Appendix A

Suggestions for RFI/RFP Network Design Exercises

Carriers often issue a network design exercise as part of a Request for Information (RFI) or a Request for Proposal (RFP). System vendors perform the network designs using their respective equipment to provide carriers with information regarding architecture, technology, and pricing. To streamline the design process and to enable carriers to glean relevant information when comparing results, some suggestions to assist carriers in preparing a design exercise are provided here.

1. Provide all data (e.g., nodes, links, demands) in text files or in spreadsheets. Do not require any manual entry of data by the vendors, as this is likely to lead to errors.

2. Provide the latitude/longitudes of the network nodes. Most design tools can use this information to position nodes on the screen, to help visualize the design.

3. If demand sets for multiple time periods are provided, specify whether the demand sets are incremental or cumulative (e.g., are the demands in set #2 added to the demands that already exist from set #1, or does set #2 represent all of the demands). Additionally, when adding demands to the network in subsequent time periods, specify whether the demands already in the network need to be kept fixed, or whether an optimization can be performed over all of the demands.

4. If the design exercise is for extended-reach technology, then fiber span information should be provided, including fiber type, span distances, and span losses. (Information such as PMD can be helpful as well, if available.)

5. Ideally, there would be no more than three design exercises:
    a. A baseline demand set for which the design should be optimized
    b. A modified demand set, where some percentage of the baseline demands have different sources/destinations. The design is run using the equipment configuration that was chosen for the baseline design. For example, if an OADM-only architecture is being used, the orientation of the OADMs selected for the baseline design must be used in the modified design. This tests the forecast tolerance of the architecture.
    c. A projected growth demand set, to test the scalability of the design
6. Be specific about what type of protection is desired (e.g., is shared protection suitable, or must dedicated protection be used).
7. If the exercise includes subrate demands, specify whether grooming/routing devices are required at all nodes, or whether traffic backhauling is acceptable. Additionally, be specific about what types of demands can be muxed or groomed together in a wavelength (e.g., what services, what protection types).
8. Provide some guidelines regarding the routing to ensure that comparisons across designs are valid. However, forcing all connections to always use the shortest path or explicitly specifying a path for each connection is too constrictive. It is preferable to specify a guideline such as the routed path for each connection should be no longer than P% longer than the shortest possible path.
9. Request design output in a specified format so that comparisons across system vendors can be readily performed.

Appendix B
C-Code For Routing Routines

The C-code for several useful routing functions is provided in this appendix. The first set of routines uses the Breadth-First-Search method to find a shortest path between a source and a destination. The code for these routines follows the algorithm outlined in [Bhan99]. The second set of routines finds the K-shortest paths between a source and a destination. This code follows the equivalence method of [HeMS03]. The final set of routines finds N mutually disjoint paths between a source and a destination. This follows the algorithm outlined in [Bhan99]. Various parameters can be set to indicate whether the paths should be link-disjoint or link-and-node-disjoint. In scenarios where N mutually disjoint paths do not exist, the function can be used to find the N maximally disjoint paths.

For the most part, memory is pre-allocated for the routines, as this tends to run faster. The maximum size of the network can be adjusted if necessary by redefining the appropriate parameters, which appear at the start of the code.

A small *main* function is provided to demonstrate how to specify the network topology and how to call the three major routines.

Minimal amount of error checking has been added.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>

/* Adjust these parameters as needed */
#define MaxNodes 250
#define MaxLinks 1000
#define MaxNodeName 30
#define MaxNodeDegree 20
#define MaxPathHops 150

#define FALSE 0
#define TRUE 1

/* when performing graph transformations, need to add dummy nodes and links */
#define MaxNodesWithDummies MaxNodes+MaxPathHops
#define MaxLinksWithDummies MaxLinks+2*MaxPathHops

const double CommonNodePenalty = 1500000.0;       /* make greater than sum of link distances */
const double CommonLinkPenalty = 160000000.0;     /* make much greater than
                                                        CommonNodePenalty */
const double INFINITY = 17000000000.0;  /* make much greater than CommonLinkPenalty */
const double SMALL = .0001;  /* small number relative to link distances */

typedef unsigned short USHORT ;
typedef char BOOL ;

typedef struct tagNodeT {
    charName[MaxNodeName+1];
    USHORT    OutgoingLinks[MaxNodeDegree];
    USHORT    IncomingLinks[MaxNodeDegree];
    USHORT    NumOutgoingLinks;
    USHORT    NumIncomingLinks;
    } NodeT, *NodeTP;

typedef struct tagLinkT {
    USHORT          LinkNode1; /* node at one end of link */
    USHORT          LinkNode2; /* node at other end of link */
    USHORT          Status; /* active link or not (true or false) */
    double          Length; /* any metric could be used here */
    } LinkT, *LinkTP;
```

```
typedef struct tagNetworkT {
    NodeT       Nodes[MaxNodesWithDummies];
    USHORT      NumNodes;
    LinkT       Links[MaxLinksWithDummies];
    USHORT      NumLinks;
    } NetworkT, *NetworkTP;

typedef struct tagPathT{
    USHORT      PathHops[MaxPathHops];
    USHORT      NumPathHops;
    double      PathDistance;
    } PathT, *PathTP;

USHORT PredecessorNode[MaxNodesWithDummies];
USHORT PredecessorLink[MaxNodesWithDummies];
double NodeDistance[MaxNodesWithDummies];
NetworkT TempNetwork;   /* use for graph transformations */

BOOL BreadthFirstSearchShortestPath (NetworkTP NetworkP, USHORT SourceNode,
    USHORT DestinationNode, PathTP ShortestPath);
BOOL BreadthFirstSearchRelax (NetworkTP NetworkP, USHORT NodeA, USHORT Link,
    USHORT FinalDestination);
USHORT KShortestPaths (NetworkTP NetworkP, USHORT Source, USHORT Dest, USHORT K,
    PathTP KPaths);
USHORT NShortestDiversePaths (NetworkTP NetworkP, USHORT Source, USHORT Destination,
    USHORT NumDisjointPaths, BOOL LinkDisjointOnly, BOOL CommonLinksAllowed,
    BOOL CommonNodesAllowed, PathTP NPaths);
USHORT AddLinkToTopology (NetworkTP NetworkP, USHORT Node1, USHORT Node2,
    double Length, BOOL ReverseLinkStatus);


main ()
{
    USHORT n;
    NetworkT network;
    PathT Paths[10];

    /* create a simple network to demonstrate the functions */
    strcpy(network.Nodes[0].Name, "A");
    strcpy(network.Nodes[1].Name, "B");
    strcpy(network.Nodes[2].Name, "C");
    strcpy(network.Nodes[3].Name, "D");
    strcpy(network.Nodes[4].Name, "Z");
    network.NumNodes = 5;
    network.NumLinks = 0;
```

```
    for (n=0; n<network.NumNodes; n++)
        network.Nodes[n].NumIncomingLinks = network.Nodes[n].NumOutgoingLinks = 0;

    AddLinkToTopology(&network, 0, 1, 10.0, TRUE);
    AddLinkToTopology(&network, 1, 2, 10.0, TRUE);
    AddLinkToTopology(&network, 2, 3, 10.0, TRUE);
    AddLinkToTopology(&network, 3, 4, 10.0, TRUE);
    AddLinkToTopology(&network, 0, 2, 10.0, TRUE);
    AddLinkToTopology(&network, 2, 4, 10.0, TRUE);

    BreadthFirstSearchShortestPath(&network, 0, 4, &Paths[0]);

    KShortestPaths(&network, 0, 4, 10, Paths);

    NShortestDiversePaths(&network, 0, 4, 3, FALSE, TRUE, TRUE, Paths);
}



/*****************************************************************/
/*      Code for Breadth First Search Shortest Paths             */
/*      Follows algorithm of [Bhan99]                            */
/*      Finds the shortest path from source to destination       */
/*      Returns the shortest path in ShortestPathP               */
/*      Returns TRUE/FALSE depending on whether a path exists    */
/*****************************************************************/

