to implement our WA algorithm using a hybrid wavelength-routed/broadcast tree with only one switching node connecting several passive broadcast subtrees. Finally, using roughly twice the minimum number of wavelengths for a rearrangeably nonblocking WA algorithm, we can modify the WA algorithm to be strict-sense nonblocking.**

***Index Terms*—Graph theory, network reconfiguration, optical networks, resource management, wavelength assignment.**

I. Introduction

WAVELENGTH division multiplexing (WDM) technologies can provide the increase in network capacity to meet the growing traffic demands in optical networks. In a WDM network, the fiber bandwidth is divided into multiple frequency bands often called wavelengths. Using reconfigurable optical switches at the network nodes, some wavelengths can be selected at each node for termination and electronic processing, and others selected for optical bypass. In an *all-optical network* architecture, each traffic session optically bypasses electronic processing at each node on its path other than the source node and the destination node. One important benefit of this architecture is the cost saving resulting from using fewer and/or smaller elctronic switches in the network. We focus our attention on all-optical networks in this paper.

Without optical wavelength conversion, routing of traffic sessions is subjected to the *wavelength continuity constraint*, which dictates that the lightpath corresponding to a given session must travel on the same wavelength on all links from the source node to the destination node. Using wavelength converters potentially allows the network to support a larger set of traffic. However, such converters are likely to be expensive. Hence, we focus on the problem of routing and wavelength assignment (RWA) without wavelength converters.

A. RWA in WDM Networks

The RWA problem is an important problem in resource management for WDM networks. Surveys on the subject are available in [1]–[3]. We can categorize known results into two groups based on whether static or dynamic provisioning of routes and wavelengths is performed. For static provisioning, the traffic to be supported is assumed known and fixed over time. The goal is often to minimize the number of wavelengths used in the network [4], [5], or to maximize the number of supported traffic sessions for a fixed number of wavelengths [6]–[9]. These problems are known to be NP-complete [6]. Consequently, bounds on the optimal costs have been derived [7], [10], and several RWA heuristics have been developed [4], [7]–[9], [11], [12].

Dynamic provisioning of routes and wavelengths gives us flexibility in supporting traffic which may change over time through session arrivals and session departures. Dynamic provisioning is appropriate for bandwidth-on-demand services that requires temporary instead of permanent connections. To model dynamic traffic, one can assume that session arrivals form stochastic processes [13], [14]. In addition, session lifetimes are probabilistic. The goal is usually to develop an on-line RWA algorithm which minimizes the average blocking probability for a new session request given a fixed number of wavelengths in the network. This type of problem formulation is typically refered to as the *blocking* formulation. Due to the complexity in computing blocking probabilities, approximations are made to simplify the analysis. For example, session arrivals on different links are assumed to be independent [13], [15], or correlated among adjacent links in the same fashion throughout the network [14]. Based on such approximations, several dynamic RWA heuristics have been developed [16], [17].

The above blocking formulation of the RWA problem is appropriate for traffic whose frequency of changes is in the time scale of seconds to minutes, e.g., file transfers among individual users. For traffic whose frequency of changes is in the time scale of several minutes to hours or days, e.g., corporate data transfers and temporary lease lines, an alternative formulation is more meaningful. In the *nonblocking* formulation, we assume prior knowledge of the set of all the traffic matrices, or equivalently the traffic demands, to be supported. In [18], the set of traffic

Manuscript received May 25, 2004; revised September 26, 2004; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Somani.

P. Saengudomlert was with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with the Asian Institute of Technology, Pathumthani 12120, Thailand (e-mail: poompats@ait.ac.th).

E. H. Modiano and R. G. Gallager are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Digital Object Identifier 10.1109/TNET.2005.852875

matrices is characterized by the maximum link load in the network. In [19]–[22], the set of traffic matrices is characterized by the numbers of tunable transmitters and tunable receivers at each end node, i.e., a node that sources and/or sinks traffic sessions. We adopt the same traffic model in this paper. A new session is said to be *admissible* if its arrival results in a traffic matrix which is still in the set of supportable traffic, i.e., there is a free transmitter at the source and a free receiver at the destination. The goal is to develop an on-line RWA algorithm which does not block any admissible session and uses the minimum number of wavelengths.

If we allow some existing lightpaths to be rearranged in order to support a new session, the corresponding RWA algorithm is said to be *rearrangeably nonblocking*. If a new session can be supported without lightpath rearrangement, the RWA algorithm is said to be *wide-sense nonblocking*. If a new session can always be supported on *any* available route and wavelength without lightpath rearrangement, the RWA algorithm is said to be *strict-sense nonblocking*. Note that if a RWA algorithm is strict-sense nonblocking, it is also wide-sense nonblocking. If a RWA algorithm is wide-sense nonblocking, it is also rearrangeably nonblocking. Thus, for the same set of traffic matrices, the required number of wavelengths for a strict-sense nonblocking algorithm is no smaller than that for a wide-sense nonblocking algorithm which is no smaller than that for a rearrangeably nonblocking algorithm.

B. Summary of Results

In this paper, we present an on-line RWA algorithm for tree networks that is rearrangeably nonblocking and requires few lightpath rearrangements for each traffic change. We adopt the same problem formulation as in [19]–[22]. Our contribution is to provide a complete analytical solution for tree networks. In a related work, [23] provides a solution for *oriented* trees in which each edge is directed in exactly one way. Since there is no routing problem in a tree network, our RWA algorithm only has to perform wavelength assignment (WA) and will be referred to as the tree WA algorithm.

Tree topologies are interesting for two reasons. First, in the distribution parts of optical access networks, tree topologies are often chosen for their ease of implementation and low costs [24]. Second, since a tree topology can be embedded in any connected topology, the analytical approaches developed in this paper may be useful in the development of a RWA algorithm for mesh networks.

We assume that each session takes up a full wavelength, and each traffic change results from either a session arrival or a session departure. The set of supportable traffic matrices is defined by the number of tunable transmitters and receivers at the end nodes. The tree WA algorithm has the following key properties.

1) It uses the minimum number of wavelengths.
2) No reduction in the number of wavelengths can be obtained from wavelength conversion.
3) The number of lightpath rearrangements for each virtual topology reconfiguration is bounded by the degree of the most heavily used node.

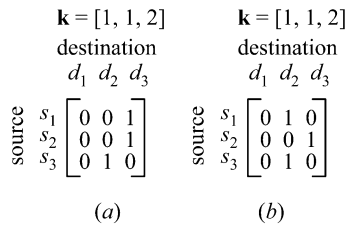

Fig. 1. Traffic matrices in and not in the $\mathbf{k}$-port traffic set. (a) Traffic matrix in the $\mathbf{k}$-port traffic set. (b) Traffic matrix not in the $\mathbf{k}$-port traffic set.

4) The number of lightpath rearrangements remains the same as the traffic scales up by an integer factor.
5) With roughly twice the number of wavelengths, we can make the algorithm wide-sense nonblocking, i.e., requiring no lightpath rearrangement.
6) It can be implemented on a hybrid wavelength-routed/broadcast tree using only one switching node to connect passive broadcast subtrees.

Property 4 is unique to our study, and is different from the observation obtained in a previous study on reconfiguration [25] that the number of rearrangements grows linearly with both the number of users and the amount of traffic for a mesh topology. Property 6 indicates that a hybrid tree can be constructed by partially upgrading an existing passive broadcast tree.

In Section II, we define the set of $\mathbf{k}$-port traffic and formulate the WA problem in a tree network. In Section III, we describe our on-line WA algorithm and prove its correctness. Section IV discusses a hybrid wavelength-routed/broadcast tree that uses only one switching node. Finally, we summarize the paper in Section V.

II. Problem Formulation

Consider a WDM all-optical tree network with no wavelength conversion. Adjacent nodes are connected by two unidirectional fibers, one in each direction. In addition, all fibers contain the same number of wavelengths. We assume there are N end nodes, which are the leaf nodes of the tree. (Later on, we shall show how the restriction on each end node being a leaf node can be relaxed in most networks.) Assume that each traffic session takes up one wavelength. At a given time, only one session can use a specific wavelength in a fiber, but multiple sessions can use the same node. Leaf node i, $1 \leq i \leq N$, is equipped with k_i *fully tunable* transmitters and k_i *fully tunable* receivers. Consequently, at any time, node i can transmit at most k_i wavelengths and receive at most k_i wavelengths. Such a traffic matrix is said to belong to a set of $\mathbf{k}$-*port traffic*, where $\mathbf{k} = [k_1, k_2, \ldots, k_N]$. Fig. 1 shows example traffic matrices in and not in the $\mathbf{k}$-port traffic set. In this example, $N = 3$ and $\mathbf{k} = [1, 1, 2]$. Notice that the traffic matrix in Fig. 1(b) is not in the $\mathbf{k}$-port traffic set because $k_2 = 1$ but leaf node 2 receives more than one wavelength.

