♦ SPIDER: A Simple and Flexible Tool for Design and Provisioning of Protected Lightpaths in Optical Networks

R. Drew Davis, Krishnan Kumaran, Gang Liu, and Iraj Saniee

Optical devices are poised to form the core of the next generation of backbone and enterprise networks. Optical routers, dense wavelength division multiplexing (DWDM) systems, and cross connects of unprecedented capacities are on the verge of large-scale commercial deployment. This massive buildup of optical gear necessitates careful planning and provisioning of the basic units of transmission—the lightpaths. This paper presents a range of techniques for efficient and reliable optical network design, covering decentralized dedicated protection to shared path-based mesh restoration. These algorithms have been incorporated into SPIDER, an extensible software tool with a browser-based user interface, Java-based visualization, and spreadsheet input/output capabilities. We describe SPIDER and report on recent core networking applications using this tool, which also illustrate the key tradeoffs in optical network designs involving a variety of grades of protection and the balance between efficient use of wavelengths and restoration time.*

Introduction

Optical components and systems are ready to play a key role in next-generation backbone and enterprise networks. Routers with optical interfaces, dense wavelength division multiplexing (DWDM) systems, and optical cross connects of unprecedented capacities are on the verge of large-scale commercial deployment. This massive buildup of optical equipment calls for careful planning and provisioning of the basic units of transmission—the optical lightpaths.

This paper presents techniques for efficient and reliable design of optical networks through failure-resilient lightpath provisioning. We consider a range of routing options, their associated tradeoffs for optimization of wavelengths, and lightpath protection against optical fiber and interface failures ranging from decentralized dedicated protection to shared path-based mesh restoration. This evaluation calls for novel network design software that not only employs the relevant optimization solvers but also has the flexibility to solve for a variety of routing and protection mechanisms. The relative merits of distinct core architectures and provisioning schemes can therefore be measured for any specific network. Furthermore, new routing protocols can be quickly evaluated in terms of their comparative efficiency and restoration properties. SPIDER is a software tool built at Bell Labs to help evaluate the various options for design and provisioning of lightpaths in core optical networks. In this paper, we review both SPIDER and the problems it solves in some detail.

In the next section, we discuss basic optical design and routing problems from an algorithmic point of view and describe the routing and protection/restoration algorithms currently incorporated into SPIDER. In the section following it, we describe the software architecture of SPIDER, including its browser-based user interface, Java*-based visualization, and spreadsheet input/output capabilities. In a subsequent section, we give examples of a SPIDER core optical design for a U.S.-wide network. We complement this with the output of various SPIDER runs to measure the network and cost efficiency of various design schemes versus their restoration times. In the final section, we conclude with the implications of these routing schemes for near-real-time and proposed new lightpath provisioning protocols such as extended open shortest path first (E-OSPF) and optical network navigator (ONN).

Bandwidth Efficiency Versus Restoration Time

In this section, we describe techniques for routing of lightpaths in core all-optical networks in which 100% (or any desired degree of) recovery is required for a *single* network link or node failure. (Doshi et al.[1] provide an overview of different restoration schemes and proposals for new restoration protocols.) Optimal design and routing for such networks, taking the cost of failure recovery into account, is a difficult computational problem (see, for example, the work of Rajagopalan et al.[2]). The effectiveness of a proposed routing mechanism is judged not only by the resulting efficiency in the use of available wavelengths but also by its complexity of provisioning and speed of restoration in operational networks. Each of our schemes described below best fits a given grade of protection, such as "platinum" (below 50 ms), "gold" (50 to 100 ms), and "silver" (~1 to 10 s) restoration, and can be implemented by the majority of emerging optical networking protocols, such as multiprotocol lambda switching (MPλS).

For prior work on routing and design of networks using synchronous optical network (SONET) or logical rings, see Ritchie[3] and Tillerot et al.;[4] for DWDM and restoration, see Lee and Li;[5] for asynchronous transfer mode (ATM), see Irascho and Grover;[6] and for mesh network design, restoration optimization, and wavelength assignment, see Mukherjee et al.,[7] Saniee,[8] and Bouillet and Bala.[9] Mukherjee et al.[7] and Stern and Bala[10] provide general background on optical networks. Cosares et al.,[11] Doshi and Harshavardhana,[12] and Doshi, Dravida, and Harshavardhana[13] describe recent work on software tools for broadband network design.

Panel 1. Abbreviations, Acronyms, and Terms

APS—automatic protection switching
CGI—common gateway interface
CGI.pm—CGI Perl module
CPAN—Comprehensive Perl Archive Network
DWDM—dense wavelength division multiplexing
E-OSPF—extended open shortest path first
I/O—input/output
ID—identification
LP—linear programming
MPλS—multiprotocol lambda switching
NSF—National Science Foundation
OC-192—optical carrier digital signal rate of 9.953 Gb/s in a SONET system
ONN—optical network navigator
OTU—optical terminating unit
OXC—optical cross connect
PC—personal computer
Perl—Practical Extraction Report Language
SDH—synchronous digital hierarchy
SONET—synchronous optical network

The following sections describe three possible network architectures applicable to optical networks with their associated routing and protection mechanisms.

1 + 1 Dedicated Protection

In this routing scheme, node pairs that have demand are connected with a sufficient number of wavelengths on an active path and an identical number of protection wavelengths on a diversely routed protection path. Note that all wavelengths between two nodes need not follow the same path. The protection wavelengths are *dedicated* for each node pair for each wavelength, but the fibers carrying active or protection wavelengths for different node pairs may be shared, as shown in **Figure 1**. The scheme for recovery from link failure generally entails:

- Propagation delay and recognition of failure by each affected node pair (~10 ms) and
- Switching to the dedicated protection wavelength(s) (~20 to 40 ms).

Propagation delays may be eliminated by duplicate transmission on the protection paths and selection of the best signal at each destination node (this is called *1 + 1 protection*). Otherwise (that is, when no transmission occurs on the back-up path until failure), for each wavelength, the equipment used at end nodes performs an operation identical to the automatic protection switching (APS) available in the earlier SONET and synchronous digital hierarchy (SDH) hardware technologies (this is called *1 : 1 protection*). The speed of recovery is thus a few tens of milliseconds. The solution methodology for both 1 + 1 and 1 : 1 protection consists of finding a disjoint pair of active and protection paths for each demand, aggregating these to obtain the total number of wavelengths on each fiber, and then determining the cost of the overall design. Phase 1 can be carried out in at least two ways. For example, each demand can be routed on its "shortest path" (by counting hops or distances, or by using an adaptive cost metric that keeps track of the "fill" of a fiber), and dedicated protection wavelengths can be routed over the disjoint residual graph. Alternatively, all node pairs with their disjoint routes can be routed simultaneously using a cost-optimization approach.