BOOL BreadthFirstSearchShortestPath (NetworkTP NetworkP, USHORT SourceNode,
    USHORT DestinationNode, PathTP ShortestPathP)
{
    USHORT n, m, numNodesOnListCurrent, numNodesOnListNew, node, link;
    USHORT nodesOnList1[MaxNodesWithDummies], nodesOnList2[MaxNodesWithDummies];
    USHORT* currentNodeList;
    USHORT* newNodeList;
    USHORT* temp;
    BOOL addedNodeToList1[MaxNodesWithDummies];
    BOOL addedNodeToList2[MaxNodesWithDummies];
    BOOL* currentAdded;
    BOOL* newAdded;
    BOOL* temp2;
    PathT tempPath;
```

```
for (n = 0; n < NetworkP->NumNodes; n++) {
    NodeDistance[n] = INFINITY;
    PredecessorNode[n] = PredecessorLink[n] = USHRT_MAX;
    addedNodeToList1[n] = addedNodeToList2[n] = FALSE;
    }

NodeDistance[SourceNode] = 0.0;
numNodesOnListCurrent = 1;
nodesOnList1[0] = SourceNode;

currentNodeList = nodesOnList1;
newNodeList = nodesOnList2;
currentAdded = addedNodeToList1;
newAdded = addedNodeToList2;

while (numNodesOnListCurrent > 0) {
    numNodesOnListNew = 0;
    for (n = 0; n < numNodesOnListCurrent; n++) {
        node = currentNodeList[n];
        currentAdded[node] = FALSE;
        for (m = 0; m < NetworkP->Nodes[node].NumOutgoingLinks; m++) {
            link = NetworkP->Nodes[node].OutgoingLinks[m];
            if (!NetworkP->Links[link].Status)
                continue;
            if (!BreadthFirstSearchRelax(NetworkP, node, link, DestinationNode))
                continue;
            if ((!newAdded[NetworkP->Links[link].LinkNode2]) &&
                (NetworkP->Links[link].LinkNode2 != DestinationNode)) {
                    newNodeList[numNodesOnListNew] = NetworkP->Links[link].LinkNode2;
                    numNodesOnListNew++;
                    newAdded[NetworkP->Links[link].LinkNode2] = TRUE;
                    }
            }
        }
    numNodesOnListCurrent = numNodesOnListNew;
    temp = currentNodeList; currentNodeList = newNodeList; newNodeList = temp;
    temp2 = currentAdded; currentAdded = newAdded; newAdded = temp2;
    }

ShortestPathP->NumPathHops = 0;
ShortestPathP->PathDistance = NodeDistance[DestinationNode];

if (NodeDistance[DestinationNode] > (INFINITY-SMALL))
    return(FALSE);
```

```
    /* the predecessor path traces from the destination back to the root */
    /*    reverse it to have it start at root  */
    node = DestinationNode;
    while (node != SourceNode) {
        if (ShortestPathP->NumPathHops >= MaxPathHops) {
            printf("Need to increase MaxPathHops\n");
            exit(-1);
            }
        link = PredecessorLink[node];
        tempPath.PathHops[ShortestPathP->NumPathHops] = link;
        node = PredecessorNode[node];
        ShortestPathP->NumPathHops++;
        }

    for (m=0; m<ShortestPathP->NumPathHops; m++)  /* reverse the order */
        ShortestPathP->PathHops[m] = tempPath.PathHops[(ShortestPathP->NumPathHops)-m-1];

    return(TRUE);
}


BOOL BreadthFirstSearchRelax (NetworkTP NetworkP, USHORT NodeA, USHORT Link,
    USHORT FinalDestination)
{
    USHORT nodeB;
    double newDistanceAB;

    nodeB = NetworkP->Links[Link].LinkNode2;
    newDistanceAB = NodeDistance[NodeA] + NetworkP->Links[Link].Length;

    if ((NodeDistance[nodeB] > (newDistanceAB + SMALL)) &&
        (NodeDistance[FinalDestination] > (newDistanceAB + SMALL))) {
            NodeDistance[nodeB] = newDistanceAB;
            PredecessorNode[nodeB] = NodeA;
            PredecessorLink[nodeB] = Link;
            return(TRUE);
            }

    return(FALSE);
}
```

```
/*********************************************************************/
/*      Code for K Shortest Paths between a source and destination   */
/*      Follows equivalence method of [HeMS03]                       */
/*      The K paths are returned in KPaths                           */
/*      The function returns the number of paths actually found      */
/*********************************************************************/

#define MaxEquivalences MaxNodes
#define MaxSplit MaxNodes
#define NodeType 0
#define LinkType 1

typedef struct tagEquivalenceClassT{
    charEquivalenceType;
    PathT    PrefixPath;   /* source to most recent split */
    PathT    SuffixPath;   /* split to next split (or to end); Don't track distance for suffix */
    PathT    ShortestPath;
    USHORT     FirstLink[MaxSplit];
    USHORT     NumFirstLinks;
    USHORT     SplitNode;
    } EquivalenceClassT, *EquivalenceClassTP;

EquivalenceClassT Equivalences[MaxEquivalences];

void CreatePathFromEquivalence (EquivalenceClassTP EquivalenceP, double Distance,
    PathTP PathP);
void FindBestPathInEquivalence (NetworkTP NetworkP, EquivalenceClassTP EquivalenceP,
    USHORT Dest);
void UpdateEquivalences (NetworkTP NetworkP, EquivalenceClassTP Equivalences,
    USHORT* NumEquivalences, USHORT BestPath, USHORT Dest);


USHORT KShortestPaths (NetworkTP NetworkP, USHORT Source, USHORT Dest, USHORT K,
    PathTP KPaths)
{
    USHORT j, n, bestPath, numEquivalences, firstLink;
    double minDist;

    for (j=0; j<K; j++) {
        KPaths[j].NumPathHops = 0;
        KPaths[j].PathDistance = 0.0;
        }

    if (Source == Dest)
        return(K);

    /* first find shortest path */
```

```
    if (!BreadthFirstSearchShortestPath(NetworkP, Source, Dest, &KPaths[0]))
        return (0);  /* didn't find any paths from source to dest */

    if (K == 1)
        return(1);

    firstLink = KPaths[0].PathHops[0];

    Equivalences[0].EquivalenceType = NodeType;
    Equivalences[0].FirstLink[0] = firstLink;
    Equivalences[0].NumFirstLinks = 1;
    Equivalences[0].PrefixPath.NumPathHops = 0;
    Equivalences[0].PrefixPath.PathDistance = 0.0;
    Equivalences[0].SuffixPath.NumPathHops = 0;
    Equivalences[0].SuffixPath.PathDistance = 0.0;
    Equivalences[0].SplitNode = Source;

    Equivalences[1].EquivalenceType = LinkType;
    Equivalences[1].FirstLink[0] = firstLink;
    Equivalences[1].NumFirstLinks = 1;
    Equivalences[1].PrefixPath.NumPathHops = 0;
    Equivalences[1].PrefixPath.PathDistance = 0.0;
    Equivalences[1].SuffixPath = KPaths[0];
    Equivalences[1].SplitNode = Source;
    numEquivalences = 2;
    FindBestPathInEquivalence(NetworkP, &Equivalences[0], Dest);
    FindBestPathInEquivalence(NetworkP, &Equivalences[1], Dest);

    for (j=1; j<K; j++) {
        minDist = INFINITY; bestPath = USHRT_MAX;
        for (n=0; n<numEquivalences; n++) {
            /* could use a heap to make this faster */
            if (Equivalences[n].ShortestPath.PathDistance < (minDist-SMALL)) {
                bestPath = n;
                minDist = Equivalences[n].ShortestPath.PathDistance;
                }
            }
        if (bestPath == USHRT_MAX)
            break;

        CreatePathFromEquivalence(&Equivalences[bestPath], minDist, &KPaths[j]);
        if (j < (K-1))
            UpdateEquivalences(NetworkP, Equivalences, &numEquivalences, bestPath, Dest);
        }
    return(j);
}
```