We make one assumption on $\mathbf{k}$-port traffic.

Assumption 1: Let $k_{\max} = \max_{1 \leq i \leq N} k_i$. Assume that $k_{\max} \leq (\sum_{1 \leq i \leq N} k_i)/2$.

Assumption 1 is reasonable since the node with $k_{\max}$ tunable transmitters (receivers) can transmit (receive) at most $(\sum_{1 \leq i \leq N} k_i) - k_{\max}$ wavelengths to (from) all the other nodes.

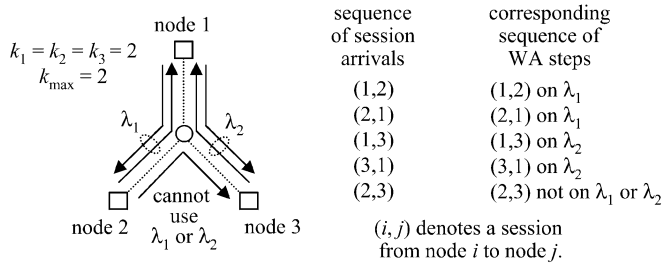

Fig. 2. Example in which a greedy approach requires more than $k_{\max}$ wavelengths.

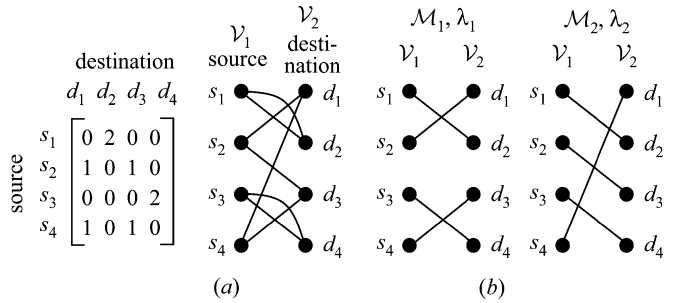

Fig. 3. Traffic bipartite graph and its matchings. (a) Traffic matrix and its traffic bipartite graph. (b) Bipartite matchings M and M_2, assigned to wavelengths λ_1 and λ_2.

Therefore, $k_{\max}$ need be no greater than $(\sum_{1\le i\le N} k_i) - k_{\max}$, yielding the condition in assumption 1.

We model dynamic traffic as a session-by-session arrival and departure process in which sessions arrive and depart one at a time. In other words, a transition from one traffic matrix to another is a result of either a single session arrival or a single session departure. A new session request is admissible if the resultant traffic matrix is still in the set of $\mathbf{k}$-port traffic. The definition implies that, for each new admissible session request, there is a free transmitter at the source node and a free receiver at the destination node. For convenience, throughout the paper, a new session is assumed to be admissible unless explicitly stated otherwise.

We want to design an on-line WA algorithm which supports $\mathbf{k}$-port traffic in a rearrangeably nonblocking fashion, uses the minimum number of wavelengths, and requires few rearrangements of existing lightpaths in order to support each new session request. Our algorithm will be centralized in nature. We assume that the WA algorithm always has correct knowledge of the current WA in the network. In addition, we focus on traffic that changes in the time scale of several minutes to hours or days, and assume there is sufficient time for lightpath rearrangements between successive transitions of the traffic matrix.

III. On-Line WA Algorithm

In this section, we present our on-line WA algorithm for $\mathbf{k}$-port traffic in a tree network. For clarity, we first present an algorithm for star networks, and then extend the results to tree networks.

A. Star Networks

Fig. 2 shows an example of a star network with three leaf nodes connected through a central hub.

Let $L_{\mathbf{k}}$ and $W_{\mathbf{k}}$ denote the minimum number of wavelengths which, if provided in each fiber, can support $\mathbf{k}$-port traffic with full wavelength conversion at all nodes and without wavelength conversion respectively. It is clear that $L_{\mathbf{k}} \le W_{\mathbf{k}}$. Notice that $L_{\mathbf{k}}$ and $W_{\mathbf{k}}$ are the number of wavelengths required to support *any* traffic matrix in the $\mathbf{k}$-port set. Thus, for a specific traffic matrix, we may need fewer wavelengths than what we need in the worst case. To derive $L_{\mathbf{k}}$, consider the fiber from the node with traffic parameter $k_{\max}$ to the hub node. This fiber must support up to $k_{\max}$ wavelengths, which is the maximum link load. It follows that $L_{\mathbf{k}} = k_{\max}$.

We shall show that $W_{\mathbf{k}} \le k_{\max}$, which implies $W_{\mathbf{k}} = L_{\mathbf{k}} = k_{\max}$. We do so by constructing an on-line WA algorithm. Fig. 2 illustrates an example scenario in which an on-line greedy WA algorithm fails to support an instance of $\mathbf{k}$-port traffic using $k_{\max}$ wavelengths. In this example, $N = 3$, $\mathbf{k} = [2, 2, 2]$, and the traffic matrix to be supported is uniform all-to-all traffic, i.e., each node sends one wavelength to each of the other two nodes. As shown in Fig. 2, the same wavelength is assigned to the oppositely directed sessions between the same pair of nodes, e.g., sessions (1, 2) and (2, 1) on wavelength λ_1. Notice that, after assigning wavelengths λ_1 and λ_2 to sessions (1, 2), (2, 1), (1, 3), and (3, 1), neither λ_1 nor λ_2 can be assigned to support session (2, 3). It follows that more than $k_{\max} = 2$ wavelengths are required. Therefore, this example scenario tells us that the WA algorithm design is not trivial. Fig. 2 also demonstrates that, to use the minimum number of wavelengths, we may need to support the oppositely directed sessions between the same pair of nodes on different wavelengths.

Our algorithm is based on bipartite matchings. For a given traffic matrix, we construct the *traffic bipartite graph*, denoted by $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, as follows. We consider each leaf node as a distinct source node and destination node. The set of nodes $\mathcal{V}_1$ contains the N source nodes. The set of nodes $\mathcal{V}_2$ contains the N destination nodes. In the set of edges $\mathcal{E}$, an edge between node i in $\mathcal{V}_1$ and node j in $\mathcal{V}_2$ exists for each traffic session from source i to destination j. Fig. 3(a) shows an example of the traffic bipartite graph and its traffic matrix. Note that there may be multiple edges between the same pair of nodes. For example, since there are two sessions from source 1 to destination 2, there are two parallel edges between s_1 in $\mathcal{V}_1$ and d_2 in $\mathcal{V}_2$ in Fig. 3(a).

A matching in a bipartite graph, or in short a bipartite matching, is a subset $\mathcal{M}$ of $\mathcal{E}$ such that no two edges in $\mathcal{M}$ are adjacent. Fig. 3(b) shows two disjoint bipartite matchings, denoted by $\mathcal{M}_1$ and $\mathcal{M}_2$, obtained from $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ in Fig. 3(a).

Observe that the sessions in a bipartite matching can be supported on a single wavelength without wavelength collision. To see this, note that, in a matching, at most one edge is incident on each source (destination) node. Thus, in each fiber to (from) the hub node, every wavelength is used at most once. Our algorithm will assign a single bipartite matching to a single wavelength. In what follows, we shall refer to the matching in the traffic bipartite graph which is assigned to wavelength λ_1 simply as the bipartite matching of λ_1. Fig. 3(b) shows example bipartite matchings of specific wavelengths. We next state a known useful lemma related to bipartite graphs [26].

Lemma 1: In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with maximum node degree m, we can color the edges in $\mathcal{E}$ so that no two adjacent edges have the same color using m colors.

Consider coloring the edges in a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ as suggested by Lemma 1. Since no two adjacent edges have the same color, the edges with the same color form a bipartite matching. Thus, we can restate Lemma 1 as follows.

Lemma 2: In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with maximum node degree m, the set $\mathcal{E}$ can be partitioned into m disjoint bipartite matchings.

Lemma 2 can be used to argue that $k_{\max}$ wavelengths are sufficient to support any traffic matrix in the $\mathbf{k}$-port set. Given a traffic matrix, we can write down the corresponding traffic bipartite graph in which each node has degree at most $k_{\max}$. By Lemma 2, the set of edges can be partitioned into $k_{\max}$ disjoint bipartite matchings. The sessions in each matching can be supported on a single wavelength. Thus, $k_{\max}$ wavelengths are sufficient to support any $\mathbf{k}$-port traffic matrix.