We give a heuristic routing algorithm for the 1 + 1 or 1 : 1 protection scheme. The key idea of this algorithm is to define an iterative shadow cost for each network resource, then route each demand along its least-cost feasible path. The cost model is as follows:

1. *Cost of λ-channel.* Each λ-channel requires a channel in a fiber. To optimize fiber usage, the use of λ-channels in some existing fibers is encouraged. This is achieved with a λ-channel cost of the form
$$y_{ij}(w_{ij},\lambda) = \frac{p_f l_{ij}}{W}\left(\frac{W - a_{ij}(\lambda)w_{ij} \bmod (W)}{(a_{ij}(\lambda)w_{ij} \bmod (W) + 1)^t} + 1\right),$$
where y_{ij} is the marginal cost of unit λ-channel, l_{ij} is the length of the link, w_{ij} is the number of λ-channels in use on link (i, j), W is the number of λ-channels that can be carried by a fiber, p_f is the cost per unit length of fiber, and $a_{ij}(\lambda)$ depends on whether a wavelength converter existed. In the case of wavelength continuity (no converter at any node), when the wavelength λ is currently not available on link (i, j), $a_{ij}(\lambda) = 0$; otherwise, $a_{ij}(\lambda) = 1$ and t is set empirically. The shape of y_{ij} as a function of w_{ij} is shown in Figure 2.

2. *Cost of port.* Each end of a λ-channel requires a port on an optical cross connect (OXC) installed at the corresponding node. Each OXC may contain a finite number, X, of ports. Similar to the λ-channel cost, port cost can be given in the form
$$x_i = \frac{p_{OXC}}{X}\left(\frac{X - u_i \bmod (X)}{(u_i \bmod (X) + 1)^t} + 1\right),$$
where x_i is the marginal cost of a port at node i, u_i is the number of ports in use at node i, X is the total number of ports in an OXC, p_{OXC} is the cost per OXC, and t is selected empirically.

3. *Cost of amplifier.* Assume that an amplifier is needed for each given length, L_a, of fiber. The amplifier cost can be integrated into the λ-channel cost if we redefine the unit distance cost c_f as

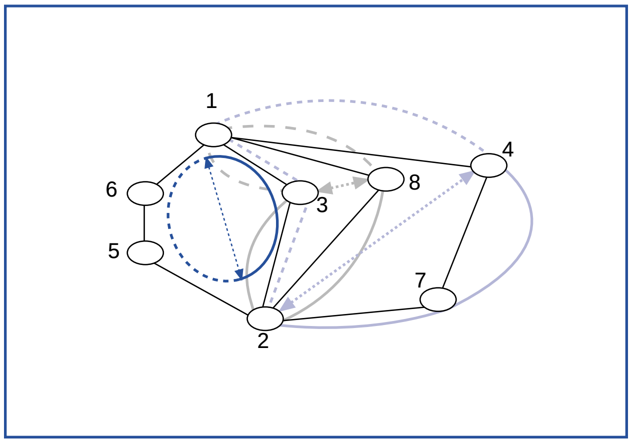


Figure 1.
Diversely routed protection (dotted line) wavelengths for demands between node pairs 1&2, 2&4, and 3&8 using active (solid line) wavelengths (both types can share fiber—for example, on links 1-3 and 2-3).

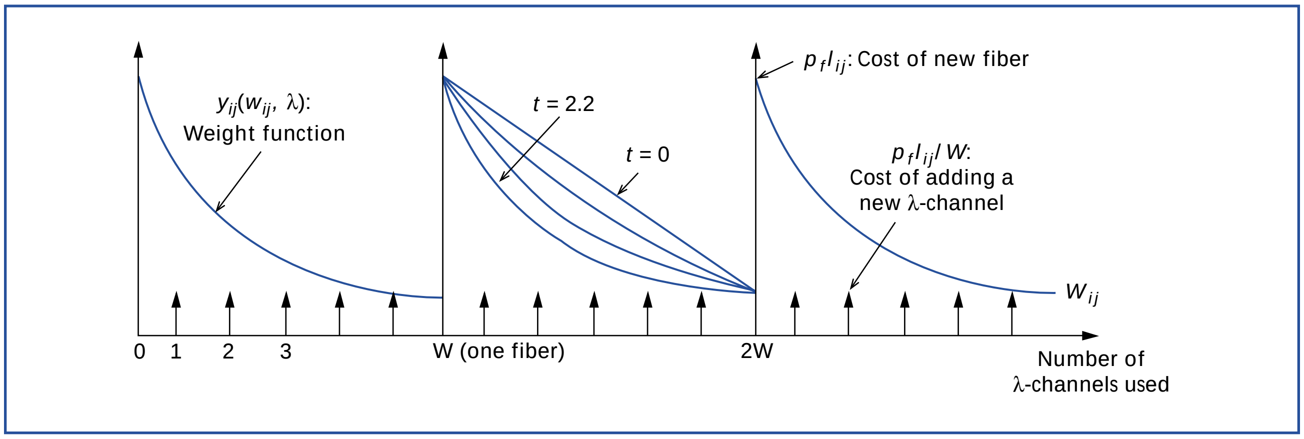


Figure 2.
Function of λ-channel cost y_{ij} based on utilization w_{ij} and iteration parameter t.

c_f = unit length cost of fiber
$+$ cost of amplifier $/ L_a$.
(Cost of regenerators is added similarly in SPIDER.)

4. *Cost of transmitters.* Since the number of transmitters needed depends only on the total number of demands rather than the routing algorithms, we ignore the cost of transmitters in the routing procedure. However, it will be counted in the total network cost computation.

Using the cost model given above, we can give the algorithm for dedicated protection as follows:

1. *Generate candidate ring list.* For each demand in the demand list, generate k shortest cycles (rings) passing through the two end nodes of the demand. A modified version of either the A*Prune algorithm[14] or the Suurballe-Tarjan algorithm[15] can be used to generate such k rings. All the generated rings are listed by their increasing lengths.
2. *List candidate path pairs for each demand.* For each ring in the ring list, generate two path pairs along the ring if both end nodes of the demand are on the ring. At most, $2R$ such path pairs for each demand can be generated, where R is the total number of rings in the ring list.
3. *Compute/update the marginal cost for each candidate path pair.* The marginal cost of a candidate path pair p is the sum of the shadow costs of all resources required by p and can be defined as

$$\Psi(p) = \sum_{(i,j)\in p} (y_{ij} + x_i + x_j).$$

4. *Order the demand list.* The demand list can be in random order.
5. *Perform one-round routing.* Do the following steps for each demand in the demand list:
   a. Remove the route assigned to the demand if it has any, and free all the resources acquired by the demand.
   b. Update the marginal cost for all candidate path pairs affected by the removed routes.
   c. Route the demand along its least-cost path pair. This includes finding the least-cost path pair and reserving all necessary resources (such as wavelengths and ports) along the path pair.
   d. Update the marginal cost for all candidate path pairs affected by the previous step.
6. *Fine-tune by loop routing.* Repeat the procedure of one-round routing until a converged routing is reached. A routing is considered to be converged if the total network cost given by the current-round routing is the same as that given by the previous-round routing.