```
void CreatePathFromEquivalence (EquivalenceClassTP EquivalenceP, double Distance,
    PathTP PathP)
{
    USHORT h, numHops;

    *PathP = EquivalenceP->PrefixPath;
    numHops = PathP->NumPathHops;

    if (EquivalenceP->EquivalenceType == LinkType) {
        PathP->PathHops[numHops] = EquivalenceP->FirstLink[0];
        numHops++;
        }

    for (h=0; h<EquivalenceP->ShortestPath.NumPathHops; h++) {
        PathP->PathHops[numHops] = EquivalenceP->ShortestPath.PathHops[h];
        numHops++;
        }

    PathP->NumPathHops = numHops;
    PathP->PathDistance = Distance;
}

void FindBestPathInEquivalence (NetworkTP NetworkP, EquivalenceClassTP EquivalenceP,
    USHORT Dest)
{
    USHORT h, n, linkID, nodeID, splitNode;
    double minDist, distToAdd;
    PathT tempPath;

    /* make a copy of the network because will perform graph transformations */
    TempNetwork = *NetworkP;

    EquivalenceP->ShortestPath.PathDistance = INFINITY;

    /* eliminate nodes along prefix path so don't get loops */
    for (h=0; h<EquivalenceP->PrefixPath.NumPathHops; h++) {
        linkID = EquivalenceP->PrefixPath.PathHops[h];
        nodeID = TempNetwork.Links[linkID].LinkNode1;
        for (n=0; n<TempNetwork.Nodes[nodeID].NumIncomingLinks; n++) {
            linkID = TempNetwork.Nodes[nodeID].IncomingLinks[n];
            TempNetwork.Links[linkID].Status = FALSE;
            }
        }

    distToAdd = EquivalenceP->PrefixPath.PathDistance;
```

```
    if (EquivalenceP->EquivalenceType == NodeType) {
        for (n=0; n<EquivalenceP->NumFirstLinks; n++) {
            linkID = EquivalenceP->FirstLink[n];
            TempNetwork.Links[linkID].Status = FALSE;
            }
        BreadthFirstSearchShortestPath(&TempNetwork, EquivalenceP->SplitNode, Dest,
            &(EquivalenceP->ShortestPath));
        }
    else { /* LinkType */
        /* kick out the vertex node also */
        nodeID = EquivalenceP->SplitNode;
        for (n=0; n<TempNetwork.Nodes[nodeID].NumIncomingLinks; n++) {
            linkID = TempNetwork.Nodes[nodeID].IncomingLinks[n];
            TempNetwork.Links[linkID].Status = FALSE;
            }

        linkID = EquivalenceP->FirstLink[0];
        distToAdd += NetworkP->Links[linkID].Length;

        splitNode = NetworkP->Links[linkID].LinkNode2;
        minDist = INFINITY;
        for (h=1; h<EquivalenceP->SuffixPath.NumPathHops; h++) {
            /* if there are many hops, [HeMS03] has a method to speed it up */
            linkID = EquivalenceP->SuffixPath.PathHops[h];
            if (!TempNetwork.Links[linkID].Status)
                continue;
            TempNetwork.Links[linkID].Status = FALSE;
            BreadthFirstSearchShortestPath(&TempNetwork, splitNode, Dest, &tempPath);
            TempNetwork.Links[linkID].Status = TRUE; /* put back the link */
            if (tempPath.PathDistance < (minDist-SMALL)) {
                EquivalenceP->ShortestPath = tempPath;
                minDist = tempPath.PathDistance;
                }
            }
        }

    if (EquivalenceP->ShortestPath.PathDistance < INFINITY-SMALL)
        EquivalenceP->ShortestPath.PathDistance += distToAdd;
}

void UpdateEquivalences (NetworkTP NetworkP, EquivalenceClassTP EquivalencesP,
    USHORT* NumEquivalences, USHORT BestPath, USHORT Dest)
{
    USHORT linkID, splitNode, h, h2, numHops;
```

```
if (EquivalencesP[BestPath].EquivalenceType == NodeType) {
    if (*NumEquivalences >= MaxEquivalences) {
        printf("Need to increase number of equivalence classes\n");
        exit(-2);
        }

    linkID = EquivalencesP[BestPath].ShortestPath.PathHops[0];
    EquivalencesP[BestPath].FirstLink[EquivalencesP[BestPath].NumFirstLinks] = linkID;
    EquivalencesP[BestPath].NumFirstLinks++;

    EquivalencesP[*NumEquivalences].EquivalenceType = LinkType;
    EquivalencesP[*NumEquivalences].FirstLink[0] = linkID;
    EquivalencesP[*NumEquivalences].NumFirstLinks = 1;
    EquivalencesP[*NumEquivalences].PrefixPath = EquivalencesP[BestPath].PrefixPath;
    EquivalencesP[*NumEquivalences].SuffixPath = EquivalencesP[BestPath].ShortestPath;
    EquivalencesP[*NumEquivalences].SuffixPath.PathDistance = 0.0;
    EquivalencesP[*NumEquivalences].SplitNode = EquivalencesP[BestPath].SplitNode;
    FindBestPathInEquivalence(NetworkP, &EquivalencesP[*NumEquivalences], Dest);
    (*NumEquivalences)++;

    /* can't set new shortest of BestPath until done with previous shortest above */
    FindBestPathInEquivalence(NetworkP, &EquivalencesP[BestPath], Dest);
    }
else { /* link type */
    if (*NumEquivalences >= (MaxEquivalences-2)) { /* will add three more */
        printf("Need to increase number of equivalence classes\n");
        exit(-2);
        }

    /* find where shortest path diverges from suffix */
    /* suffix starts at branch node; shortest starts at 2nd node */
    for (h=1; h<EquivalencesP[BestPath].SuffixPath.NumPathHops; h++) {
        if (EquivalencesP[BestPath].ShortestPath.PathHops[h-1] !=
            EquivalencesP[BestPath].SuffixPath.PathHops[h])
                break;
        }

    /* add one for new split node;  position h is where the paths diverge */
    EquivalencesP[*NumEquivalences].EquivalenceType = NodeType;
    EquivalencesP[*NumEquivalences].FirstLink[0] =
        EquivalencesP[BestPath].ShortestPath.PathHops[h-1];
    EquivalencesP[*NumEquivalences].FirstLink[1] =
        EquivalencesP[BestPath].SuffixPath.PathHops[h];
    EquivalencesP[*NumEquivalences].NumFirstLinks = 2;
```

```
EquivalencesP[*NumEquivalences].PrefixPath = EquivalencesP[BestPath].PrefixPath;
for (h2=0; h2<h; h2++) {  /* add in 1st link plus the part in common */
    linkID = EquivalencesP[BestPath].SuffixPath.PathHops[h2];
    EquivalencesP[*NumEquivalences].PrefixPath.
        PathHops[Equivalences[BestPath].PrefixPath.NumPathHops+h2] = linkID;
    EquivalencesP[*NumEquivalences].PrefixPath.PathDistance +=
        NetworkP->Links[linkID].Length;
    }
EquivalencesP[*NumEquivalences].PrefixPath.NumPathHops += h;
EquivalencesP[*NumEquivalences].SuffixPath.NumPathHops = 0;
EquivalencesP[*NumEquivalences].SuffixPath.PathDistance = 0.0;
linkID = EquivalencesP[BestPath].SuffixPath.PathHops[h];
splitNode = NetworkP->Links[linkID].LinkNode1;
EquivalencesP[*NumEquivalences].SplitNode = splitNode;
FindBestPathInEquivalence(NetworkP, &EquivalencesP[*NumEquivalences], Dest);
(*NumEquivalences)++;