The main idea of our on-line WA algorithm involves keeping $k_{\max}$ disjoint bipartite matchings of $k_{\max}$ wavelengths such that each traffic session corresponds to an edge in one bipartite matching. When a session departs, we simply remove its corresponding lightpath from the network. When a new session, say (i, j), arrives, we find one wavelength that is not used by source i, and one wavelength that is not used by destination j. If the two wavelengths are the same, we can support the new session without any lightpath rearrangement. Otherwise, we rearrange some existing lightpaths on the two wavelengths to support the new session. The following lemma makes the above discussion concrete and states an upper bound on the number of lightpath rearrangements.

Lemma 3: In a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with $|\mathcal{V}_1| = |\mathcal{V}_2| = V$, given a new edge $(s_i, d_j), s_i \in \mathcal{V}_1, d_j \in \mathcal{V}_2$, a matching $\mathcal{M}_1$ of wavelength λ_1 which is not incident on s_i, and a matching $\mathcal{M}_2$ of wavelength λ_2 which is not incident on d_j, there exist two disjoint bipartite matchings which cover all the edges in $\mathcal{M}_1$ and $\mathcal{M}_2$ as well as the new edge (s_i, d_j).

In addition, these two disjoint bipartite matchings can be assigned to λ_1 and λ_2 so that the number of lightpath rearrangements is at most $V - 1$.

Proof: Consider the bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$ whose set of edges $\mathcal{E}'$ contains all of the edges in $\mathcal{M}_1$ and $\mathcal{M}_2$ as well as the new edge (s_i, d_j). Observe that each node has degree at most 2. From Lemma 2 with $m = 2$, there exist two disjoint bipartite matchings, denoted by $\mathcal{M}'_1$ and $\mathcal{M}'_2$, which cover all the edges.

Without loss of generality, assume that (s_i, d_j) belongs to $\mathcal{M}'_1$. Let set $\mathcal{P}$ contain the edges in $\mathcal{M}_1$ assigned to $\mathcal{M}'_2$ and the edges in $\mathcal{M}_2$ assigned to $\mathcal{M}'_1$. Let set $\mathcal{Q}$ contain the edges in $\mathcal{M}_1$ assigned to $\mathcal{M}'_1$ and the edges in $\mathcal{M}_2$ assigned to $\mathcal{M}'_2$. Notice that $\mathcal{P}$ and $\mathcal{Q}$ cover all the edges in $\mathcal{M}_1$ and $\mathcal{M}_2$. Since there are at most $2V - 2$ edges in $\mathcal{M}_1$ and $\mathcal{M}_2$, it follows that $|\mathcal{P}| + |\mathcal{Q}| \leq 2V - 2$.

If $|\mathcal{P}| \leq V - 1$, assigning $\mathcal{M}'_1$ to λ_1 and $\mathcal{M}'_2$ to λ_2 yields the desired result that the number of lightpath rearrangements, which is equal to the sum of the number of edges in $\mathcal{M}_1$ assigned to $\mathcal{M}'_2$ and the number of edges in $\mathcal{M}_2$ assigned to $\mathcal{M}'_1$, is at most $V - 1$. Otherwise, it is true that $|\mathcal{Q}| \leq V - 1$. In this case, assigning $\mathcal{M}'_1$ to λ_2 and $\mathcal{M}'_2$ to λ_1 yields the desired result. □

Note that $V = N$ for a N-node star. The best known algorithm for finding a maximum bipartite matching in [27] has the running time $O(\Delta E)$, where Δ is the maximum node degree and E is the number of edges in the bipartite graph.[1] For a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$ in the proof of Lemma 3, $\Delta = 2$ and $E \leq 2V$, where $V = |\mathcal{V}_1| = |\mathcal{V}_2|$. Thus, to find two disjoint matchings for each WA update, the running time of the algorithm in [27] is $O(V)$. In the Appendix, we provide an alternative and simpler procedure specialized for our task with the same running time $O(V)$.

The following is our on-line WA algorithm for a star network with $\mathbf{k}$-port traffic which uses $k_{\max}$ wavelengths in each fiber, is rearrangeably nonblocking, and requires at most $N-1$ lightpath rearrangements per new session request. We shall refer to this algorithm as the *star WA algorithm.*

Star WA algorithm: (Use $k_{\max}$ wavelengths.)

Session termination: When a session terminates, simply remove its associated lightpath from the network without any futher lightpath rearrangement.

Session arrival: When a new session arrives and the resultant traffic matrix is still $\mathbf{k}$-port, proceed as follows. Assume that the new session is from source i to destination j.

Step 1: If there is a wavelength, denoted by λ_0, which is used by neither source i nor destination j, then assign the new session to λ_0. In this case, no lightpath rearrangement is required. Otherwise, proceed to step 2.

Step 2: Find a wavelength, denoted by λ_1, which is not used by source i, i.e., its bipartite matching is not incident on s_i, and another wavelength, denoted by λ_2, which is not used by destination j, i.e., its bipartite matching is not incident on d_j. Since the new session is admissible, there are at most $k_{\max} - 1$ sessions from source i. Since there are $k_{\max}$ available wavelengths, it follows that λ_1 exists. By the same argument, λ_2 always exists.

Modify the WA of only the sessions on λ_1 and λ_2. Contruct the traffic bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$ in which the set of edges $\mathcal{E}'$ contains the bipartite matchings of λ_1 and λ_2 as well as the new edge (s_i, d_j). From Lemma 3, we can partition the set $\mathcal{E}'$ into two disjoint bipartite matchings. In addition, since $|\mathcal{V}_1| = |\mathcal{V}_2| = N$, Lemma 3 tells us that the two matchings can be assigned to λ_1 and λ_2 such that at most $N-1$ existing lightpaths need to be rearranged.

The construction of the star WA algorithm implies the following theorem.

Theorem 1: For the star network with N nodes and $\mathbf{k}$-port traffic, $W_{\mathbf{k}}$ is given by

$$W_{\mathbf{k}} = L_{\mathbf{k}} = k_{\max} = \max_{1 \leq i \leq N} k_i.$$

In addition, there exists, by construction, an on-line WA algorithm which uses $k_{\max}$ wavelengths in each fiber and requires at most $N - 1$ lightpath rearrangements per new session request.

[1] By running time $O(g(n))$, we mean the running time can be expressed as a function $f(n)$ of the problem size n such that there exist a positive real constant c and a positive integer n_0 satisfying $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

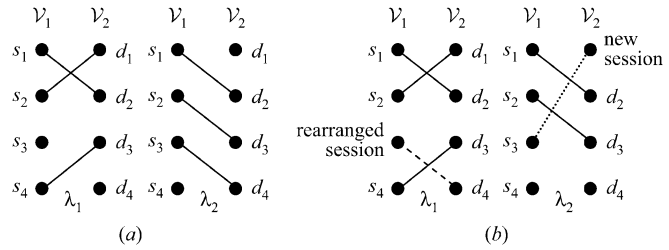

Fig. 4. Example operations of the star WA algorithm. (a) Bipartite matchings of λ_1 and λ_2. (b) Updated bipartite matchings of λ_1 and λ_2.

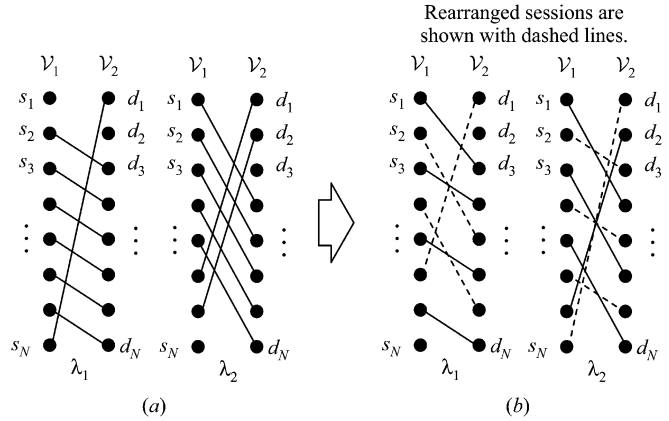

Fig. 5. Example case in which $N-1$ lightpath rearrangements are made to support a new session.

The following example illustrates the operations of the star WA algorithm.

Example 1: Consider a 4-node star network with the traffic matrix given in Fig. 3(a). Note that $W_{\mathbf{k}} = 2$. Assume that the WA is given by the two bipartite matchings of wavelengths λ_1 and λ_2 as shown in Fig. 3(b). Now assume the following changes in the traffic matrix.

1) Existing session (3, 4) on λ_1 terminates.
2) Existing session (4, 1) on λ_2 terminates.
3) A new session (3, 1) arrives.