Shared Protection Using Logical Rings

In this architecture, node pairs are grouped into logical DWDM rings, each of which carries no more than the predefined number of wavelengths per fiber for active as well as protection wavelengths. Active paths are defined for all node pairs on the same logical ring, and protection wavelengths are reserved in the complementary routes on the ring for each node pair, as shown in **Figure 3**. The number of protection wavelengths for each demand is thus the same as the number of active wavelengths used for carrying the demands allocated to each ring. However, nonoverlapping demands on the same ring can *share* protection wavelengths. For example, demands between nodes 1-3 and 3-2 can share protection wavelengths on the ring 1-3-2-5-6-1. The rings 1-4-7-2-5-6-1 and 1-3-2-5-6-1 can share fibers on links 2-5, 5-6, and 6-1. Although in principle it is possible to share protection wavelengths *across* rings using OXCs, we do not use such sharing due to the complexity of recovery when failures occur.

The scheme for recovery from failure is only somewhat more complex than dedicated protection, described above. Similar to dedicated protection, only the end nodes of each wavelength affected by the failure need to take care of the failure (~10 ms). Once a failure condition is recognized, the switch over to protection wavelengths is made. In our logical ring design solution, each demand is mapped to a single ring. Therefore, it will not be necessary to coordinate multiple rings for recovery. The single ring autorecovery takes ~50 ms, allowing for ~40-ms cross-connect remapping in the intermediate nodes. OXCs are needed for autorecovery on each logical ring and for sharing of fibers. Without fiber sharing, wavelength-selective cross connects would be adequate. Using the same shadow cost as in the previous section, the routing algorithm for logical rings is given as follows:

1. *Generate candidate ring list.* Use the same procedure as that used for the dedicated protection algorithm given in the “1 + 1 Dedicated Protection” section.
2. *List candidate paths for each demand.* For each ring in the ring list, generate two paths along the ring if both end nodes of the demand are on the ring. At most, $2R$ such paths for each demand can be generated, where R is the total number of rings in the ring list.
3. *Compute/update the marginal cost for each candidate path.* The marginal cost of a candidate path p^r retrieved from ring r is defined as

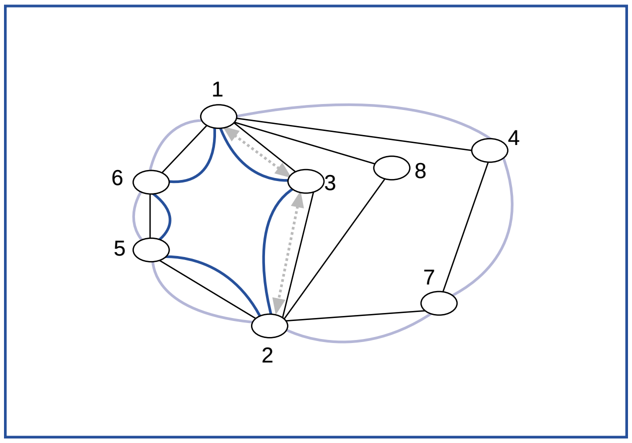


Figure 3.
Shared-protection wavelengths for demands between 1&3 and 3&2 on ring 1-3-2-5-6-1, and shared fibers on links 2-5, 5-6, and 6-1 for the rings 1-4-7-2-5-6-1 and 1-3-2-5-6-1.

$$\Psi(r,p) = \sum_{(i,j)\in p^r} (y_{ij} + x_i + x_j) + \delta(r,p) \sum_{(i,j)\in r} (y_{ij} + x_i + x_j).$$

Here, $\delta(r,p)$ is 1 if p^r is the first path to invoke a new protection channel; otherwise, it is 0.

4. *Order the demand list.* The demand list is preferred in the decreasing order of the length of its shortest rings so that the demand which invokes a larger ring is routed first.
5. *Perform one-round routing.* Do the following steps for each demand in the demand list:
   a. Let d be the demand to be routed.
   b. If d has already been assigned a route p, remove route p and free all the resources acquired by p. Assume p is originally protected by the ring protection channel r. Remove r if there are no other working channels to be protected by r.
   c. Update the marginal cost for all candidate paths affected by the removed routes.
   d. Route the demand along its least-cost ring path. This includes finding the least-cost ring path and claiming all necessary resources along the ring path.
   e. Update the marginal cost for all candidate ring paths affected by the routing procedure (step 3).
6. *Fine-tune by loop routing.* Use the same procedure as that used for the dedicated protection algo-

rithm given in the "1 + 1 Dedicated Protection" section.

Both dedicated protection and logical ring protection algorithms can also have variant versions, such as shortest-path routing or wavelength-continuity routing. The shortest-path routing version routes all demands along their shortest paths. The wavelength-continuity case can be handled by selecting the appropriate value of $a_{ij}(\lambda)$ in the cost model.

1 : *N* Protection on Single Demand

A computationally simple method for the shared-protection mesh network design can be obtained by considering shared-protection routing for single point-to-point demands. We first address only the sharing among multiple candidate paths between the source and the destination for the single demand under consideration, as shown in **Figure 4**.

However, these single-demand solutions can subsequently be combined to account for sharing across demands. The procedure can thus be used in designing a reliable network from the start ("greenfield" scenario), and it provides a fast and approximate solution to a more comprehensive mesh design to be discussed in the next subsection. We refer to this algorithm as the 1 : *N* protection algorithm in our numerical results later.

Consider the restoration-adapted version of the standard multicommodity flow problem:

$$\text{Minimize } \sum_{e \in E} \lambda_e {}^* l_e$$

$$\text{Subject to } \sum_{p' \in P_k,\, p' \neq p} x_{p'} \geq d_k \;\forall k \in K, \forall p \in P_k \qquad (1)$$

which seeks to minimize the total wavelength distance traversed in the network while protecting against single link/node failures. As explained earlier, we focus on a single source-destination pair *s*-*t*, which is assumed multiply connected by *N* node-disjoint candidate paths, each path *i* with a given cost c_i of carrying unit flow. The costs may be ordered as $c_1 \leq c_2 \leq .... \leq c_N$. For single-path failure reliability, it is easy to see that $c_N \leq c_1 + c_2$—that is, the set of candidate paths may be pruned in this manner without loss of optimality. We seek to route, at minimum cost, a total demand *D* between *s* and *t* along the paths so that service is not interrupted by single-path failure. This problem has

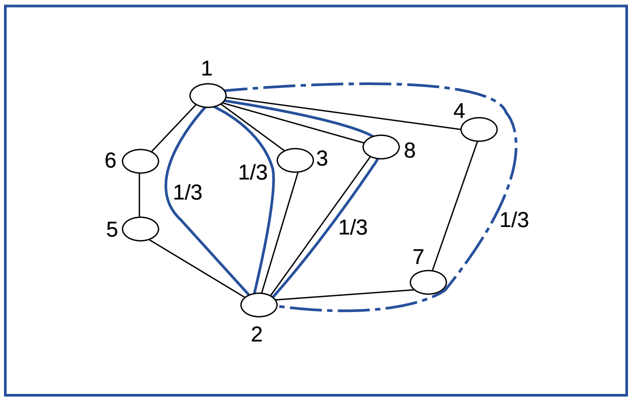


Figure 4.
Short active (solid) paths and longer restoration (dotted) path for demand between nodes 1 and 2 (fractions denote the demand carried by the respective paths).