/* add one for one split path */
EquivalencesP[*NumEquivalences].EquivalenceType = LinkType;
EquivalencesP[*NumEquivalences].FirstLink[0] =
    Equivalences[BestPath].ShortestPath.PathHops[h-1];
EquivalencesP[*NumEquivalences].NumFirstLinks = 1;
EquivalencesP[*NumEquivalences].PrefixPath =
    Equivalences[(*NumEquivalences)-1].PrefixPath;
EquivalencesP[*NumEquivalences].SuffixPath.PathDistance = 0.0;
numHops = 0;
for (h2=h-1; h2<Equivalences[BestPath].ShortestPath.NumPathHops; h2++) {
    linkID = Equivalences[BestPath].ShortestPath.PathHops[h2];
    Equivalences[*NumEquivalences].SuffixPath.PathHops[numHops] = linkID;
    numHops++;
    }
EquivalencesP[*NumEquivalences].SuffixPath.NumPathHops = numHops;
EquivalencesP[*NumEquivalences].SplitNode = splitNode;
FindBestPathInEquivalence(NetworkP, &EquivalencesP[*NumEquivalences], Dest);
(*NumEquivalences)++;

/* add one for other split path */
EquivalencesP[*NumEquivalences].EquivalenceType = LinkType;
EquivalencesP[*NumEquivalences].FirstLink[0] =
    Equivalences[BestPath].SuffixPath.PathHops[h];
EquivalencesP[*NumEquivalences].NumFirstLinks = 1;
EquivalencesP[*NumEquivalences].PrefixPath =
    Equivalences[(*NumEquivalences)-1].PrefixPath;
EquivalencesP[*NumEquivalences].SuffixPath.PathDistance = 0.0;
numHops = 0;
```

```
        for (h2=h; h2<EquivalencesP[BestPath].SuffixPath.NumPathHops; h2++) {
            linkID = EquivalencesP[BestPath].SuffixPath.PathHops[h2];
            EquivalencesP[*NumEquivalences].SuffixPath.PathHops[numHops] = linkID;
            numHops++;
            }
        EquivalencesP[*NumEquivalences].SuffixPath.NumPathHops = numHops;
        EquivalencesP[*NumEquivalences].SplitNode = splitNode;
        FindBestPathInEquivalence(NetworkP, &EquivalencesP[*NumEquivalences], Dest);
        (*NumEquivalences)++;

        EquivalencesP[BestPath].SuffixPath.NumPathHops = h;
        FindBestPathInEquivalence(NetworkP, &EquivalencesP[BestPath], Dest);
        }
}

/*********************************************************************/
/*      Code for N Shortest Diverse Paths                            */
/*      Follows method of [Bhan99]                                   */
/*      The N paths are returned in NPaths                           */
/*      The function returns number of paths found                   */
/*      The paths are not necessarily in order from shortest to longest */
/*********************************************************************/

void KDualPathGraphTranformation (NetworkTP NetworkP, PathTP PathP, USHORT Source,
    USHORT Destination, BOOL LinkDisjointOnly, BOOL CommonLinksAllowed,
    BOOL CommonNodesAllowed, USHORT DummyNodeThreshold);
void GenerateTwoRealPaths (NetworkTP NetworkP, PathTP TempPath1P, PathTP TempPath2P,
    PathTP RealPath1P, PathTP RealPath2P);
void CleanPath (NetworkTP NetworkP, USHORT DummyLinkThreshold, USHORT Source,
    USHORT Destination, PathTP NewPathP);
void AdjustNodeInfoForNewLink (NetworkTP NetworkP, USHORT LinkID);
void ChangeLinkSource (NetworkTP NetworkP, USHORT LinkID, USHORT OldSource,
    USHORT NewSource);
void ChangeLinkDestination (NetworkTP NetworkP, USHORT LinkID, USHORT OldDest,
    USHORT NewDest);
USHORT GetReverseLink (NetworkTP NetworkP, USHORT LinkID);

USHORT NShortestDiversePaths (NetworkTP NetworkP, USHORT Source, USHORT Destination,
    USHORT NumDisjointPaths, BOOL LinkDisjointOnly, BOOL CommonLinksAllowed,
    BOOL CommonNodesAllowed, PathTP NPaths)
    /* looks for N paths that are mutually disjoint, or maximally disjoint depending on the settings */
    /* if LinkDisjointOnly is TRUE, then it doesn't try to avoid common nodes among the paths */
    /* if CommonLinksAllowed is TRUE, then if fully disjoint paths can't be found, it will accept */
    /*    paths with common links (although it still minimizes the number of common links) */
    /* if CommonNodesAllowed is TRUE, then if fully disjoint paths can't be found, it will accept */
    /*    paths with common nodes (althouth it still minimizes the number of common nodes) */
```

```
{
    USHORT h, n, numFoundPaths;
    int j, k;
    PathTP tempPaths;

    for (n=0; n<NumDisjointPaths; n++) {
        NPaths[n].NumPathHops = 0;
        NPaths[n].PathDistance = 0.0;
        }

    if (Source == Destination)
        return(NumDisjointPaths);

    /* first find shortest path */
    if (!BreadthFirstSearchShortestPath(NetworkP, Source, Destination, &NPaths[0]))
        return (0);  /* didn't find any paths */
    if (NumDisjointPaths == 1)
        return(1);

    tempPaths = (PathTP)malloc(NumDisjointPaths*sizeof(PathT));
    if (tempPaths == NULL) {
        printf("Out of Memory\n");
        exit(-20);
        }

    tempPaths[0] = NPaths[0];
    for (n=1; n<NumDisjointPaths; n++) {
        /* make a copy of the network because will perform graph transformations */
        TempNetwork = *NetworkP;
        for (j=0; j<n; j++)
            KDualPathGraphTranformation(&TempNetwork, &NPaths[j], Source, Destination,
                LinkDisjointOnly, CommonLinksAllowed, CommonNodesAllowed,
                NetworkP->NumNodes);

        /* run shortest path on the transformed graph */
        if (!BreadthFirstSearchShortestPath(&TempNetwork, Source, Destination, &tempPaths[n]))
            break;

        /* clean path up; may have dummy nodes in it */
        CleanPath(&TempNetwork, NetworkP->NumLinks, Source, Destination, &tempPaths[n]);

        /* paths found above may not be the true paths; need to unravel any interleaving */
        for (j=n; j>0; j--) {
            for (k=j-1; k>=0; k--) {
```

```
                    GenerateTwoRealPaths(&TempNetwork, &tempPaths[j],
                        &tempPaths[k], &NPaths[j], &NPaths[k]);
                    tempPaths[j] = NPaths[j];
                    tempPaths[k] = NPaths[k];
                    }
                }
            }

        numFoundPaths = n;
        for (n=0; n<numFoundPaths; n++) {
            NPaths[n].PathDistance = 0.0;
            for (h=0; h<NPaths[n].NumPathHops; h++)
                NPaths[n].PathDistance += NetworkP->Links[NPaths[n].PathHops[h]].Length;
            }
        free(tempPaths);

        return(numFoundPaths);
}

void KDualPathGraphTranformation (NetworkTP NetworkP, PathTP PathP, USHORT Source,
        USHORT Destination, BOOL LinkDisjointOnly, BOOL CommonLinksAllowed,
        BOOL CommonNodesAllowed, USHORT DummyNodeThreshold)
{
        USHORT n, prevNode, link, newLink, dummyID, forwardLink, reverseLink, reverseLink2;
        int h;
        double tempLength;

        for (h=PathP->NumPathHops-1; h>=0; h--) {
            forwardLink = PathP->PathHops[h];
            if (h == 0) { /* don't split source node, but handle the link */
                if (NetworkP->Links[forwardLink].Length > (CommonLinkPenalty - SMALL)) {
                    /* must be a common link in the previous paths that have been found */
                    /* increase penalty if use it again */
                    NetworkP->Links[forwardLink].Length += CommonLinkPenalty;
                    continue;
                    }

                reverseLink = GetReverseLink(NetworkP, forwardLink);
                tempLength = NetworkP->Links[forwardLink].Length;

