”Light-mesh” time division multiplexing for CWDM/DWDM networks [1]

Alpár Jüttner, Jie Zhang
CWIND, University of Bedfordshire, Luton, UK.
e-mail: alpar@cs.elte.hu, jie.zhang@beds.ac.uk,

ABSTRACT

As an extension of the known Light Trail and Light Tree techniques, we explore the most possible general setting of time division multiplexing in circuit switched all optical networks. It will be shown that this technique improves the scalability and adaptability of all-optical networks without the need of expensive techniques like wavelength converters and burst switching or packet switching capable network nodes.

1 Introduction

In response to the emerged need for a transport technology that would achieve an efficient use of all-optical networks when carrying IP traffic, a number of frameworks have been proposed, such as *lightpaths* (wavelength routing networks)[8], *Optical Burst Switching* (OBS)[5], *Optical Packet Switching* (OPS)[4] and the *light-trails*[3].

This latter technology provides sub-wavelength bandwidth granularity to all-optical networks by using time division demultiplexing of a single wavelength light-path and apply a drop-and-continue sharing scheme. It turned to be superior the other listed technologies mainly because it is based on readily available physical components while allowing efficient handling of sub-wavelength demands.

The downside of this technology is its highly restricted topology (each light-trail is a path), that makes it poorly configurable and impossible to provide full N^2 connectivity using only light-trail.

To overcome this drawback, Section 2 proposes an extension to light-trails called *light-mesh*, which — at the cost of a slightly more complex synchronization architecture — allows more complex logical single wavelength topologies instead of paths therefore provide a better utilization of resources and higher flexibility.

However, not all logical topologies are admissible as a light-mesh and in fact it is a non-trivial task to check whether or a set of routes can be accommodated in a light-mesh. Therefore Section 3 proposes an easy-to-check condition for feasible configuration.

Once the light-mesh topologies are established and the demands are assigned to them, the next step is to find a collision free and bandwidth optimal way of sharing resources between the demands. Section 4 proves that is can be optimally solved by an efficient greedy-type algorithm.

These two algorithms can be bases of optimization tools for planning a set of light-meshes maintaining full connectivity and carrying the demands in a bandwidth efficient way.

2 Light-Trails and Light-Meshes

A light-trail is sequence of a several neighboring nodes with one directional data transmission (from upstream to downstream) on a single wavelength. The data is injected into the network and leaves it using “add” and “drop” couplers, respectively. The resource reuse is established by the use of an optical shutter at each node, making it possible to either allow or block parts of the optical signal or further propagate downstream. We assume that it is done by defining a cyclic time-frame, which in turn divided into certain number of time-slots. The optical shutter are able to block these slots individually.

For the correct behavior, it is crucial to *synchronize* the time frames of each node in the light-trail. This process is easy because the downstream nodes can directly synchronize themselves to the signal arriving from upstream.

The granularity of bandwidth is therefore a single slot. For the sake of simplicity, we assume that the bandwidth of each demand is 1 unit, larger demands are established by allocating more demands between the same endpoints.

An obvious extension is to combine light-trails with the *light-tree*[6] technology, i.e. to allow the trail to *fork* at the nodes. Then the underlying topology will form a rooted *branching* (i.e. a directed rooted tree, where the edges are

[1]The work reported in this paper was supported by GAWIND, grant number MTKD-CT-2006-042783

oriented oppositely to the root node). By using additional optical shutters, we can realize a full control over which portion of the traffic should be propagated to each individual branches. The frame synchronization is as easy as in case of the traditional light trails.

The next step in the generalization is to allow light-trails *to merge* together. The frame synchronization is significantly more complex in this case, as the synchronicity must also be propagated upstream. It can be done by implementing a feedback mechanism that reports upstream about the phase of the arriving signal.

The proposed *light-mesh* topologies are just those are resulted by allowing *both forking and merging of light-trails* in the freest possible ways. It enables complex single wavelength network topologies, but the frame synchronization can be impossible in certain configurations due to cyclic dependencies.

3 Admissible Light-Mesh Topologies

This section presents an easy-to-check necessary and sufficient condition whether or not a set D of demands with given routes fits into a single wavelength using light-mesh time division multiplexing. In addition it also provides a scheme for consistent synchronization master selection for the links.

From now on, the physical network topology is represented by a directed graph $G_{net} = (V, A)$ with the set V of nodes and the set A of optical links. Furthermore, we consider to types of demands.

Point-to-point (unicast). This kind of demand has a single source and a single destination, thus the route of the traffic is a simple path between them. The traffic must use the same time slot in each links of this path.