the following simple linear programming (LP) formulation directly derived from (1) above:

$$\text{Minimize } \sum_{k=1}^{N} c_k x_k \text{ subject to } \sum_{k=1,\, k \neq j}^{N} x_k \geq D, x_k \geq 0$$
$$\forall j \in \{1,2,...,N\}. \qquad (2)$$

Letting $\sum_{k=1}^{N} x_k = D + D'$, the above LP formulation is written as a classical *knapsack problem* for fixed *D*′:

$$\text{Minimize } \sum_{k=1}^{N} c_k x_k \text{ subject to } \sum_{k=1}^{N} x_k = D + D',$$
$$0 \leq x_k \leq D' \;\; \forall j \in \{1,2,...,N\}. \qquad (3)$$

It can be shown that (3) has the following simple solution: Let $K = \lceil 1 + D/D' \rceil$.

$$x_k = D' \;\forall k < K; x_k = 0 \;\forall k > K;$$
$$x_K = D - (K - 2)D'.$$

Further, a simple search over *D*′: $D/(N - 1) \leq D' \leq D$ produces the globally optimal x_k independent of whether or not they are restricted to take integer values. The end result of this procedure is that the optimal set of chosen paths for large demands will satisfy $\frac{1}{K-1}\sum_{k=1}^{K} c_k < c_{K+1}$. This relation has the interesting interpretation that the optimal solution, independent of *D* (for large *D*), chooses to add new paths to route the demand only until the incremental additional cost of unit restoration capacity continues to decrease, beyond which it load balances among the previously chosen paths. This implies (except for integrality of flows) equal allocation of flows to the set of chosen paths. It is also possible to show that complete load balancing over all paths, that is, the choice $x_k = \lceil D / (N - 1) \rceil$, is never more than a factor of 2 in total cost from the optimal solution. Finally, all of the above conclusions can be generalized for the cases of reliability against multiple failures, individual capacity limits for each path, variable degrees of protection ranging from full backup of the affected demand to a smaller percentage, and differing fixed installation costs for each path.

Motivated by the above-described tractability of the single-demand routing problem, we propose the following heuristic solution to the more general shared mesh network design that accounts for sharing across demands. The solution is as follows:

1. Solve the routing problem for each demand sequentially using a precomputed set of paths.
2. Choose the lowest-cost $K - 1$ paths as *primary* paths for each demand.
3. Cumulate the primary flows on each link to determine its primary capacity.
4. To compute total link capacities, fail one link/node at a time and determine the maximum net flow on surviving links when affected flows are routed on their respective K^{th} paths.

This algorithm provides a limited form of sharing between demands, as well as between routes for a given demand, and produces a suboptimal approximation to the fully shared mesh network design. However, it provides a simple solution to the routing and restoration problem that is easy to compute, update with new demands, and manage when failures occur. Note that only the demands affected by a particular failed link/node need to be rerouted, and, further, the rerouting for the affected set demands is always the same, thus making fault isolation unnecessary.

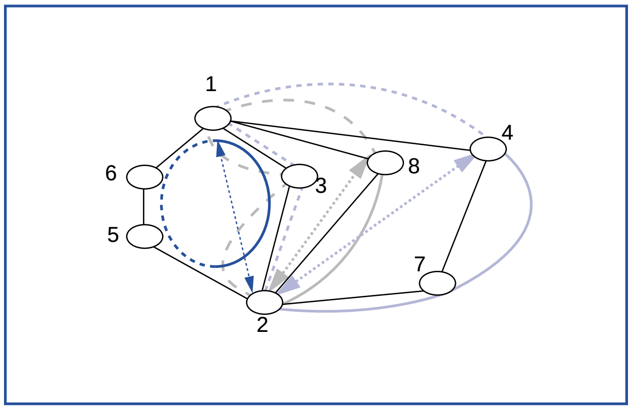


Figure 5.
Active and restoration paths and wavelengths for demands between nodes 1-2 (1-3-2 and 1-6-5-2), 2-4 (2-7-4 and 2-3-1-4), and 2-8 (2-8 and 2-3-1-8).

When the demands are small and grooming/packing efficiency is important, it is possible to adapt the heuristic to pack more efficiently by altering the choice of primary paths without losing the desirable features of the result.

Optimal Shared-Protection/Mesh Design Using Mixed-Integer Linear Programming

This architecture generalizes the shared-protection ring scheme described above without explicit partitioning into rings. As before, active and restoration paths are constructed for all node pairs, but the restoration capacities, computed as the maximum number of wavelengths required on each link when a single failure occurs, are fully shared. This scheme optimizes the redundant (or spare) capacity requirements but has increased restoration complexity compared to the ring method. **Figure 5** shows a simple example in which protection path 2-3-1-8 for demand 2-8 on path 2-8, protection path 2-3-1-4 for demand 2-4 on path 2-7-4, and the active path 1-3-2 for demand 1-2 on path 1-3-2 all can share wavelengths on edges 2-3 and 3-1.

This scheme requires careful preplanning for an efficient implementation. Each failure invokes an optimally recomputed reconfiguration map at each node, with possible wavelength conversion to avoid collision on the same set of links during path restoration events. The maps can be precomputed using optimization. The objective is to minimize the total number of wavelengths needed. The resulting maps are stored in each

OXC. Path restoration is not conditional on isolation of fault: by using multiple alternative *disjoint* protection paths for each active path, the end nodes initiate the recovery procedure once the signal loss is registered, and the disjointness of restoration routes ensures their availability when the single failure condition disables the normal route(s). Once the end nodes encounter a failure condition on a path (~10 ms), they communicate the restoration routes of the wavelengths to be restored to each intermediate OXC on alternative paths and the maps to be uploaded by each OXC (40 to 80 ms). These maps change, often only incrementally, when new demands are added. In the case of topology changes, all active and back-up routes may require updating, but topology changes surely affect other architectures as well. To reuse and extend the notation introduced in the previous sections, let

N = set of all network nodes

E = set of all network edges

K = set of all node pairs

P = set of all paths for all node pairs

D = set of all node-pair demands

d_k = demand for node pair k

x_p = flow (in units of λ) assigned to path p

P_k = set of all paths for node pair k, assumed to be (node) edge disjoint

Q_e = set of all paths that include edge e

C_e = capacity (in number of wavelengths) on edge e

l_e = distance of edge e

y_p^f = flow (in units of λ) assigned to path p when edge f is in failure mode.

To ensure that bidirectional demands are routed the same way, we restrict $(i,j) \in K$ to $i < j$. Furthermore, to reduce problem dimensionality, we include a path in P only if its end nodes have demand. To ensure that restoration from failures does not require fault isolation, we assume that paths in each P_k are disjoint for each commodity k. Therefore, when an end node pair recognize a fault in (one of) their primary path(s), the node pair initiate restoration over the remaining paths, which, due to their being disjoint from the failed path, must be operational in cases of single (link) failures.