                NetworkP->Links[reverseLink].Length = -1.0*tempLength;
                NetworkP->Links[reverseLink].Status = TRUE;

                NetworkP->Links[forwardLink].Status = FALSE;
                /* add big amount to length to discourage its use */
```

```
        NetworkP->Links[forwardLink].Length = tempLength + CommonLinkPenalty;
        if (CommonLinksAllowed) { /* common links OK if can't be avoided */
            NetworkP->Links[forwardLink].Status = TRUE;
            }
        continue;
        }

    prevNode = NetworkP->Links[forwardLink].LinkNode1;

    if (prevNode >= DummyNodeThreshold) {
        /* must be a common node in the previous paths that have been found */
        /* don't add another dummy node */

        if (NetworkP->Links[forwardLink].Length > (CommonLinkPenalty - SMALL)) {
            /* must also be a common link in the previous paths that have been found */
            /* increase penalty if use it again */
            NetworkP->Links[forwardLink].Length += CommonLinkPenalty;
            continue;
            }

        reverseLink = GetReverseLink(NetworkP, forwardLink);
        tempLength = NetworkP->Links[forwardLink].Length;

        NetworkP->Links[reverseLink].Length = -1.0*tempLength;
        NetworkP->Links[reverseLink].Status = TRUE;

        NetworkP->Links[forwardLink].Status = FALSE;
        /* add big amount to length to discourage its use */
        NetworkP->Links[forwardLink].Length = tempLength + CommonLinkPenalty;
        if (CommonLinksAllowed) { /* common links OK if can't be avoided */
            NetworkP->Links[forwardLink].Status = TRUE;
            }

        if ((!LinkDisjointOnly) && (CommonNodesAllowed)) {
            /* increase penalty if use the node again */
            for (n=0; n<NetworkP->Nodes[prevNode].NumIncomingLinks; n++) {
                link = NetworkP->Nodes[prevNode].IncomingLinks[n];
                if (NetworkP->Links[link].Length > (CommonNodePenalty-SMALL)) {
                    NetworkP->Links[link].Length += CommonNodePenalty;
                    break;
                    }
                }
            }

        continue;
```

```
    }

/* split prevNode */
if (NetworkP->NumNodes >= MaxNodesWithDummies) {
    printf("Need to increase MaxNodes\n");
    exit(-4);
    }
if (NetworkP->NumLinks >= (MaxLinksWithDummies-1)) { /* will add 2 links */
    printf("Need to increase MaxLinks\n");
    exit(-5);
    }

dummyID = NetworkP->NumNodes;
NetworkP->NumNodes++;
strcpy(NetworkP->Nodes[dummyID].Name, "DummyAdd");
NetworkP->Nodes[dummyID].NumOutgoingLinks = 0;
NetworkP->Nodes[dummyID].NumIncomingLinks = 0;

reverseLink = GetReverseLink(NetworkP, forwardLink);
tempLength = NetworkP->Links[forwardLink].Length;

NetworkP->Links[reverseLink].Length = -1.0*tempLength;
NetworkP->Links[reverseLink].Status = TRUE;
ChangeLinkDestination(NetworkP, reverseLink, prevNode, dummyID);

NetworkP->Links[forwardLink].Status = FALSE;
/* add big amount to length to discourage its use */
NetworkP->Links[forwardLink].Length = tempLength + CommonLinkPenalty;
if (CommonLinksAllowed) { /* common links OK if can't be avoided */
    NetworkP->Links[forwardLink].Status = TRUE;
    ChangeLinkSource(NetworkP, forwardLink, prevNode, dummyID);
    }

reverseLink = GetReverseLink(NetworkP, PathP->PathHops[h-1]);
for (n=0; n<NetworkP->Nodes[prevNode].NumOutgoingLinks; n++) {
    link = NetworkP->Nodes[prevNode].OutgoingLinks[n];
    if (link == reverseLink) {
        continue;
        }
    else {  /* for all other links emanating from prevNode, change the source to dummy */
        ChangeLinkSource(NetworkP, link, prevNode, dummyID);
        n--;
        }
    }
```

```
        /* add dummy link to point from dummy to prevNode - give it distance 0 */
        newLink = AddLinkToTopology(NetworkP, dummyID, prevNode, 0.0, FALSE);

        if (LinkDisjointOnly) {
            reverseLink2 = GetReverseLink(NetworkP, newLink); /* prevNode to dummy */
            NetworkP->Links[reverseLink2].Status = TRUE;
            NetworkP->Links[reverseLink2].Length = SMALL;
            }
        else if (CommonNodesAllowed) { /* common nodes OK if can't be avoided */
            reverseLink2 = GetReverseLink(NetworkP, newLink);/* prevNode to dummy */
            NetworkP->Links[reverseLink2].Status = TRUE;
            /* add big amount to length to discourage its use */
            NetworkP->Links[reverseLink2].Length = CommonNodePenalty;
            }
        }
}

void GenerateTwoRealPaths (NetworkTP NetworkP, PathTP TempPath1P, PathTP TempPath2P,
    PathTP RealPath1P, PathTP RealPath2P)
{
    int i, j, k, lastpos, lastj;
    USHORT link;
    BOOL exchange;

    /* to generate the shortest paths, look for overlapping sections in the two paths */
    RealPath1P->NumPathHops = RealPath2P->NumPathHops = 0;
    exchange = FALSE;
    lastpos = 0;
    for (i=0; i<TempPath1P->NumPathHops; i++) {
        for (j=0; j<TempPath2P->NumPathHops; j++) {
            if (GetReverseLink(NetworkP, TempPath2P->PathHops[j]) ==
             TempPath1P->PathHops[i]) {
                /* found an overlapping section; remove it and then exchange links */
                lastj = j;
                for (i++,j--; i<TempPath1P->NumPathHops, j>=0; i++, j--) {
                    if (GetReverseLink(NetworkP, TempPath2P->PathHops[j]) !=
                        TempPath1P->PathHops[i])
                            break;
                    }
                j++; i--; /* go back to last position of overlap */
                for (k=lastpos; k<j; k++) {
                    link = TempPath2P->PathHops[k];
                    if (!exchange) {
                        RealPath2P->PathHops[RealPath2P->NumPathHops] = link;
                        RealPath2P->NumPathHops++;
```

```
                    }
                else {
                    RealPath1P->PathHops[RealPath1P->NumPathHops] = link;
                    RealPath1P->NumPathHops++;
                    }
                }
            exchange = !exchange;
            lastpos = lastj + 1;
            break;
            }
        }
    if (j == TempPath2P->NumPathHops) { /* not an overlapping link */
        if (!exchange) {
            RealPath1P->PathHops[RealPath1P->NumPathHops] = TempPath1P->PathHops[i];
            RealPath1P->NumPathHops++;
            }
        else {
            RealPath2P->PathHops[RealPath2P->NumPathHops] = TempPath1P->PathHops[i];
            RealPath2P->NumPathHops++;
            }
        }
    }
for (k=lastpos; k<TempPath2P->NumPathHops; k++) {
    link = TempPath2P->PathHops[k];
    if (!exchange) {
        RealPath2P->PathHops[RealPath2P->NumPathHops] = link;
        RealPath2P->NumPathHops++;
        }
    else {
        RealPath1P->PathHops[RealPath1P->NumPathHops] = link;
        RealPath1P->NumPathHops++;
        }
    }
}

void CleanPath (NetworkTP NetworkP, USHORT DummyLinkThreshold, USHORT Source,
    USHORT Destination, PathTP NewPathP)
{
    USHORT h, linkID, numHops;

    numHops = 0;
    for (h=0; h<NewPathP->NumPathHops; h++) {
        linkID = NewPathP->PathHops[h];
        if (linkID >= DummyLinkThreshold)
            continue;
```

```
            NewPathP->PathHops[numHops] = linkID;
            numHops++;
            }
        NewPathP->NumPathHops = numHops;
}