After the second session termination, the bipartite matchings of λ_1 and λ_2 are shown in Fig. 4(a). To support the new session, the star WA algorithm performs step 2. In particular, it creates a traffic bipartite graph whose edges are the bipartite matchings of λ_1 and λ_2 as well as the new edge (s_3, d_1). The algorithm then partitions the set of edges into two disjoint bipartite matchings and assigns them to λ_1 and λ_2, as shown in Fig. 4(b). In particular, session (3, 4) on λ_2 is reassigned to λ_1, and the new session is then assigned to λ_2. In this example, one rearrangement of an existing lightpath is made to support the new session.

The following example demonstrates that the star WA algorithm may perform up to $N-1$ lightpath rearrangements to support a new session request.

Example 2: Consider the following WA scenario. Assume that each wavelength supports one of the two bipartite matchings shown in Fig. 5(a). Suppose the new session is transmitted from source 1 to destination 3. In this case, the star WA algorithm needs to perform step 2. After choosing two bipartite matchings of wavelengths λ_1 and λ_2, as shown in Fig. 5(a), the algorithm creates a traffic bipartite graph whose edges are all the edges in the bipartite matchings of λ_1 and λ_2 as well as the new edge (s_1, d_3). The algorithm then partitions the set of edges into two disjoint bipartite matchings and assign them to λ_1 and λ_2, as shown in Fig. 5(b). In this example, the algorithm needs to perform $N-1$ lightpath rearrangements to support the new session.

B. Arbitrary Tree Networks

In this section, we extend the star WA algorithm to create an on-line WA algorithm for tree networks. In a given tree network, assume there are $N > 2$ end nodes which are the leaf nodes of the tree. We shall ignore all the nonleaf nodes with degree 2 since their removal does not change the WA problem. We describe a tree by a set of nodes $\mathcal{N}$ and a set of bidirectional links $\mathcal{T}$.

We first determine $L_{\mathbf{k}}$, the minimum number of wavelengths which, if provided in each fiber, can support $\mathbf{k}$-port traffic given full wavelength conversion at all nodes. Each link e in the tree corresponds to a cut which separates N leaf nodes into two sets, denoted by $\mathcal{N}_{e,1}$ and $\mathcal{N}_{e,2}$. The maximum possible traffic, in wavelength units, in a fiber across this link is equal to $\min(\sum_{i\in\mathcal{N}_{e,1}} k_i, \sum_{i\in\mathcal{N}_{e,2}} k_i)$. The overall maximum possible traffic across any link, denoted by w^*, is the value of $L_{\mathbf{k}}$ given below:

$$L_{\mathbf{k}} = w^* = \max_{e\in\mathcal{T}} \min\left(\sum_{i\in\mathcal{N}_{e,1}} k_i, \sum_{i\in\mathcal{N}_{e,2}} k_i\right). \qquad (1)$$

Let $W_{\mathbf{k}}$ be the minimum number of wavelengths which, if provided in each fiber, can support $\mathbf{k}$-port traffic with no wavelength conversion. We shall show that $W_{\mathbf{k}} \le w^*$, which implies $L_{\mathbf{k}} = W_{\mathbf{k}} = w^*$. We do so by constructing an on-line WA algorithm. We shall refer to w^* as the *worst case* number of wavelengths since w^* wavelengths are necessary and sufficient to support *any* traffic matrix in the $\mathbf{k}$-port traffic set. Since a star network is also a tree network, Fig. 2 illustrates that the WA algorithm design is not trivial for a tree network.

We now derive a few useful properties related to the worst case number of wavelengths w^*. Let e^* denote the link associated with w^*. Note that there may be multiple choices for e^*. We shall refer to e^* as the *bottleneck link* since it is the link with the maximum load under the worst case traffic.

Link e^* separates the leaf nodes into two sets $\mathcal{N}_{e^*,1}$ and $\mathcal{N}_{e^*,2}$. Without loss of generality, choose $\mathcal{N}_{e^*,1}$ such that the sum of k_i's in this set is w^*. We assume for now that $\mathcal{N}_{e^*,2}$ contains multiple leaf nodes, as illustrated in Fig. 6. Define the *bottleneck node* v^* to be the end point of e^* opposite to $\mathcal{N}_{e^*,1}$, i.e., the subtree connected to v^* by e^* has the sum of k_i's equal to w^*, as illustrated in Fig. 6.

We shall refer to each subtree connected to v^* as a *top-level subtree*. Note that a top-level subtree can be a single node. Let

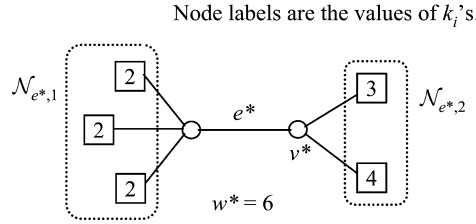

Fig. 6. Definition of the bottleneck node v^*.

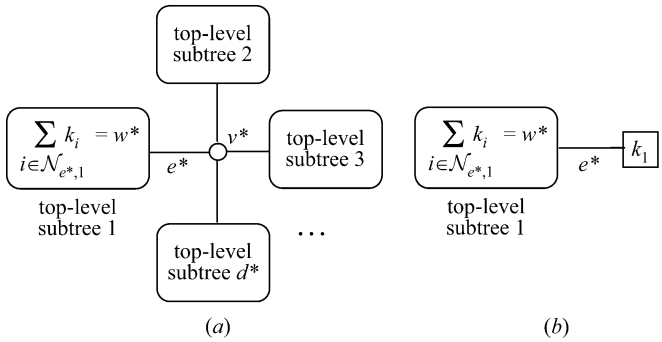

Fig. 7. The bottleneck node v^* and the top-level subtrees. (a) Multiple leaf nodes in N_{e^*2}. (b) Single leaf node in N_{e^*2}.

d^* be the degree of v^*.[2] Since v^* is a nonleaf node, $d^* \geq 3$. It follows that there are $d^* \geq 3$ top-level subtrees, as illustrated in Fig. 7(a).

If the set $\mathcal{N}_{e^*,2}$ contains a single node, we have the scenario illustrated in Fig. 7(b). In this case, assumption 1 implies that the value of k_i for the leaf node in $\mathcal{N}_{e^*,2}$ is equal to w^*. We argue that, with $N > 2$ leaf nodes, this scenario can be transformed to the scenario in Fig. 7(a) by exchanging the roles of $\mathcal{N}_{e^*,1}$ and $\mathcal{N}_{e^*,2}$. After the exchange, the set $\mathcal{N}_{e^*,2}$ will contain multiple nodes, and we have a scenario as illustrated in Fig. 7(a). Therefore, we shall consider only the scenarios in which v^* exists and $d^* \geq 3$, as illustrated in Fig. 7(a).

Note that the location of the bottleneck node v^* depends on the specific tree network and the traffic vector $\mathbf{k}$, but not on the current traffic matrix being supported. The following lemma provides useful properties of the top-level subtrees connected to v^* as well as the bounds on the worst-case number of wavelengths w^*.

Lemma 4: Under assumption 1, the following properties hold.

1) Let $K_j, 1 \leq j \leq d^*$, denote the sum of k_i's in top-level subtree j. For all $1 \leq j \leq d^*, K_j \leq w^*$.
2) Let $K = \sum_{1\leq i\leq N} k_i$. The worst-case number of wavelengths w^* is bounded by

$$K/d^* \leq w^* \leq K/2.$$

Proof:

1) We shall prove statement 1 by contradiction. Number the d^* top-level subtrees from 1 to d^* such that top-level subtree 1 is connected to v^* by link e^*. By the definition of v^*, we know that $K_1 = w^*$. Suppose now that $K_j > w^*$ for some j with $2 \leq j \leq d^*$. Consider the link e_j connecting subtree j to v^*. Since there are at least three subtrees, subtree j can send more than w^* sessions on link e_j to v^*, e.g., w^* sessions to subtree 1 and one more session to another subtree. It follows that link e_j can carry more than w^* sessions, contradicting the fact that e^* is the bottleneck link.
2) From the definition of w^*, it is clear that $w^* \leq K/2$. To prove the lower bound, we use statement 1 of the lemma, i.e., $K_j \leq w^*$ for all $1 \leq j \leq d^*$, to show that

$$K = \sum_{1\leq j\leq d^*} K_j \leq d^* w^*.$$

The above inequality yields the desired lower bound $w^* \geq K/d^*$. □

As in the star WA algorithm, the algorithm for tree networks is based on bipartite matchings. The main difference has to do with what a node in a bipartite graph represents. In the star WA algorithm, a node represents a single source or a single destination. In this section, a node represents a set of sources or a set of destinations in a top-level subtree.