To see how the mesh network should be designed and its demand routed using the above notation, consider the following problem:[5]

$P0$: Find $x_p \; \forall p \in P$

$$(1) \quad \sum_{p \in P_k} x_p \geq d_k, \text{ each } k \in K$$

$$(2) \quad \sum_{p \in Q_e} x_p \leq C_e, \text{ each } e \in E$$

These conditions ensure that:

1. Demand for each node pair k is met and
2. Link capacities are not exceeded.

Thus, any allocation of flow x_p to path p that meets conditions 1 and 2 is considered feasible. To avoid solutions with meandering paths—those that either are unusually long or visit the same node more than once—the set of paths P needs to be carefully generated. Examples include paths that do not exceed prespecified hop counts or distance limits between their end nodes. Further, more careful path generation reduces the problem complexity, as will be shown in the following discussion. In general, K-disjoint paths that meet path qualifications are generated for each node pair with nonzero demand for a sufficiently large K. The *mesh network routing problem* can be formulated according to a variety of criteria using the foregoing notation. For example, it may be important to route as much of the demand as possible on the shortest possible routes. In this case, the natural criterion would be the sum of wavelengths times the distance each wavelength traverses. Alternatively, it may be desired to spread the load as evenly as possible so that all the edges in the network carry more or less the same number of wavelengths. For the latter criterion, the following formulation results:

$P1$:(even_load = Λ)

Minimize Λ

Subject to:

$$(1) \quad \sum_{p \in P_k} x_p \geq d_k \; \forall k \in K$$

$$(2) \quad \sum_{p \in Q_e} x_p \leq C_e \; \forall e \in E$$

$$(3) \quad \sum_{p \in Q_e} x_p \leq \Lambda \; \forall e \in E$$

Constraint set 1 ensures that demands (in number of wavelengths) are met for each node pair, and constraint set 2 guarantees that link/DWDM capacities are not exceeded. Constraint set 3 evens out the load on the network. The same routing design problem with the minimum sum of wavelength distances as the objective function is as follows:

$P2$: (λ*distance)

Minimize $\sum_{e \in E} \lambda_e * l_e$

Subject to:

$$(1) \sum_{p \in P_k} x_p \geq d_k \; \forall k \in K$$

$$(2) \sum_{p \in Q_e} x_p \leq \lambda_e \; \forall e \in E$$

$$(3) \; \lambda_e \leq C_e \; \forall e \in E$$

Constraint set 3 for $P2$ counts the number of wavelengths used on each edge within the objective function. For the mesh network restoration problem, the same set of objective functions as $P1$ and $P2$ apply. This time, however, additional constraints needed to ensure 100% protection against failure are also provided. Let y_p^f denote the overflow assignment of wavelengths to path p when edge f is in "fail" condition. Then, the assignment of normal routes x_p and restoration routes y_p^f for the minimized sum of wavelength*distance objective (as in $P2$) is given by the solution of

$P3$: (λ*distance)

Minimize $\sum_{e \in E} \lambda_e * l_e$

Subject to:

$$(1) \sum_{p \in P_k} x_p \geq d_k \; \forall k \in K$$

$$(2) \sum_{p \in Q_e} x_p \leq C_e \; \forall e \in E$$

$$(3) \sum_{p \in Q_e} x_p \leq \lambda_e \; \forall e \in E$$

$$(4) \sum_{p \in P_k - Q_f} y_p^f \geq d_k \; \forall f \in E, k \in K$$

$$(5) \sum_{p \in Q_e - Q_f} y_p^f \leq \lambda_e \; \forall e \neq f \in E$$

$$(6) \; \lambda_e \leq C_e \; \forall f \in E$$

We refer to this formulation as the restoration problem. Note that in this formulation, a large number of reconfigurations may be necessary for each failure condition, since the overflow variables are only constrained to meet demand and not exceed edge capacities. To enforce reconfiguration only for routes affected by the failure condition, a slightly different set of constraints is needed. This formulation is given below:

$P4$: (λ*distance)

Minimize $\sum_{e \in E} \lambda_e * l_e$

Subject to:

$$(1) \sum_{p \in P_k} x_p \geq d_k \quad \forall k \in K$$

$$(2) \sum_{p \in Q_e} x_p \leq \lambda_e \quad \forall e \in E$$

$$(3) \; \lambda_e \leq C_e \quad \forall f \in E$$

$$(4) \sum_{p \in P_k - Q_f} y_p^f + \sum_{p \in P_k - Q_f} x_p \geq d_k r_k \quad \forall f \in E, \forall k \in K$$

$$(5) \sum_{p \in Q_e - Q_f} y_p^f + \sum_{p \in P_k - Q_f} x_p \leq \lambda_e \quad \forall e \neq f \in E$$

Note that in formulation $P4$ of the restoration problem, we have also allowed a more general recovery percentage r_k for each node pair k than recovery of all paths disrupted by each link failure, $r_k = 1$ or 100% recovery, which is what was done in $P3$. However, this extension adds no complexity to the problem and substantially simplifies the execution of restoration. The above formulation can be further enriched to allow for situations in which fault localization is difficult or too time consuming to achieve (this is done in SPIDER). However, these models give a sufficient description to illustrate the necessary tradeoffs between restoration efficiency and time, as described in the "Examples of Network Design and Numerical Results" section.

Software Architecture for SPIDER

As we have shown, there are a variety of design, routing, and protection schemes for optical core networks. It is likely that, in the near future, hybrids or even new schemes will be invented and implemented by Lucent Technologies and other manufacturers. Since SPIDER aims to accurately model current and future routing schemes, protection schemes, and protocols, its software needs to be highly flexible and modular. This calls for separation of application, input/output (I/O), visualization, and computational engines. To this end, the SPIDER application was built of five parts:

- A text-based (comma-separated value) front end for network and data I/O,

- A graphical (Java) front end for visualization and graphical I/O,
- A common gateway interface (CGI)-scripted Web interface in Practical Extraction Report Language (Perl) for client/server architecture,
- Back-end glue in Perl, and
- Back-end algorithm processing in C where heavy-duty optimization processing is implemented.

Evolution

At first, we solved the algorithmic problems using C, and, in some instances, we used third-party software such as MINOS[16] and CPLEX.[17] Following test runs and verification of designs, we initiated implementation of the Java front end to allow for graphical entry and visualization of the network. We implemented the original front end as a standalone Java 1 application. We were less than thrilled with the challenges of implementing a graphical interface on the Java 1 base and eventually learned that this front end was not well matched to the needs of the intended users of the application.

What the users wanted was to be able to collect their network definitions in spreadsheets using the Microsoft* Excel application and then feed the spreadsheet data to SPIDER. We also realized that delivering the application as something to install on a personal computer (PC) would leave us with configuration management problems. Our preference was to keep the application on a central server and provide access to it via Web browsers. We have been reasonably pleased with the result of that decision but have found there are still challenges due to the variety of Web browsers in use. We had read elsewhere[18] of the dangers of an application getting too close to the proprietary formats of Excel. To avoid this problem, we opted for a particularly simple format that Microsoft calls *comma-separated value (CSV)* or, simply, a "flat file" format. We believe this format will likely remain usable without problems for us as Microsoft potentially makes changes to Excel in future releases.