USHORT AddLinkToTopology (NetworkTP NetworkP, USHORT Node1, USHORT Node2,
        double Length, BOOL ReverseLinkStatus)
{
        USHORT newLinkID, reverseLinkID;

        /* assumes links always added in pairs */
        /* if reverse link direction not needed, pass in its status as FALSE */

        if (NetworkP->NumLinks >= (MaxLinksWithDummies-1)) {
            printf("Need to increase MaxLinks\n");
            exit(-5);
            }

        newLinkID = NetworkP->NumLinks;
        NetworkP->NumLinks += 2;

        NetworkP->Links[newLinkID].LinkNode1 = Node1;
        NetworkP->Links[newLinkID].LinkNode2 = Node2;
        NetworkP->Links[newLinkID].Length = Length;
        NetworkP->Links[newLinkID].Status = TRUE;
        AdjustNodeInfoForNewLink(NetworkP, newLinkID);

        reverseLinkID = GetReverseLink(NetworkP, newLinkID);
        NetworkP->Links[reverseLinkID].LinkNode1 = Node2;
        NetworkP->Links[reverseLinkID].LinkNode2 = Node1;
        NetworkP->Links[reverseLinkID].Length = Length;
        NetworkP->Links[reverseLinkID].Status = ReverseLinkStatus;
        AdjustNodeInfoForNewLink(NetworkP, reverseLinkID);

        return(newLinkID);
}

void AdjustNodeInfoForNewLink (NetworkTP NetworkP, USHORT LinkID)
{
        USHORT node;

        node = NetworkP->Links[LinkID].LinkNode1;
        if (NetworkP->Nodes[node].NumOutgoingLinks >= MaxNodeDegree) {
            printf("Need to increase MaxNodeDegree\n");
            exit(-6);
            }
```

```
    NetworkP->Nodes[node].OutgoingLinks[NetworkP->Nodes[node].NumOutgoingLinks] =
        LinkID;
    NetworkP->Nodes[node].NumOutgoingLinks++;

    node = NetworkP->Links[LinkID].LinkNode2;
    if (NetworkP->Nodes[node].NumIncomingLinks >= MaxNodeDegree) {
        printf("Need to increase MaxNodeDegree\n");
        exit(-6);
        }
    NetworkP->Nodes[node].IncomingLinks[NetworkP->Nodes[node].NumIncomingLinks] =
        LinkID;
    NetworkP->Nodes[node].NumIncomingLinks++;
}

void ChangeLinkSource (NetworkTP NetworkP, USHORT LinkID, USHORT OldSource,
    USHORT NewSource)
{
    USHORT n;

    if (NetworkP->Links[LinkID].LinkNode1 != OldSource) {
        printf("Inconsistent\n");
        exit(-7);
        }
    if (NetworkP->Nodes[NewSource].NumOutgoingLinks >= MaxNodeDegree) {
        printf("Need to increase MaxNodeDegree\n");
        exit(-6);
        }

    NetworkP->Links[LinkID].LinkNode1 = NewSource;
    NetworkP->Nodes[NewSource].OutgoingLinks[NetworkP->Nodes[NewSource].
        NumOutgoingLinks] = LinkID;
    NetworkP->Nodes[NewSource].NumOutgoingLinks++;

    /* remove it from OldSource list*/
    for (n=0; n<NetworkP->Nodes[OldSource].NumOutgoingLinks; n++)
        if (NetworkP->Nodes[OldSource].OutgoingLinks[n] == LinkID) break;
    for (; n<NetworkP->Nodes[OldSource].NumOutgoingLinks-1; n++) {
        NetworkP->Nodes[OldSource].OutgoingLinks[n] =
            NetworkP->Nodes[OldSource].OutgoingLinks[n+1];
        }
    NetworkP->Nodes[OldSource].NumOutgoingLinks--;
}
```

```
void ChangeLinkDestination (NetworkTP NetworkP, USHORT LinkID, USHORT OldDest,
    USHORT NewDest)
{
    USHORT n;

    if (NetworkP->Links[LinkID].LinkNode2 != OldDest) {
        printf("Inconsistent\n");
        exit(-8);
        }
    if (NetworkP->Nodes[NewDest].NumIncomingLinks >= MaxNodeDegree) {
        printf("Need to increase MaxNodeDegree\n");
        exit(-6);
        }

    NetworkP->Links[LinkID].LinkNode2 = NewDest;
    NetworkP->Nodes[NewDest].IncomingLinks[NetworkP->Nodes[NewDest].
        NumIncomingLinks] = LinkID;
    NetworkP->Nodes[NewDest].NumIncomingLinks++;

    /* remove it from OldDest list*/
    for (n=0; n<NetworkP->Nodes[OldDest].NumIncomingLinks; n++)
        if (NetworkP->Nodes[OldDest].IncomingLinks[n] == LinkID) break;

    for (; n<NetworkP->Nodes[OldDest].NumIncomingLinks-1; n++) {
        NetworkP->Nodes[OldDest].IncomingLinks[n] =
            NetworkP->Nodes[OldDest].IncomingLinks[n+1];
        }
    NetworkP->Nodes[OldDest].NumIncomingLinks--;
}