For a given traffic matrix, we construct the *top-level subtree bipartite graph*, denoted by $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, as follows. We consider each leaf node as one distinct source and one distinct destination. Number the d^* top-level subtrees from 1 to d^*. The set $\mathcal{V}_1$ contains d^* abstract nodes, denoted by $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{d^*}$. Node $\mathcal{S}_i, 1 \leq i \leq d^*$, represents the set of sources contained in top-level subtree i. Similarly, the set $\mathcal{V}_2$ contains d^* abstract nodes, denoted by $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{d^*}$. Node $\mathcal{D}_j, 1 \leq j \leq d^*$, represents the set of destinations contained in top-level subtree j. In the set of edges $\mathcal{E}$, an edge from node $\mathcal{S}_i$ in $\mathcal{V}_1$ to node $\mathcal{S}_j$ in $\mathcal{V}_2$ exists for each traffic session from a source in top-level subtree i to a destination in top-level subtree j. Fig. 8 shows an example of the top-level subtree bipartite graph and its traffic matrix. Note that there may be multiple edges between the same pair of nodes. For example, since there are two sessions from top-level subtree 3 to top-level subtree 4, there are two parallel edges between the set of sources $\mathcal{S}_3$ and the set of destinations $\mathcal{D}_4$ in Fig. 8(d).

Define a *local* session to be a traffic session whose source and destination are in the same top-level subtree. Accordingly, a *nonlocal* session has its source and its destination in different top-level subtrees. A nonlocal session has to travel through the bottleneck node v^*, whereas a local session does not have to travel all the way to v^* and back to its destination, i.e., each session need not use the same link twice in the opposite directions. A nonlocal session corresponds to an edge from some node $\mathcal{S}_i$ in $\mathcal{V}_1$ and some node $\mathcal{D}_j$ in $\mathcal{V}_2$, where $i \neq j$. On the other hand, a local session corresponds to an edge between some node $\mathcal{S}_i$ in $\mathcal{V}_1$ and node $\mathcal{D}_i$ in $\mathcal{V}_2$. For example, the top-level subtree bipartite graph in Fig. 8(d) contains seven nonlocal sessions and one local session. The local session is from a source in top-level subtree 2 to a destination in the same top-level subtree.

Observe that the sessions belonging to a matching in the top-level subtree bipartite graph can be supported on a single wavelength without wavelength collision. To see this, note that any two sessions in a bipartite matching are transmitted from different top-level subtrees and to different top-level subtrees. Consequently, if these two sessions travel in the same top-level sub-

[2]Since we assume that each link consists of two fibers, one in each direction, the indegree and the outdegree of any given network node are the same. We simply refer to their value as the node degree.

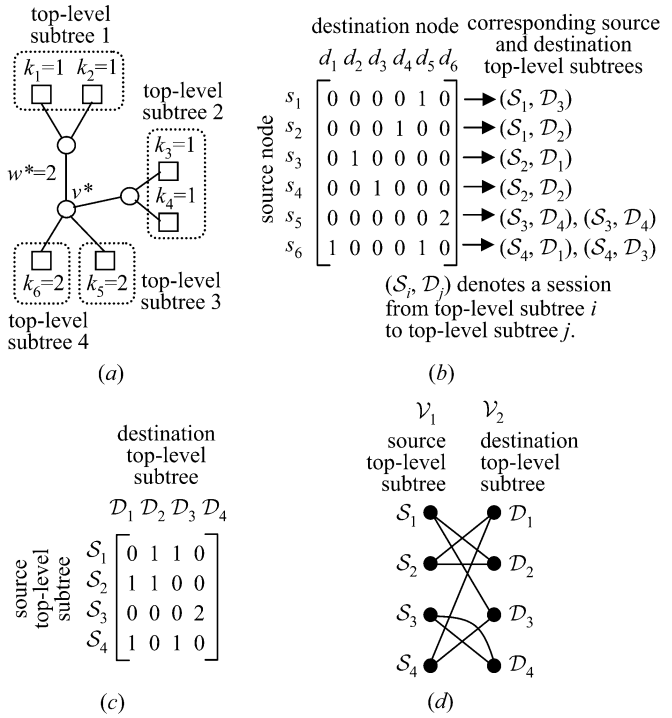

Fig. 8. Top-level subtree bipartite graph. (a) Tree topology with traffic parameter $\mathbf{k}$, (b) Traffic matrix for individual leaf nodes, (c) Traffic matrix for top-level subtrees, (d) Top-level subtree bipartite graph.

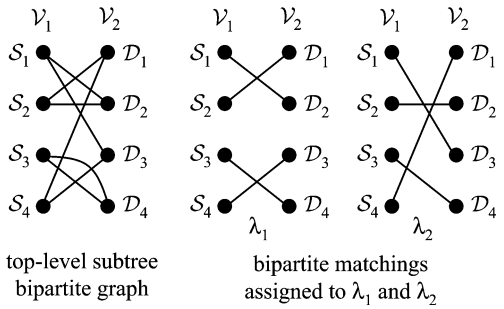

Fig. 9. Bipartite matchings of specific wavelengths.

tree, i.e., each session uses some fiber in that subtree, one session must be transmitted from that subtree while the other session must be received in that subtree. It follows that these two sessions always traverse links belonging to the same top-level subtree in the opposite directions and do not collide.

Our algorithm will assign a single bipartite matching to a single wavelength. We shall refer to the matching assigned to wavelength λ_1 as the bipartite matching of λ_1. Fig. 9 shows example bipartite matchings of specific wavelengths.

We now argue that w^* wavelengths are sufficient to support any traffic matrix in the $\mathbf{k}$-port set. From statement 1 of Lemma 4, each top-level subtree can transmit at most w^* wavelengths and receive at most w^* wavelengths. Thus, for a given a traffic matrix, each node in the corresponding top-level subtree bipartite graph has degree at most w^*. By Lemma 2, the set of edges can be partitioned into w^* disjoint bipartite matchings. The sessions in each matching can be supported on a single wavelength. Thus, w^* wavelengths are sufficient to support any $\mathbf{k}$-port traffic matrix. Notice that, by finding w^* disjoint bipartite matchings, we provide the WA for both local and nonlocal sessions simultaneously.

The main idea of our on-line WA algorithm involves keeping w^* disjoint bipartite matchings of w^* wavelengths such that each traffic session corresponds to an edge in one bipartite matching. When a session departs, we simply remove its corresponding lightpath from the network. When a new (local or nonlocal) session arrives, we update the WA by finding up to two wavelengths whose bipartite matchings can be reassigned to include the new session.

The following is our on-line WA algorithm for a tree network with $\mathbf{k}$-port traffic which uses w^* wavelengths in each fiber, is rearrangeably nonblocking, and requires at most $d^* - 1$ lightpath rearrangements per new session request. We refer to this algorithm as the *tree WA algorithm.*

Tree WA algorithm: (Use w^* wavelengths.)

Session termination: When a session terminates, simply remove its associated lightpath from the network without any futher lightpath rearrangement.

Session arrival: When a new session arrives and the resultant traffic matrix is still $\mathbf{k}$-port, proceed as follows. Assume that the new session is from a source in top-level subtree i to a destination in top-level subtree j. When $i = j$, the new session is local. Otherwise, it is nonlocal. In either case, follow the same procedures below.

Step 1: If there is a wavelength, denoted by λ_0, which is used by neither a source in top-level subtree i nor a destination in top-level subtree j, then assign the new session to λ_0. In this case, no lightpath rearrangement is required. Otherwise, proceed to step 2.

Step 2: Find a wavelength, denoted by λ_1, which is not used by any source in top-level subtree i, i.e., its bipartite matching is not incident on $\mathcal{S}_i$, and another wavelength, denoted by λ_2, which is not used by any destination in top-level subtree j, i.e., its bipartite matching is not incident on $\mathcal{D}_j$. Since the new session is admissible, there are at most $w^* - 1$ sessions from top-level subtree i. Since there are w^* available wavelengths, it follows that λ_1 exists. By the same argument, λ_2 always exists.

Modify the WA of only the sessions on λ_1 and λ_2. Contruct the top-level subtree bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$ in which the set of edges $\mathcal{E}'$ contains the bipartite matchings of λ_i and λ_j as well as the new edge $(\mathcal{S}_i, \mathcal{D}_j)$. From Lemma 3, we can partition the set $\mathcal{E}'$ into two disjoint bipartite matchings. In addition, since the value of V in Lemma 3 is equal to d^* for the top-level subtree bipartite graph, the two matchings can be assigned to λ_1 and λ_2 such that at most d^*-1 existing lightpaths need to be rearranged.