The next problem to solve was how to interact with the user to transfer his/her spreadsheet data over to our application on its central server. Since we did not have a lot of prior experience implementing Web-based applications, and this did not look like it should be a rare aspect of such applications, we did not want to "reinvent the wheel." We looked for standards and found RFC1867.[19] Without too much more searching, we located an inexpensive commercial product that implemented that standard in a Java applet.[20] The vendor, Infomentum, made a free evaluation period for its software available to us.

Unfortunately, we encountered problems during the evaluation. Source code for the product was not available to us, so we turned to the prospective supplier for help. They were responsive to e-mail but unable to determine the cause of our difficulties. We suspect the troubles were caused by variations in Java implementation from PC to PC. Even when the same browser is used (for example, Microsoft Internet Explorer), there are still more versions of that browser and the underlying Java implementation than we want to have to consider. The experience leaves us questioning the viability of the hoped-for market in applets,[20] at least for the case of applets sold as "black boxes" without access to the source code.

Fortunately, we found an open source implementation of RFC1867 in the CGI Perl module (CGI.pm)[21] and enough documentation to be able to figure out how to make it work.[22] The CGI.pm implementation of RFC1867 did not quite do everything that was provided by the Infomentum product, but it was good enough and seemed much more stable in our testing. Later, we decided that the application needed a visual display of its results and returned to Java for that display. This time we used Sun Microsystems' Java 2 plug-in within the browser. The libraries for graphical display generation in Java 2 were much more satisfactory, and we are hoping the users will be able to handle the "automatic" installation of the plug-in to their browsers.

Security

There are obvious concerns associated with making an application available on the Web. Deploying it on a Web server inside the Lucent firewall would block access from the world at large but would still make our application accessible to more than 100,000 Lucent associates and potential users. Since we were unprepared to vouch for everyone with computer access within Lucent, we needed a way to contain the application. Our standard Web server in Lucent's Mathematical Sciences Research Center runs on a computer that allows a user ID of "nobody." That would not do for all of our applications (such as CPLEX,[17] for example, since its license only allows it to run on a Lucent computer with a specific licensed name). Therefore, we needed a Web server that could have access to the third-party software. Our solution was to set up a new user ID, "SPIDER," and an Apache Web server on a PC running Linux* in our lab. Since the Web server is strictly for our application, it can run with the SPIDER user ID. We then needed to make sure that the SPIDER user ID is only able to access software and data we do not need to protect from the vast Lucent internal user community. This met our objective of not having to administer user accounts, but we recognize that the system is possibly more "open" than our users might want it to be.

Lessons Learned

The old adage, "Plan to throw one away," is applicable here. In fact, when learning a new language with the complexity of Java, you may be well advised to plan to throw more than one away. Time spent searching for standards and available libraries instead of "reinventing the wheel" is time well spent. "Black-box" third-party software is more challenging to live with than open source libraries. We still "have our fingers crossed" about how well Java will work out for us in the field—a maze of twisty virtual machines, all different.

The project needs to have control of its environment. The Linux base for our front end was invaluable to us. We were able to tweak the Web server configuration and update the Perl libraries with new modules from the Comprehensive Perl Archive Network (CPAN) as we needed them, without having to push for time from people outside of our project to make those updates for us. "We would like to try this new library" is not a request that typically gets high-priority attention from overburdened system administrators. The downside is that it means our project must continue to pay attention to system administration of Linux into the future. Keeping a Linux box up to date with security updates is nontrivial work.

In retrospect, we should have set up more than one instance of our application. With real users, it is impractical to make and test changes. A simple first step would be to modify all hard-coded paths.

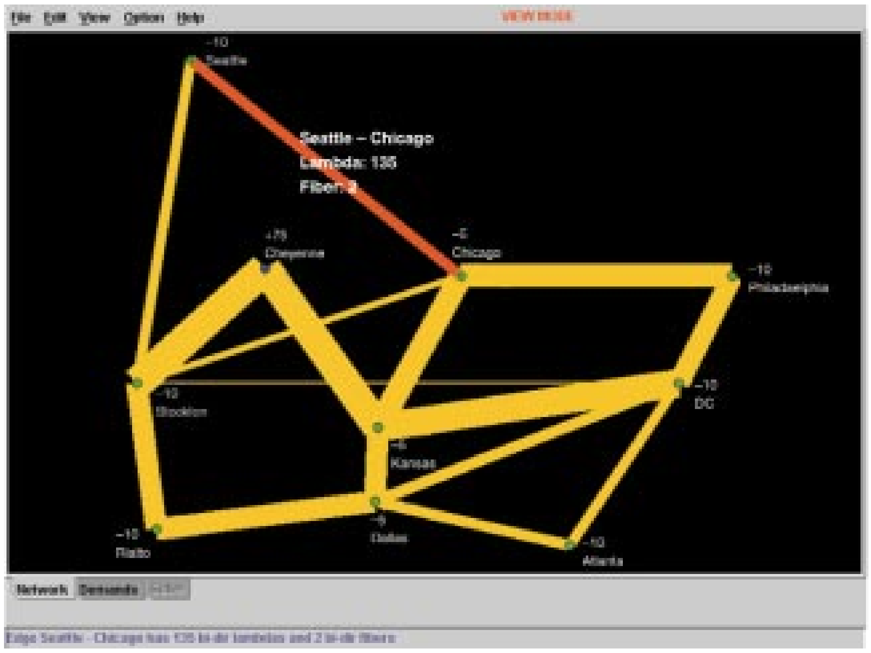
Examples of Network Design and Numerical Results

An idealized National Science Foundation (NSF) backbone network was assumed to consist of 10 nodes and 17 fiber-optic links, as shown in **Figure 6**. The variable link thickness is proportional to the number of wavelengths/fibers through it, as shown when the cursor is positioned on the Seattle-Chicago link (red). The topology and load data for a SPIDER study of this network are shown in **Tables I** and **II**. Network visualization in SPIDER is shown in **Figure 7**.

To illustrate SPIDER, this network was designed under five different protection plan alternatives, ranging from no-protection shortest-path routing (base) to a complete mesh with shared protection. The results are presented in **Table III**. This comparative study assumed the following costs and associated parameters:

80 =	number of bidirectional wavelengths per fiber pair
\$100K to \$1M =	cost of OXCs without any ports/plug-ins
256 =	size of OXC (maximum number of wavelength-pair terminations)
\$5K =	cost of optical terminating unit (OTU) used for signal regeneration and wavelength conversion
\$1.5K =	cost of general-purpose 10G fiber pair per km (including amplifiers)
\$100K =	cost of a pair of DWDM multiplexers
\$50K =	cost of regenerators per fiber pair
600 km =	regeneration distance

Our experiments assumed the presence of OXCs at each node, which are almost certainly required for robustness and flexibility in the core of all-optical networks. Under this assumption, mesh-restoration-based shared-protection architectures seem to be the most cost effective and efficient. From the standpoint of recovery time, dedicated protection solutions, for



Edge Seattle - Chicago has 135 bi-dir lambdas and 2 bi-dir fibers

Figure 6.
An idealized NSF backbone network designed for 5 to 20 OC-192 circuits between all node pairs.