Point-to-multipoint (multicast). Here the traffic is originated from a single source s but the *same data* is sent to several destinations.The usual topology for this kind of demand is a rooted tree, which fits well in the the acyclic nature of the synchronization dependency mentioned above. Because the electronic to optical conversion is done at the source node s, *we can use different time-slots for the different outgoing links of* s, but then the corresponding subtrees must use the same slots.

As it was mentioned above, the frame cycles in a light-mesh must be synchronized. More precisely, if there is a demand arriving at a node on link l_1 and leaves it on link l_2, then the frame cycle of these two links must be synchronous. Due to the finite speed of light, these can only be done if the synchronicity dependency is acyclic, otherwise we get an overdetermined system.

To describe this phenomenon precisely, we use the notion of *line graphs*[7].

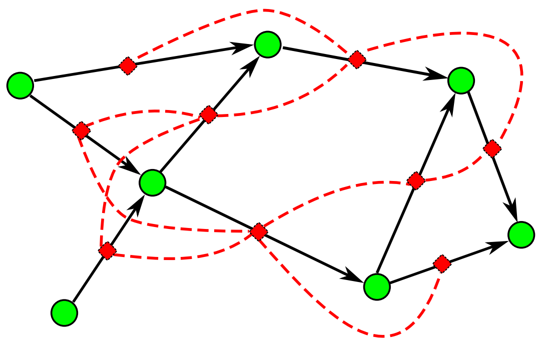
Figure 1: Line graph (the red graph is the line graph of the black/green one)

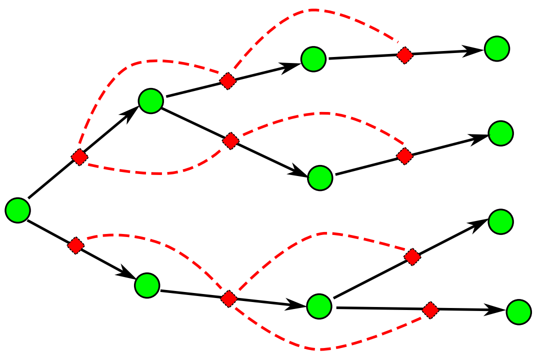
Figure 2: Line graph of a multicast tree

Definition 1 (Line-Graph) *Let $G = (V, A)$ be a directed graph. Its* ***line-graph*** *$L(G) = (A, E)$ is an undirected graph the nodes of which correspond to the edges of G, and two nodes a_1 and a_2 are connected by an edge if and only if the head a_1 and the tail of a_2 is the same node in G, i.e.*

$$E := \{(\overrightarrow{xu}, \overrightarrow{uy}) : x, y, u \in V \text{ and } \overrightarrow{xu}, \overrightarrow{uy} \in A\} \quad (1)$$

A path $p = (a_1, a_2, \dots, a_k)$ of length k in G corresponds to a path $L(p)$ of length $k-1$ in $L(G)$, where

$$L(p) := ((a_1a_2), (a_2a_3), \dots, (a_{k-1}a_k)) \quad (2)$$

If the length of p *is 1, then* $L(p)$ *is a single node in* $L(G)$*.*

Similarly, a rooted branching T *(i.e. a subtree with all edges directed oppositely to the root) naturally correspond the a forest (a set of disconnected trees)* $L(T)$ *in* $L(G)$*.*

As we can see from their definitions, the line-graphs of both unicast and multicast routes are branchings in $L(G)$. The following theorem tell us whether or not a set of demands of this type can be accommodated in a single light-mesh.

Theorem 2 *A system* $\{d_1, d_2, \dots, d_k\}$ *of demands are assignable to one single wavelength light-mesh if and only if the union* U *of their images* $L(d_1), L(d_2), \dots, L(d_k)$ *in the line graph forms a forest (i.e. an acyclic subgraph) in* $L(G)$*.*

Proof. An edge e in the line-graph $L(G)$ connects two physical links in the network. Clearly, if $e \in L(d_i)$ for some demand d_i, it means that these physical links must be synchronized. Therefore a cycle in U represents an inadmissible cyclic dependency.

Now, assume that U is acyclic. In this case, we give a consistent synchronization rule, i.e. for each physical link l, we define a neighboring link $master(l)$ to which l must be synchronized (except for some root links, which can generate their frame cycle independently). For this, let us choose an arbitrary root link (i.e. a vertex of $L(G)$) in each component of U and orient each edge of U towards the root in its subtree. Obviously, l again corresponds to a vertex in U and — unless it is a root — there is exactly one outgoing edges from l in U. Let $master(l)$ be the other end of this node. It is straightforward to check that this choice establishes a consistent synchronization dependency. □

4 Optimal Slot Allocation In a Light Mesh

Assume we are given k demands $d_1, d_2, \dots, d_k$ (either unicast or multicast) which fit into a light-mesh, i.e. the union of their images $L(d_i)$ in the line graph forms a forest F. This section discusses the nontrivial question of how to assign the demands to the slots in a collision-free way. Naturally, a link cannot be used by more demands than the number of available slots. The following theorem says that this limit can always be achieved.