USHORT GetReverseLink (NetworkTP NetworkP, USHORT LinkID)
{
    /* Assumes links are always added in pairs */
    if (LinkID % 2 == 0)
        return(LinkID + 1);
    else return(LinkID - 1);
}
```

Abbreviations

ADM - Add/Drop Multiplexer
ANSI - American National Standards Institute
ASE - Amplified Spontaneous Emission
ASON - Automatically Switched Optical Network
ATM - Asynchronous Transfer Mode
AWG - Arrayed Waveguide Grating
BFS – Breadth First Search
BLSR - Bi-directional Line-Switched Ring
CapEx – Capital Expenditure
CDS – Connected Dominating Set
CSP - Constrained Shortest Path
dB – Decibel
DCE - Dynamic Channel Equalizer
DCF - Dispersion Compensating Fiber
DSE - Dynamic Spectral Equalizer
EDC - Electronic Dispersion Compensation
EDFA - Erbium Doped Fiber Amplifier
E-NNI - External Network-Network Interface
FEC - Forward Error Correction
FWM - Four-Wave Mixing
Gb/s – Gigabit per second (10^9 bits per second)
GC – Grooming Connection
GHz – Gigahertz
GMPLS - Generalized Multi-Protocol Label Switching
IETF - Internet Engineering Task Force
I-NNI – Internal Network-Network Interface
IP - Internet Protocol
ITU - International Telecommunication Union
JET - Just Enough Time
JIT - Just In Time
km – kilometer
MAC - Media Access Control

Mb/s – Megabit per second (10^6 bits per second)
MCP - Multi-Constrained Path
MEMS – Micro-electro-mechanical System
MLSE - Maximum Likelihood Sequence Estimation
ms - millisecond
NDSF - Non Dispersion-Shifted Fiber
NF - Noise Figure
nm – nanometer
NNI – Network-Network Interface
NRZ - Non-Return-to-Zero
NZ-DSF - Non-Zero Dispersion-Shifted Fiber
OADM - Optical Add/Drop Multiplexer
OADM-MD - Multi-Degree Optical Add/Drop Multiplexer
OAM - Operations, Administration, and Maintenance
OBS – Optical Burst Switching
OC - Optical Carrier
OChSPRing - Optical-Channel Shared Protection Ring
O-E-O – Optical-Electrical-Optical
OFS - Optical Flow Switching
OIF - Optical Internetworking Forum
OMS-SPRing - Optical Multiplex Section Shared Protection Ring
OOK - On-Off Keying
OpEx – Operational Expenditure
OPM - Optical Performance Monitor
OPS – Optical Packet Switching
OSC - Optical Supervisory Channel
OSNR - Optical-Signal-to-Noise-Ratio
OTN - Optical Transport Network
OTU - Optical channel Transport Unit
OXC – Optical Cross-Connect
PCE - Path Computation Element
PDL - Polarization Dependent Loss
PIC - Photonic Integrated Circuit
PMD - Polarization-Mode Dispersion
PON - Passive Optical Network
ps - picosecond (10^{-12} second)
RFC - Request for Comments
RFI – Request for Information
RFP – Request for Proposal
ROADM – Reconfigurable Optical Add/Drop Multiplexer
RSP - Restricted Shortest Path
RWA - Routing and Wavelength Assignment
RZ - Return-to-Zero
SDH - Synchronous Digital Hierarchy
SLA - Service Level Agreement

SONET - Synchronous Optical Network
SPDP - Shortest Pair of Disjoint Paths
SPM - Self-Phase Modulation
SPRing – Shared Protection Ring
SRG - Shared Risk Group
SRLB - Selective Randomized Load Balancing
SRLG - Shared Risk Link Group
STM - Synchronous Transport Module
STS - Synchronous Transport Signal
Tb/s – Terabit per second (10^{12} bits per second)
THz – Terahertz
TWIN - Time-Domain Wavelength Interleaved Networking
TxRx – Transmitter/Receiver Card (Transponder)
UNI - User-Network Interface
VCAT - Virtual Concatenation
WDM - Wavelength Division Multiplexing
WGR – Wavelength Grating Router
WSS - Wavelength-Selective Switch
WSXC - Wavelength-Selective Cross-connect
XPM - Cross-phase Modulation

Bibliography

[ADHN01] G. P. Austin, B. T. Doshi, C. J. Hunt, R. Nagarajan, and M. A. Qureshi, "Fast, scalable, and distributed restoration in general mesh optical networks," *Bell Labs Technical Journal*, pp. 67-81, Jan.-Jun. 2001.

[ABGL01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," Internet Engineering Task Force, Request for Comments (RFC) 3209, Dec. 2001.

[BaGe06] R. Batchellor and O. Gerstel, "Cost effective architectures for core transport networks," *Proceedings, Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC'06)*, Anaheim, CA, Mar. 5-10, 2006, Paper PDP42.

[BaHu96] R. A. Barry and P. A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 5, pp. 858-867, Jun. 1996.

[BaKi02] P. Bayvel and R. Killey, "Nonlinear optical effects in WDM transmission," in *Optical Fiber Telecommunications IV B*, I. Kaminow and T. Li, Editors, San Diego: Academic Press, 2002, pp. 611-641.

[BaLe02] N. Barakat and A. Leon-Garcia, "An analytic model for predicting the locations and frequencies of 3R regenerations in all-optical wavelength-routed WDM networks," *Proceedings, IEEE International Conference on Communications (ICC'02)*, New York, NY, Apr. 28-May 2, 2002, vol. 5, pp. 2812-2816.

[BCRV06] G. Bernstein, D. Caviglia, R. Rabbat, and H. Van Helvoort, "VCAT-LCAS in a clamshell," *IEEE Communications Magazine*, vol. 44, no. 5, pp. 34-36, May 2006.

[Berg03] L. Berger, Editor, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description," Internet Engineering Task Force, Request for Comments (RFC) 3471, Jan. 2003.

[BeRS03] G. Bernstein, B. Rajagopalan, and D. Saha, *Optical Network Control: Architecture, Protocols, and Standards*, Reading, MA: Addison-Wesley Professional, 2003.

[Bhan99] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Boston, MA: Kluwer Academic Publishers, 1999.

[BhSF01] N. M. Bhide, K. M. Sivalingam, and T. Fabry-Asztalos, "Routing mechanisms employing adaptive weight functions for shortest path routing in optical WDM networks," *Photonic Network Communications*, vol. 3, no. 3, pp. 227-236, Jul. 2001.

[BLRC02] E. Bouillet, J.-F. Labourdette, R. Ramamurthy, and S. Chaudhuri, "Enhanced algorithm cost model to control tradeoffs in provisioning shared mesh restored lightpaths," *Proceedings, Optical Fiber Communication (OFC'02)*, Anaheim, CA, Mar. 17-22, 2002, Paper ThW2.

[Blum04] D. J. Blumenthal, "Optical packet switching," *Proceedings, 17th Annual Meeting of the IEEE LEOS*, Puerto Rico, Nov. 7-11, 2004, Paper ThU1.

[BoUh98] A. Boroujerdi and J. Uhlmann, "An efficient algorithm for computing least cost paths with turn constraints," *Information Processing Letters*, vol. 67, pp. 317-321, 1998.

[BSAL02] A. Boskovic, M. Sharma, N. Antoniades, and M. Lee, "Broadcast and select OADM nodes application and performance trade-offs," *Proceedings, Optical Fiber Communication (OFC'02),* Anaheim, CA, Mar. 17-22, 2002, Paper TuX2.

[BuWW03] P. Bullock, C. Ward, and Q. Wang, "Optimizing wavelength grouping granularity for optical add-drop network architectures," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper WH2.

[CaAQ04] X. Cao, V. Anand, and C. Qiao, "Multi-layer versus single-layer optical cross-connect architectures for waveband switching," *Proceedings, IEEE INFOCOM 2004*, Hong Kong, Mar. 7-11, 2004, vol. 3, pp. 1830-1840.

[CaCH04] H. S. Carrer, D. E. Crivelli, and M. R. Hueda, "Maximum likelihood sequence estimation receivers for DWDM lightwave systems," *Proceedings, IEEE Global Telecommunications Conference* (*GLOBECOM'04)*, Dallas, TX, Nov. 29-Dec. 3, 2004, vol. 2, pp. 1005–1010.

[ChAN03] C. Chigan, G. W. Atkinson, and R. Nagarajan, "Cost effectiveness of joint multilayer protection in packet-over-optical networks," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2694-2704, Nov. 2003.

[ChCF04] T. Y. Chow, F. Chudak, and A. M. Ffrench "Fast optical layer mesh protection using pre-cross-connected trails," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 539-548, Jun. 2004.

[ChGn06] S. Chandrasekhar and A. H. Gnauck, "Performance of MLSE receiver in a dispersion-managed multispan experiment at 10.7 Gb/s under nonlinear transmission," *IEEE Photonics Technology Letters*, vol. 18, no. 23, pp. 2448–2450, Dec. 1, 2006.

[ChLH06] A. L. Chiu, G. Li, and D.-M. Hwang, "New problems on wavelength assignment in ULH networks," *Proceedings, Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC'06)*, Anaheim, CA, Mar. 5-10, 2006, Paper NThH2.

[ChQY04] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: A new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16-23, May/Jun. 2004.

[ChSc07] M. W. Chbat and H.-J. Schmidtke, "Falling boundaries from metro to ULH optical transport equipment," *Proceedings, Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC'07)*, Anaheim, CA, Mar. 25-29, 2007, Paper NTuA3.