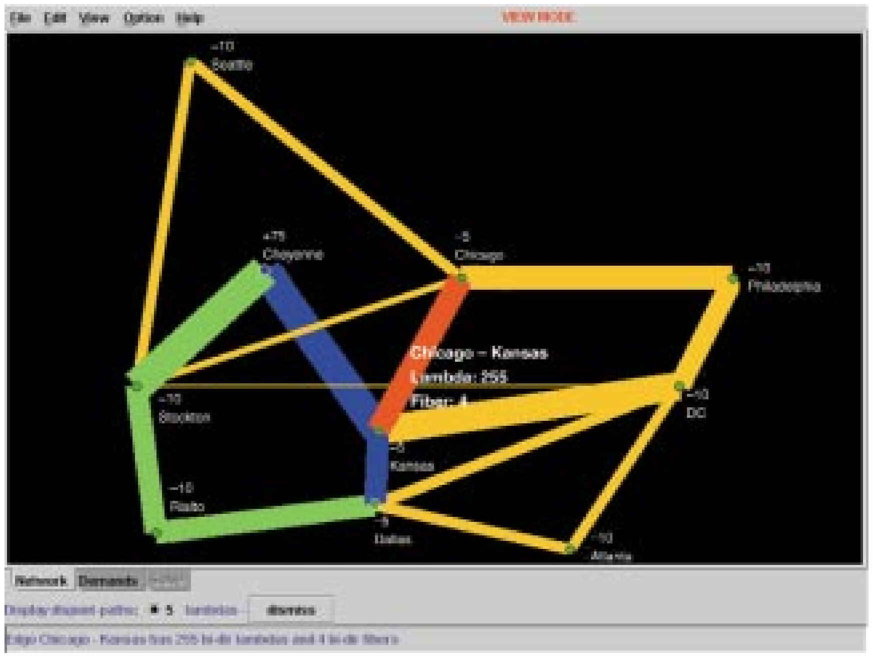
Table I. Fiber network topology with distances and spare capacity on each link.

Node	Node	Distance (km)	Spare λs	Node	Node	Distance (km)	Spare λs
Seattle	Chicago	1,200	0	Dallas	Kansas	300	0
Seattle	Stockton	800	0	Dallas	D.C.	800	0
Stockton	Chicago	1,400	0	Dallas	Atlanta	600	0
Stockton	Cheyenne	600	0	Chicago	Kansas	300	0
Stockton	D.C.	2,000	0	Kansas	D.C.	600	0
Stockton	Rialto	300	0	Chicago	Philadelphia	800	0
Rialto	Dallas	800	0	Philadelphia	D.C.	150	0
Cheyenne	Kansas	400	0	D.C.	Atlanta	800	0

the immediate future, have an advantage over alternative designs, with the shared-protection ring-based designs closely behind. Mesh restoration, however, is likely to be the architecture of choice in the future due to its greater flexibility and faster provisioning. We also observed that the solutions differ substantially in their efficiency of use of the wavelength resource (λ-efficiency), which is the total number of wavelengths (or λ*km) used to route demands divided by the total number of wavelengths (or λ*km) needed for both routing and 100% protection against single failures (row 5 in Table III). Thus, we observe a

Table II. Projected node-pair demands in units of OC-192 (demand is assumed symmetric; the lower triangular portion is not shown).

	Seattle	Stockton	Rialto	Cheyenne	Chicago	Kansas	Dallas	Philadelphia	D.C.	Atlanta
Seattle		20	20	10	10	10	10	20	20	15
Stockton			20	10	10	10	10	20	20	15
Rialto				10	10	10	10	20	20	15
Cheyenne					5	5	5	10	10	10
Chicago						10	10	20	20	20
Kansas							5	10	10	10
Dallas								10	20	10
Philadelphia									20	20
D.C.										20
Atlanta										



Edge Chicago - Kansas has 255 bi-dir lambdas and 4 bi-dir fibers

Figure 7.
Network visualization in SPIDER showing how each lightpath (blue) is routed together with its protection (possibly shared) lightpath (green).

reverse relationship between efficiency and cost ratio (row 12 in Table III) on the one hand and recovery speed on the other, which leads to natural differentiation of lightpath services based on the required grade of protection.

Conclusions

SPIDER is a software tool for network design cost optimization and comparative routing as well as protection performance evaluation of opaque and transparent optical core networks. It currently includes

Table III. Comparison of five distinct designs using SPIDER.

	No protection		1 + 1 protection		1 : *N* protection		Shared-protection ring		Shared-protection mesh	
Equipment/cost	Number of units	Cost ($)	Number of units	Cost ($)	Number of units	Cost ($)	Number of units	Cost ($)	Number of units	Cost ($)
Bidirectional λs	1,215	N/A	3,081	N/A	3,117	N/A	2,739	N/A	1,967	N/A
Bidirectional λ*km	688K	N/A	1,837K	N/A	1,799K	N/A	1,687K	N/A	1,270K	N/A
λ-efficiency	**100%**		**~40%**		**~40%**		**~40%**		**~60%**	
Optical cross connects	20	2,000K	34	3,400K	34	3,400K	30	3,000K	25	2,500K
Bidirectional OTUs	3,640	18,200K	7,372	36,860K	7,444	37,220K	6,688	33,440K	5,144	25,720K
DWDM MUX pairs	23	2,300K	43	4,300K	45	4,500K	39	3,900K	27	2,700K
Bidirectional fiber km	14,200	21,300K	26,450	39,675K	27,400	41,100K	24,450	36,675K	16,700	25,050K
Regenerators	10	500K	19	950K	20	1,000K	18	900K	12	600K
Total costs	N/A	44,300K	N/A	85,185K	N/A	87,220K	N/A	77,915K	N/A	56,570K
Cost ratio	**1**		**1.9**		**1.9**		**1.7**		**1.2**	
Operational complexity	None		Easy		Moderate		Moderate		Complex	
Real-time provisioning	Simple		Impractical		Easy		Easy		Easy	
Restoration time (including end-to-end propagation delays)	N/A		~10 to 100 ms		~50 to 100 ms		~50 to 100 ms		~50 to 200 ms	

DWDM – Dense wavelength division multiplexing
MUX – Multiplexer
OTU – Optical translator unit

novel and very efficient implementations of path-based restoration algorithms for wavelength routing and protection under single-physical-layer network failures. The modularized Java-based visualization, the browser-based graphical user interface (GUI), and the library of routing and network design engines in C/C++ provide a flexible platform from which to evaluate and compare different designs and capacity expansions as well as make informed decisions for specific networks. The flexible linkage between the routing/design engines and the visualization modules makes it possible to add new routing engines easily as needed. As an example, we expect to add the specific instances of algorithms used in Lucent's centralized (SoftWave) and decentralized (ONN) routing schemes to SPIDER in the near future.

The examples in the previous section illustrate the kinds of tradeoffs that exist for routing and protection in next-generation all-optical networks. As the degree of protection sharing increases from purely 1 + 1 dedicated protection, the network efficiency increases and the overall network cost decreases, but intelligent near-real-time provisioning of lightpaths will be needed to take advantage of this added efficiency. The increase in restoration time is moderate and well within the accepted thresholds in optical networking. Of course, for any specific lightpath and grade of protection, the appropriate protection will be utilized. These are near-real-time issues that link SPIDER to lightpath provisioning systems, such as SoftWave, with significant implications for emerging applications, such as bandwidth trading.