The construction of the tree WA algorithm implies the following theorem.

Theorm 2: For an arbitrary tree network with $\mathbf{k}$-port traffic among N leaf nodes and the bottleneck node v^* with degree $d^*, W_{\mathbf{k}}$ is given by

$$W_{\mathbf{k}} = L_{\mathbf{k}} = w^* = \max_{e \in \mathcal{T}} \min \left(\sum_{i \in \mathcal{N}_{e,1}} k_i, \sum_{i \in \mathcal{N}_{e,2}} k_i \right).$$

In addition, there exists, by construction, an on-line WA algorithm which uses w^* wavelengths in each fiber and requires at most $d^* - 1$ lightpath rearrangements per new session request.

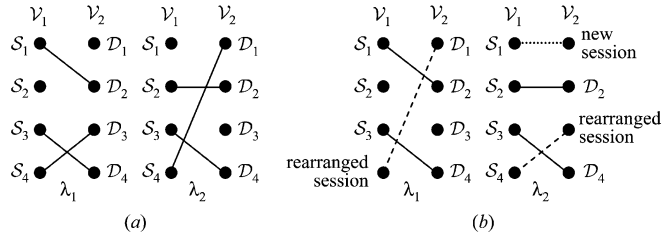

Fig. 10. Example operations of the tree WA algorithm.

Theorem 2 tells us that wavelength conversion cannot decrease the wavelength requirement for $\mathbf{k}$-port traffic in an arbitrary tree topology. In addition, if we scale the traffic vector $\mathbf{k}$ by an integer factor, then the location of the bottleneck node v^* remains fixed, and the upper bound on the number of lightpath rearrangements per new session request does not increase. Finally, from statement 2 of Lemma 4, among the tree topologies with N leaf nodes, the minimum value of the worst-case number of wavelengths w^* is at least $(\sum_{1\le i\le N} k_i)/d^*$. The tree topologies with w^* close to this lower bound are the ones in which each top-level subtree has the sum of k_i's approximately equal to $(\sum_{1\le i\le N} k_i)/d^*$. Roughly speaking, it is desirable to have all the subtrees support an equal amount of traffic.

The following example illustrates the operations of the tree WA algorithm.

Example 3: Consider the tree network with the traffic matrix given in Fig. 8. Note that $W_{\mathbf{k}} = 2$. Assume that the corresponding WA is given by the two bipartite matchings of wavelengths λ_1 and λ_2 as shown in Fig. 9. Now assume the following changes in the traffic matrix.

1) The existing session from source 3 in top-level subtree 2 to destination 2 in top-level subtree 1 on λ_1 terminates.
2) The existing session from source 1 in top-level subtree 1 to destination 5 in top-level subtree 3 on λ_2 terminates.
3) A new session from source 1 to destination 2 in top-level subtree 1 arrives.

After the second session termination, the bipartite matchings of λ_1 and λ_2 are shown in Fig. 10(a). To support the new session, the tree WA algorithm performs step 2. In particular, it creates a top-level subtree bipartite graph whose edges are the bipartite matchings of λ_1 and λ_2 as well as the new edge $(\mathcal{S}_1, \mathcal{D}_1)$. The algorithm then partitions the set of edges into two disjoint bipartite matchings and assign them to λ_1 and λ_2, as shown in Fig. 10(b). In particular, the session from top-level subtree 4 to top-level subtree 3 on λ_1 is reassigned to λ_2. In addition, the session from top-level subtree 4 to top-level subtree 1 on λ_2 is reassigned to λ_1. The new session is then assigned to λ_2. In this example, two rearrangements of existing lightpaths are made to support the new session.

An example similar to the one based on Fig. 5 for a star network can be constructed to show that the tree WA algorithm may perform up to $d^* - 1$ lightpath rearrangements to support a new session request. We omit the details here.

C. Tree Networks With Non-Leaf End Nodes

In this section, we relax the assumption that only leaf nodes in a given tree network are end nodes, i.e., nodes which source and sink traffic sessions. This relaxation also allows us to embed a tree network in an arbitrary mesh network. We emphasize that the purpose of this section is to demonstrate a possibility of using our tree WA algorithm for a tree with nonleaf end nodes. In general, embedding a tree in a mesh network does not yield an optimal solution to the RWA problem. In addition, note that there are several possible embedded trees for a given mesh network. We make no effort here in selecting an embedded tree with the minimum wavelength requirement, i.e., minimum value of $W_{\mathbf{k}}$.

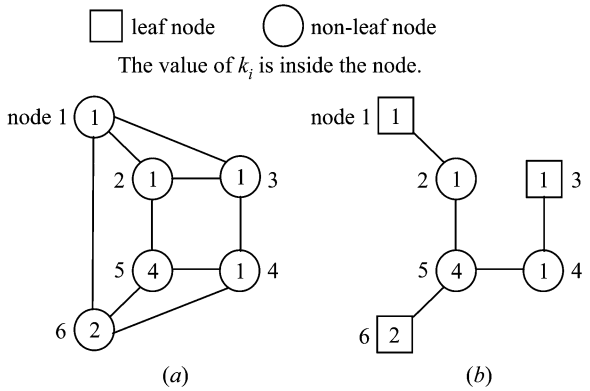

Fig. 11. Example embedded tree in a mesh network.

As an example, Fig. 11(a) shows an arbitrary mesh network. One possible embedded tree is shown in Fig. 11(b). Note that nodes 2, 4, and 5 are nonleaf end nodes.

We now show that, for a tree with nonleaf end nodes, the minimum required number of wavelengths $W_{\mathbf{k}}$ is equal to w^* as given in (1). We define the bottleneck link e^* and the bottleneck node v^* as before. Since w^* is the maximum possible link load, it is clear that $W_{\mathbf{k}} \ge w^*$. To show that $W_{\mathbf{k}} \le w^*$, consider two possible cases: 1) v^* is not an end node, and 2) v^* is an end node.

In either case, we can repeat the proof of statement 1 in Lemma 4 to show that each top-level subtree contains at most w^* transmitters and at most w^* receivers. Consider case 1 in which v^* is not an end node. We can use the tree WA algorithm developed so far without any modification as a proof that $W_{\mathbf{k}} \le w^*$. It remains to consider case 2 in which v^* is an end node.

With v^* being an end node, we modify the top-level subtree bipartite graph as follows. In the modified bipartite graph, there are $2d^*$ source nodes, where d^* is the node degree of v^*. There are d^* nodes, denoted by $\mathcal{S}_1$ to $\mathcal{S}_{d^*}$, that correspond to the set of sources in the d^* subtrees connected to v^*. In addition, there are d^* nodes, denoted by $\mathcal{S}_{0,1}$ to $\mathcal{S}_{0,d^*}$, that represent v^* as a source transmitting to subtrees 1 to d^* respectively. Fig. 12 shows an example modified bipartite graph. Note that a session from v^* to a node in subtree 1 corresponds to an edge adjacent to source node $\mathcal{S}_{0,1}$ and destination node $\mathcal{D}_1$ in the bipartite graph. Similarly, there $2d^*$ destination nodes.

It is clear that the sessions from each bipartite matching can be supported on a single wavelength. Because the number of transmitters/receivers in each subtree is at most w^*, it follows that the maximum node degree in the modified bipartite graph is

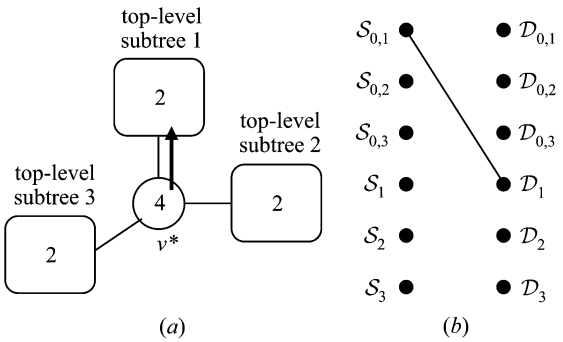

Fig. 12. Modified top-level subtree bipartite graph.

at most w^*. Consequently, the set of edges in the bipartite graph can be partitioned into w^* disjoint matchings. Thus, $W_{\mathbf{k}} \leq w^*$.

We can apply Lemma 3 and the fact that the number of source/destination nodes in the modified bipartite graph is $2d^*$ to conclude that the number of lightpath rearrangements required for each new session is bounded by $2d^* - 1$. We summarize the results in this section formally below.