Algorithm 1 Assign the demands to the slots

```
1: Let $T_1, T_2, \dots, T_C$ be the connected components of $F$.
2: for all $c = 1$ to $C$ do
3:     Choose and arbitrary root vertex $r_c \in T_c$.
4: end for
5: for all $d_i$ do
6:     Let $a_i \in L(d_i)$ be the vertex that is the closest to the root of its component.
7:     Let $dist(i)$ be the distance between $a_i$ and the root.
8: end for
9: for all vertices $l$ in $L(G)$ do
10:    Let $free\_slots(l)$ be the list of available slots.
11: end for
12: for all $d_i$ in increasing order according to $dist(i)$ do
13:    Let $s \in free\_slots(a_i)$.
14:    Assign $d_i$ to slot $s_i$.
15:    for all $l$ vertices in $L(d_i)$ do
16:        Remove $s_i$ from $free\_slots(l)$.
17:    end for
18: end for
```

Theorem 3 *Assume that the cyclic time frame is divided into* S *slots. Then, the demands can be assigned to the slots in a collision-free way if and only if each link is used by at most* S *demands.*

Proof. Clearly, if a link is used by more than S demands, a feasible assignment cannot exists.

To complete the proof, we show a greedy algorithm that finds an appropriate assignment, assuming that no edges are used by more than s demands. The algorithm works as follows (see Algorithm 1).

Let $T_1, T_2, \ldots, T_C$ denote the connected components of F and choose an arbitrary root vertex $r_c \in T_c$ in each component. For each demand d_i, let $a_i \in L(d_i)$ be the vertex that is the closest to the root of its component and let $dist(i)$ be the distance between a_i and the root. Now, simply take each demand one-by-one in an increasing order according to $dist(i)$ and assign the demand d_i to any slot that is unused at vertex a_i at the time of the assignment.

Observe, that the algorithm is always able to find an unused slot, otherwise the link a_i would have been used more than S demands.

Finally, we show that the algorithm provides a feasible assignment, i.e. for each demand d_i, the chosen slot does not collide with the previous assignments. To the contrary, assume that slot s_i is used by demand d_j for some $j < i$ and $L(d_i) \cap L(d_j) \neq \emptyset$. However, both $L(d_i)$ and $L(d_j)$ are connected subtrees of F and $dist(j) \leq dist(i)$, yielding that $a_i \in L(d_i) \cap L(d_j)$, which contradicts the choice of s_i. $\square$

5 Conclusion

This paper proposes an extension to the well know light-trail technology in order to allow more complex logical network topologies the provide better resource utilization and higher flexibility. The presented algorithms for admissibility checking and demand to time slot assignment call for their application in network planning tools as a further research.

Acknowledgment

The author wishes to thank Tibor Cinkler and András Frank for their useful comment.

References

[1] A. S. Ayad, K. M. El Sayed, and S. H. Ahmed. Efficient solution of the traffic grooming problem in light-trail optical networks. In *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications*, pages 622–627, Washington, DC, USA, 2006. IEEE Computer Society.

[2] J. Fang, W. He, and A. Somani. Optimal light trail design in wdm optical networks. In *International Conference on Communications*, volume 3, pages 1699–1703. IEEE, jun 2004.

[3] A. Gumaste. *Light-trail and Light-frame Architectures for Optical Networks*. Dallas, The University of Texas, 2003.

[4] M. Mahony, D. Simeonidou, D. Hunter, and A. Tzanakaki. The application of optical packet switching in future communication networks. *IEEE Communication Magazine*, page 128135, mar 2001.

[5] C. Qiao and M. Yoo. Optical burst switching (OBS) — a new paradigm for an optical internet. *J. High Speed Networks*, 8:6984, 1999.

[6] L. H. Sahasrabuddhe and B. Mukherjee. Light-trees: 0ptical multicasting for improved performance in wavelength-routed networks. *IEEE Communications Magazine*, 37(2):67–73, feb 1999.

[7] H. Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54:150–168, 1932.

[8] Z. Zhang, J. Fu, D. Guo, and L. Zhang. Lightpath routing for intelligent optical networks. *IEEE Network*, 15(4):28–35, jul-aug 2001.