The most recent additions to SPIDER include two modules for transparent and partially transparent routing with use of wavelength-selective cross connects and an optimization module for ultralong-reach transport systems. Preliminary studies show "selective transparency" to hold much promise for post-opaque optical design in future generations of networks.

Acknowledgments

We wish to thank Debasis Mitra of Bell Labs as well as K. G. Ramakrishnan and Eric Bouillet, formerly of Bell Labs, for their substantial contributions to the SPIDER effort; Thomas Mueller of Lucent's Optical Networking Group as well as Deirdre Doherty and Mohcene Mezhoudi of Bell Labs Advanced Technologies for providing much perspective on network provider needs; and Bharat Doshi of Bell Labs for helpful and detailed comments on an earlier draft of this paper.

*Trademarks

Java is a trademark of Sun Microsystems, Inc.

Linux is a registered trademark of Linus Torvalds.

Microsoft is a registered trademark of Microsoft Corporation.

References

1. B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang, "Optical Network Design and Restoration," *Bell Labs Tech. J.*, Vol. 4, No. 1, Jan.–Mar. 1999, pp. 58–84.
2. B. Rajagopalan, D. Pendarakis, D. Saha, R. S. Ramamoorthy, and K. Bala, "IP over Optical Networks: Architectural Aspects," *IEEE Commun. Mag.*, Vol. 38, No. 9, Sept. 2000, pp. 94–102.
3. G. R. Ritchie, "SONET Lays the Roadbed for Broadband Networks," *Networking Management*, Vol. 8, No. 3, Mar. 1990, pp. 30–35.
4. F. Tillerot, E. Didelet, A. Daviaud, and G. Claveau, "Efficient Network Upgrade Based on a WDM Optical Layer with Automatic Protection Switching," *Optical Fiber Commun. Conf. (OFC '98) Tech. Digest*, San Jose, Calif., Feb. 22–27, 1998, pp. 296–297.
5. K.-C. Lee and V. O. K. Li, "A Circuit Rerouting Algorithm for All-Optical Wide-Area Networks," *Proc. IEEE Conf. on Computer Commun. (INFOCOM '94)*, Toronto, June 1994, pp. 954–961.
6. R. Iraschko and W. Grover, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks," *IEEE/ACM Trans. on Networking*, Vol. 6, No. 3, June 1998, pp. 325–336.
7. B. Mukherjee, S. Ramamurthy, D. Banerjee, and A. Mukherjee, "Some Principles for Designing a Wide-Area Optical Network," *Proc. IEEE Conf. on Computer Commun. (INFOCOM '94)*, Toronto, June 12–16, 1994, pp. 110–119.
8. I. Saniee, "Optimal Routing Designs in Self-Healing Communications Networks," *Intl. Trans. in Operations Research*, Vol. 3, No. 2, Apr. 1996, pp. 187–195.
9. K. Bala, E. Bouillet, and G. Ellinas, "Benefits of Minimal Wavelength Interchange in WDM Rings," *Optical Fiber Commun. Conf. (OFC '97) Tech. Digest*, Dallas, Tex., Feb. 16–21, 1997, pp. 120–121.
10. T. E. Stern and K. Bala, *Multiwavelength Optical Networks: A Layered Approach*, Addison-Wesley, Reading, Mass., 1999.
11. S. Cosares, D. N. Deutsch, I. Saniee, and O. J. Wasem, "SONET Toolkit: A Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks," *Interfaces*, Vol. 25, No. 1, Jan.–Feb. 1995, pp. 20–40.
12. B. T. Doshi and P. Harshavardhana, "Broadband Network Infrastructure of the Future: Roles of Network Design Tools in Technology Deployment Strategies," *IEEE Commun. Mag.*, Vol. 36, No. 5, May 1998, pp. 60–71.
13. B. T. Doshi, S. Dravida, and P. Harshavardhana, "Overview of INDT: A New Tool for Next-Generation Network Design," *Proc. IEEE Global Telecommun. Conf. (GLOBECOM '95)*, Vol. 3, Singapore, Nov. 13–17, 1995, pp. 1942–1946.
14. G. Liu and K. G. Ramakrishnan, "A*Prune: An Algorithm for Finding *K* Shortest Paths Subject to Multiple Constraints," *Proc. IEEE Conf. on Computer Commun. (INFOCOM '01)*, Anchorage, Alas., Apr. 22–26, 2001.
15. J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, Vol. 14, No. 2, Summer 1984, pp. 325–336.
16. <http://www.sbsi-sol-optimize.com>.
17. <http://www.ilog.com/products/cplex/>.
18. <http://www.fourmilab.ch/hackdiet/palm/>.
19. E. Nebel and L. Masinter, "Form-Based File Upload in HTML," IETF RFC 1867, Nov. 1995, <http://www.ietf.org/rfc/rfc1867.txt>.
20. <http://www.infomentum.com/activefile/>.
21. <http://www.wiley.com/compbooks/stein/>.
22. L. Stein, *Official Guide to Programming with CGI.pm*, John Wiley, New York, 1998.

(Manuscript approved April 2001)

R. DREW DAVIS, a member of technical staff in the Mathematics of Networks and Systems Research Department at Bell Labs in Murray Hill, New Jersey, is working on software design and implementation for telecommunication networks. He holds a B.S. in industrial engineering and operations research from Cornell University in Ithaca, New York, and an M.S. in computer, informa-

tion, and control engineering from the University of Michigan in Ann Arbor.

KRISHNAN KUMARAN is a member of technical staff in the Mathematics of Networks and Systems Research Department at Bell Labs in Murray Hill, New Jersey. He holds a B.Tech. degree in mechanical engineering from the Indian Institute of Technology in Madras and a Ph.D. in physics from Rutgers University in New Brunswick, New Jersey. Dr. Kumaran's interests are in modeling, analysis, and simulation of resource management and scheduling issues in communication networks.

GANG LIU was formerly a member of technical staff in the Mathematics of Networks and Systems Research Department at Bell Labs in Murray Hill, New Jersey, where he worked on optical network design and planning, routing algorithms, optimization techniques, and economic modeling and strategy analysis for telecommunication networks. He holds a B.S. in mechanical engineering from China University of Science and Technology in Hefei, an M.S. in optics from Peking University in Beijing, and an M.S. in computer science from Columbia University in New York.

IRAJ SANIEE, a technical manager in the Mathematics of Networks and Systems Research Department at Bell Labs in Murray Hill, New Jersey, holds B.A. and M.A. degrees in mathematics and a Ph.D. in operations research and control from Cambridge University in England. Dr. Saniee's current research projects are in approximation and optimization techniques, data networking, algorithms for routing and protection in optical networks, multiscaling models of data networks, VoIP traffic and congestion modeling, and network design. He was the leader of the team that received INFORM's Franz Edelman Runner-Up Award in 1994. ◆