Lemma 5: For an arbitrary tree with nonleaf end nodes and $\mathbf{k}$-port traffic, $W_{\mathbf{k}} = w^*$, where w^* is given in (1). In addition, there exists an on-line WA algorithm that uses w^* wavelengths and requires no more than $2d^* - 1$ lightpath rearrangements per new session request.

D. Strict-Sense Nonblocking WA Algorithm

We show in this section that, by using roughly twice the number of wavelengths, we can modify the tree WA algorithm to become strict-sense nonblocking. In particular, with $2w^* - 1$ wavelengths where w^* is given in (1), we can use a simple *first-fit* WA scheme in which a new session is assigned to *any* available wavelength. The following lemma provides a justification.

Lemma 6: For a tree network with $\mathbf{k}$-port traffic, the number of wavelengths for strict-sense nonblocking, denoted by $\tilde{W}_{\mathbf{k}}$, is bounded by

$$w^* < \tilde{W}_{\mathbf{k}} \leq 2w^* - 1.$$

Proof: The lower bound is justified by example 2 which shows the need for lightpath rearrangement when w^* wavelengths are used. To prove the upper bound, consider the first-fit WA scheme. Suppose the new session is from subtree i to subtree j. Then there are at most $w^* - 1$ sessions transmitted from subtree i, and at most $w^* - 1$ sessions received in subtree j. With $2w^* - 1$ wavelengths available, there must be at least one free wavelength. We can use any free wavelength to support the new session without lightpath rearrangement. □

The following example shows that the above upper bound is tight. Consider top-level subtrees 1 to 3 each of which contains w^* transmitters and w^* receivers. Suppose there are $w^* - 1$ sessions from subtree 1 to subtree 3 supported on wavelengths 1 to $w^* - 1$. In addition, there are $w^* - 1$ sessions from subtree 3 to subtree 2 supported on wavelengths w^* to $2w^* - 2$. At this point, a new session from subtree 1 to subtree 2 requires a new wavelength, for a total of $2w^* - 1$ wavelengths.

To our knowledge, the minimum number of wavelengths for a wide-sense nonblocking WA algorithm is still unknown. From the fact that the number of wavelengths for wide-sense nonblocking is no more than $\tilde{W}_{\mathbf{k}}$, it follows that the bounds in Lemma 6 also applies for wide-sense nonblocking.

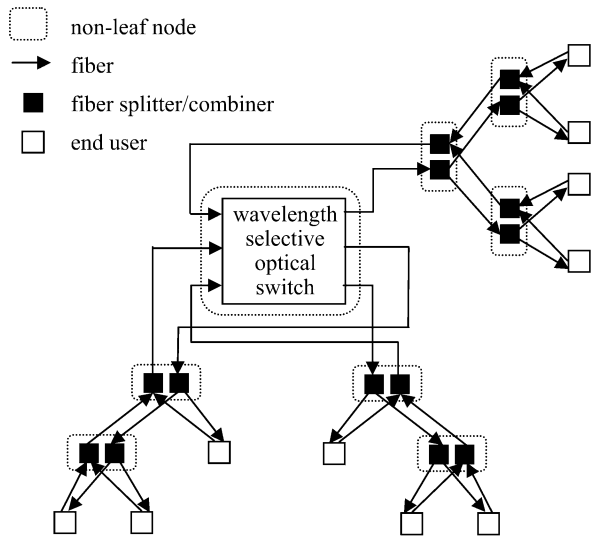

Fig. 13. Hybrid wavelength-routed/broadcast tree that uses the same number of wavelengths as a wavelength-routed tree but only one switching node.

IV. Hybrid Wavelength-Routed/Broadcast Trees

In this section, we point out that the tree WA algorithm can be implemented on a tree with only one switching node connecting several passive broadcast subtrees. The key observation is that, in the tree WA algorithm, routing each local session from the source all the way to the bottleneck node and back to its destination does not cause any wavelength collision and allows the implementation of each subtree as a passive broadcast tree. Fig. 13 illustrates an example of such a hybrid wavelength-routed/broadcast tree. Note that the hybrid tree is wavelength efficient; it uses the same number of wavelengths as the corresponding wavelength-routed tree. In particular, the hybrid tree uses w^* wavelengths as opposed to $\sum_{i=1}^{N} k_i$ wavelengths for the corresponding broadcast tree. In terms of hardware, the hybrid tree differs from a passive broadcast tree only by one switching node.

For a practical network such as a distribution tree in a metropolitan optical network, it is desirable to locate this only switch at the access node of the tree and keep the rest of the tree passive. However, the bottleneck node and the access node may not be the same. In that case, we could still locate the only switch at the access node at the expense of additional wavelengths. In particular, suppose that the access node connects $\tilde{d}$ subtrees. Let $\tilde{w}$ be the maximum total number of transmitters/receivers among all the subtrees. Then, using $\tilde{w}$ wavelengths, we can construct a hybrid tree using the only switch at the access node. The WA algorithm is exactly the same as the tree WA algorithm but with the access node taking a role of the bottleneck node. We omit the details here.

V. Conclusion

We developed an on-line WA algorithm for dynamic $\mathbf{k}$-port traffic in a WDM all-optical tree network. Our algorithm is re-

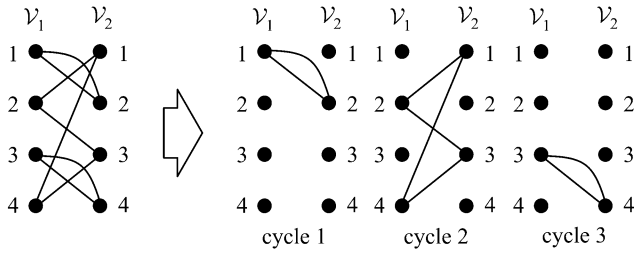

Fig. 14. Cycles in a bipartite graph with node degree 2.

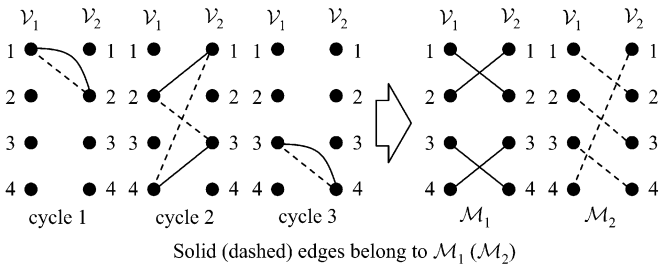

Fig. 15. Assignment of edges to matchings $\mathcal{M}_1$ and $\mathcal{M}_2$.

arrangeably nonblocking, uses the minimum number of wavelengths, and requires at most $d^* - 1$ lightpath rearrangements per new session request, where d^* is the degree of the bottleneck node. We observed that the number of lightpath rearrangements per new session request does not increase as the amount of traffic $\mathbf{k}$ scales up by an integer factor. We also showed that, using roughly twice the number of wavelengths, we can modify the tree WA algorithm to become strict-sense nonblocking.

For an arbitrary tree topology, we found that the minimum numbers of wavelengths required to support $\mathbf{k}$-port traffic with full wavelength conversion at all nodes and without wavelength conversion are the same. This implies that the use of wavelength converters will not decrease the required number of wavelengths.

Most of the complexity in our on-line WA algorithm involves partitioning the edges in a bipartite graph with maximum node degree 2 into two disjoint matchings. We pointed out that this can be done in time $O(d^*)$ for each WA update. matching exist, we developed a specialized for our task.

Finally, we showed that the tree WA algorithm can be implemented on a hybrid wavelengh-routed/broadcast tree using only one optical switch at the bottleneck node. For a practical network such as a distribution tree in a metropolitan optical network, it is desirable to use the only switch at the access node so that the rest of the tree is passive. In the case where the access node is not the bottleneck node, we described how the tree WA algorithm can be used at the expense of additional wavelengths.

Our future goal is to develop an on-line RWA algorithm for other types of network topologies. Since a tree network can be embedded in any connected network, we hope that our analytical approach in this paper can be extended to create an on-line RWA algorithm for other types of network topologies.

APPENDIX
EFFICIENT BIPARTITE MATCHING WITH MAXIMUM NODE DEGREE 2

We present below a simple and efficient algorithm for partitioning the edges in a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with $|\mathcal{V}_1| = |\mathcal{V}_2| = V$ and maximum node degree 2 into two disjoint matchings. The algorithm has the running time $O(V)$. We omit the proof details which can be found in [28].

Assume for now that each node has degree 2. The main idea of our algorithm is as follows. In a bipartite graph with node degree 2, the edges in $\mathcal{E}$ form a set of disjoint cycles each of which contains an even number of edges. For example, Fig. 14 shows three disjoint cycles in a bipartite graph with node degree 2.

For each cycle, we move along the edges of the cycle and alternately assign them to two matchings, denoted by $\mathcal{M}_1$ and $\mathcal{M}_2$, such that no two adjacent edges belong to the same matching. Note that this is possible since there are even number of edges in each cycle. Finally, we collect the edges in all disjoint cycles to form two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$, as illustrated in Fig. 15. We describe the algorithm formally below. We shall refer to this algorithm as the *degree-2 bipartite matching algorithm*.

Degree-2 Biparite Matching Algorithm: Given a bipartite graph $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ with $|\mathcal{V}_1| = |\mathcal{V}_2| = V$ and node degree 2, form two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ as follows.

Step 1: Form a new cycle disjoint from all the previous cycles starting from the lowest-index node in $\mathcal{V}_1$ with an incident edge not yet assigned to either $\mathcal{M}_1$ or $\mathcal{M}_2$. Assign the edges in this cycle alternately to $\mathcal{M}_1$ and $\mathcal{M}_2$ such that no two adjacent edges in the cycle are assigned to the same matching.

Step 2: Look for the next lowest-index node in $\mathcal{V}_1$ with an incident edge not yet assigned to either $\mathcal{M}_1$ or $\mathcal{M}_2$. If such a node exists, proceed to step 1. If such a node does not exist, terminate the algorithm.

The running time $O(V)$ can be seen from the fact that the above algorithm visits each node in the bipartite graph exactly once.

We now relax the assumption that each node has degree 2. If some node has degree less than 2, we can add extra edges to the bipartite graphs to make all the nodes have degree 2. After using the above algorithm to find two disjoint matchings, we can remove the extra edges to obtain the two desired matchings.

REFERENCES

[1] J. Y. Yoo and S. Banerjee. Design, Analysis, and Implementation of Wavelength-Routed All-Optical Networks. [Online]. Available: home.att.net/sbanerjee/papers/YoBa97.pdf

[2] B. Beauquier *et al.*, "Graph problems arising from wavelength-routing in all-optical networks," presented at the 2nd Workshop on Optics and Computer Science (WOCS '97), Geneva, Switzerland, Apr. 1997.

[3] H. Zang, J. P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Opt. Netw. Mag.*, vol. 1, no. 1, pp. 47–60, Jan. 2000.

[4] D. Banerjee and B. Mukherjee, "A practical approach for routing and wavelength assignment in large wavelength-routed optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 903–908, Jun. 1996.

[5] R. Ramaswami and K. N. Sivarajan, "Design of logical topologies for wavelength-routed optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 840–851, Jun. 1996.

[6] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171–1182, Jul. 1992.

[7] R. Ramaswami and K. N. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 5, pp. 489–500, Oct. 1995.

[8] Z. Zhang and A. S. Acampora, "A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength reuse," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 281–288, Jun. 1995.
[9] C. Chen and S. Banerjee, "A new model for optimal routing and wavelength assignment in wavelength division multiplexed optical networks," in *Proc. IEEE INFOCOM*, Mar. 1996, pp. 164–171.
[10] S. Banerjee, J. Yoo, and C. Chen, "Design of wavelength-routed optical networks for packet switched traffic," *IEEE J. Lightw. Technol.*, vol. 15, no. 9, pp. 1636–1646, Sep. 1997.
[11] K. C. Lee and V. O. K. Li, "A wavelength rerouting algorithm in wide-area all-optical networks," *IEEE J. Lightw. Technol.*, vol. 14, no. 6, pp. 1218–1229, Jun. 1996.
[12] B. Mukherjee *et al.*, "Some principles for designing a wide-area WDM optical network," *IEEE/ACM Trans. Netw.*, vol. 4, no. 5, pp. 684–696, Oct. 1996.
[13] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 852–857, Jun. 1996.
[14] S. Subramaniam, M. Azizoglu, and A. K. Somani, "All-optical networks with sparse wavelength conversion," *IEEE/ACM Trans. Netw.*, vol. 4, no. 4, pp. 544–557, Aug. 1996.
[15] R. A. Barry and P. A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 858–867, Jun. 1996.
[16] L. Li and A. K. Somani, "Dynamic wavelength routing using congestion and neighborhood information," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 779–786, Oct. 1999.
[17] Y. Zhu, G. N. Rouskas, and H. G. Perros, "A path decomposition algorithm for computing blocking probabilities in wavelength routing networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 6, pp. 747–762, Dec. 2000.
[18] O. Gerstel, G. Sasaki, S. Kutten, and R. Ramaswami, "Worst-case analysis of dynamic wavelength allocation in optical networks," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 833–845, Dec. 1999.
[19] R. K. Pankaj, "Architectures for Linear Lightwave Networks," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, Sep. 1992.
[20] A. Narula-Tam, P. J. Lin, and E. H. Modiano, "Efficient routing and wavelength assignment for reconfigurable WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 1, pp. 75–88, Jan. 2002.
[21] P. Saengudomlert, E. H. Modiano, and R. G. Gallager, "An on-line routing and wavelength assignment algorithm for dynamic traffic in a WDM bidirectional ring," in *Proc. Joint Conf. Information Sciences (JCIS)*, Durham, NC, Mar. 2002, pp. 1331–1334.
[22] ——, "On-line routing and wavelength assignment for dynamic traffic in WDM ring and torus networks," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 164–171.
[23] C. L. Monma and V. K. Wei, "Intersection graphs of paths in a tree," *J. Combinatorial Theory*, ser. B, vol. 41, no. 2, pp. 141–181, Oct. 1986.
[24] V. W. S. Chan *et al.*, "Architectures and technologies for high-speed optical data networks," *IEEE J. Lightw. Technol.*, vol. 16, no. 12, pp. 2146–2168, Dec. 1998.
[25] J. P. Labourdette, G. W. Hart, and A. S. Acampora, "Branch-exchange sequences for reconfiguration of lightwave networks," *IEEE Trans. Commun.*, vol. 42, no. 10, pp. 2822–2832, Oct. 1994.
[26] C. Berge, *Graphs*. Amsterdam, The Netherlands: North-Holland, 1985.
[27] A. Schrijver, "Bipartite edge coloring in $O(\Delta m)$ time," *SIAM J. Computing*, vol. 28, no. 3, pp. 841–846, 1998.
[28] P. Saengudomlert, "Architectural Study of High-Speed Networks with Optical Bypassing," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, Sep. 2002.

Poompat Saengudomlert (S'94–M'03) received the B.S.E. degree in electrical engineering from Princeton University in 1996. He then received the M.S. and Ph.D. degrees, both in electrical engineering and computer science, from Massachusetts Institute of Technology (MIT), Cambridge, in 1998 and 2002, respectively.

From 2002 to 2004, he was a Postdoctoral Associate at the Laboratory for Information and Decision Systems at MIT. Currently, he is an Assistant Professor at the Asian Institute of Technology, Thailand. His research interest includes communication theory, data networks, and resource allocation problems.

Dr. Saengudomlert received the King Scholarship from Thailand in 1991.

Eytan H. Modiano (S'90–M'93–SM'00) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986 and the M.S. and Ph.D. degrees, both in electrical engineering, from the University of Maryland, College Park, in 1989 and 1992, respectively.

He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow during 1992–1993 while he was conducting research on security and performance issues in distributed network protocols. Between 1993 and 1999, he was with the Communications Division at MIT Lincoln Laboratory where he designed communication protocols for satellite, wireless, and optical networks and was the project leader for MIT Lincoln Laboratory's Next Generation Internet (NGI) project. He joined the MIT faculty in 1999, where he is presently an Associate Professor in the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS). His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks.

Robert G. Gallager (S'58–M'61–F'68–LF'96) received the B.S. degree from the University of Pennsylvania, Philadelphia, in 1953, and the S.M. and Sc.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, in 1957 and 1960, respectively.

He has been a faculty member in electrical engineering and computer science at MIT since 1960. He has worked primarily on information theory, digital communication, and networks. He is the author of the textbooks *Information Theory and Reliable Communication* (Wiley, 1968), *Data Networks* (Prentice-Hall, 2nd ed., 1992, coauthored with Bertsekas), and *Discrete Stochastic Processes* (Kluwer, 1995). He is a Life Fellow of the IEEE, member of the National Academy of Engineering (1979) and the National Academy of Sciences (1992), and a Fellow of the American Academy of Arts and Sciences (1999). He received the IEEE Medal of Honor in 1990, the Technion Harvey Prize in Science and Technology in 1999, the Eduard Rhein Prize for basic research in 2002, and the Marconi Award in 2003.