[ChWM06] V. W. S. Chan, G. Weichenberg, and M. Médard, "Optical flow switching," *Workshop on Optical Burst Switching (WOBS)*, San Jose, CA, Oct. 2006.

[ChYu94] K. Chan and T. P. Yum, "Analysis of least congested path routing in WDM lightwave networks," *Proceedings, IEEE INFOCOM 1994*, Toronto, Ontario, Jun. 12-16, 1994, vol. 2, pp. 962-969.

[ChYu03] A. Chiu and C. Yu, "Economic benefits of transparent OXC networks as compared to long systems with OADMs," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper WQ2.

[CMSG04] T. J. Carpenter, R. C. Menendez, D. F. Shallcross, J. W. Gannett, J. Jackel, and A. C. Von Lehmen, "Cost-conscious impairment-aware routing," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper MF88.

[CoLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Cambridge, MA: MIT Press, 1990.

[Conr02] J. Conradi, "Bandwidth-efficient modulation formats for digital fiber transmission systems," in *Optical Fiber Telecommunications IV B*, I. Kaminow and T. Li, Editors, San Diego: Academic Press, 2002, pp. 862-901.

[CSGJ03] T. Carpenter, D. Shallcross, J. Gannett, J. Jackel, and A. Von Lehmen, "Maximizing the transparency advantage in optical networks," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper FA2.

[CXBT02] N. Chi, L. Xu, K. S. Berg, T. Tokle, and P. Jeppesen, "All-optical wavelength conversion and multichannel 2R regeneration based on highly nonlinear dispersion-imbalanced loop mirror," *IEEE Photonics Technology Letters*, vol. 14, no. 11, pp. 1581-1583, Nov. 2002.

[DGGM02] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, "Resource management with hoses: Point-to-cloud services for virtual private networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 679-692, Oct. 2002.

[DoOk06] C. R. Doerr and K. Okamoto, "Advances in silica planar lightwave circuits," *Journal of Lightwave Technology*, vol. 24, no. 12, pp. 4763-4789, Dec. 2006.

[DuRo02] R. Dutta and G. N. Rouskas, "Traffic grooming in WDM networks: Past and future," *IEEE Network,* vol. 16, no. 6, pp. 46-56, Nov./Dec. 2002.

[EBRL02] G. Ellinas, E. Bouillet, R. Ramamurthy, J.-F. Labourdette, S. Chaudhuri, and K. Bala, "Restoration in layered architectures with a WDM mesh optical layer," *International Engineering Consortium (IEC) Annual Review of Communications*, vol. 55, Jun. 2002.

[EBRL03] G. Ellinas, E. Bouillet, R. Ramamurthy, J.-F. Labourdette, S. Chaudhuri, and K. Bala, "Routing and restoration architectures in mesh optical networks," *Optical Networks Magazine*, vol. 4, no. 1, pp. 91-106, Jan./Feb. 2003.

[ElBa06] T. S. El-Bawab, Editor, *Optical Switching*, New York, NY: Springer, 2006.

[ElMW06] A. Elwalid, D. Mitra, and Q. Wang, "Distributed nonlinear integer optimization for data-optical internetworking," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1502-1513, Aug. 2006.

[EMSW03] A. Elwalid, D. Mitra, I. Saniee, and I. Widjaja, "Routing and protection in GMPLS networks: From shortest paths to optimized designs," *Journal of Lightwave Technology,* vol. 21, no. 11, pp. 2828-2838, Nov. 2003.

[Epps94] D. Eppstein, "Finding the *k* shortest paths," *Proceedings, 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, Nov. 20-22, 1994, pp. 154-165.

[FaVA06] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," Internet Engineering Task Force, Request for Comments (RFC) 4655, Aug. 2006.

[FoTC97] F. Forghieri, R. W. Tkach, and A. R. Chraplyvy, "Fiber nonlinearities and their impact on transmission systems," in *Optical Fiber Telecommunications III A*, I. Kaminow and T. Koch, Editors, San Diego: Academic Press, 1997, pp. 196-254.

[GaJo79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York, NY: W.H. Freeman and Co., 1979.

[GeRa00] O. Gerstel and R. Ramaswami, "Optical layer survivability – an implementation perspective," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1885-1899, Oct. 2000.

[GeRa04] O. Gerstel and H. Raza, "Predeployment of resources in agile photonic networks," *Journal of Lightwave Technology*, vol. 22, no. 10, pp. 2236-2244, Oct. 2004.

[GeRS98] O. Gerstel, R. Ramaswami, and G. Sasaki, "Cost effective traffic grooming in WDM rings," *Proceedings, IEEE INFOCOM 1998,* San Francisco, CA, Mar. 29-Apr. 2, 1998, pp. 69-77.

[GoLY02] M. Goyal, G. Li, and J. Yates, "Shared mesh restoration: a simulation study," *Proceedings, Optical Fiber Communication (OFC'02)*, Anaheim, CA, Mar. 17-22, 2002, Paper ThO2.

[Gora02] W. J. Goralski, *SONET/SDH*, Third Edition, New York, NY: McGraw-Hill, 2002.

[Grov03] W. Grover, *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*, Upper Saddle River, NJ: Prentice-Hall, 2003.

[GrSt98] W. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," *Proceedings, IEEE International Conference on Communications (ICC'98)*, Atlanta, GA, Jun. 7-11, 1998, vol. 1, pp. 537–543.

[GuCh03] A. Gumaste and I. Chlamtac, "Mesh implementation of light-trails: A solution to IP centric communication," *Proceedings, International Conference on Computer Communication and Networks (ICCCN'03),* Dallas, TX, Oct. 20-22, 2003, pp. 178-183.

[GuKh98] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, 1998, pp. 374-387.

[GuOr99] R. A. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Transactions on Networking,* vol. 7, no. 3, pp. 350-364, Jun. 1999.

[GuOr02] R. Guerin and A. Orda, "Computing shortest paths for any number of hops," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 613-620, Oct. 2002.

[GuPM03] K. P. Gummadi, M. J. Pradeep, and C. S. R. Murthy, "An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 81-94, Feb. 2003.

[HeBr06] J. He and M. Brandt-Pearce, "RWA using wavelength ordering for crosstalk limited networks," *Proceedings, Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC'06)*, Anaheim, CA, Mar. 5-10, 2006, Paper OFG4.

[HeMS03] J. Hershberger, M. Maxel, and S. Suri, "Finding the k shortest simple paths: A new algorithm and its implementation," *Proceedings, Fifth Workshop on Algorithm Engineering and Experiments*, Baltimore, MD, Jan. 11, 2003, pp. 26-36.

[HoMo02] P. H. Ho and H. T. Mouftah, "A framework for service-guaranteed shared protection in WDM mesh networks," *IEEE Communications Magazine*, vol. 40, no. 2, pp. 97-103, Feb. 2002.

[HPWY07] N. J. A. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. W. S. Chan "Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs," *Proceedings, IEEE INFOCOM'07*, Anchorage, AK, May 6-12, 2007, pp. 697-705.

[HSKO99] K. Harada, K. Shimizu, T. Kudou, and T. Ozeki, "Hierarchical optical path cross-connect systems for large scale WDM networks," *Proceedings, Optical Fiber Communication (OFC'99)*, San Diego, CA, Feb. 21-26, 1999, Paper WM55.

[HuDu07] S. Huang and R. Dutta, "Dynamic traffic grooming: The changing role of traffic grooming," *IEEE Communications Surveys and Tutorials*, 1st Quarter 2007, vol. 9, no. 1, pp. 32-49.

[HXGB05] Y.-K. Huang, L. Xu, I. Glesk, V. Baby, B. Li, and P. R. Prucnal, "Simultaneous all-optical 3R regeneration of multiple WDM channels," *Proceedings, 18th Annual Meeting of the IEEE LEOS*, Sydney, Australia, Oct. 23-27, 2005, Paper MM2.

[IGKV03] R. Izmailov, S. Ganguly, V. Kleptsyn, and A. C. Varsou, "Non-uniform waveband hierarchy in hybrid optical networks," *Proceedings, IEEE INFOCOM 2003*, San Francisco, CA, Mar. 30-Apr. 3, 2003, vol. 2, pp.1344-1354.

[IrMG98] R. R. Iraschko, M. H. MacGregor, and W. D. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks," *IEEE/ACM Transactions on Networking,* vol. 6, no. 3, pp. 325-336, Jun. 1998.

[ITU01] International Telecommunication Union, Architecture of Optical Transport Networks, ITU-T Rec. G.872, Nov. 2001.

[ITU03] International Telecommunication Union, Interfaces for the Optical Transport Network (OTN), ITU-T Rec. G.709, Mar. 2003.

[ITU06] International Telecommunication Union, Architecture for the Automatically Switched Optical Network (ASON), ITU-T Rec. G.8080/Y.1304, Jun. 2006.

[JoFN04] G. L. Jones, W. Forysiak, and J. H. B. Nijhof, "Economic benefits of all-optical cross connects and multi-haul DWDM systems for European national networks," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper WH2.

[KaAr04] E. Karasan and M. Arisoylu, "Design of translucent optical networks: Partitioning and restoration," *Photonic Network Communications*, vol. 8, no. 2, pp. 209-221, Mar. 2004.

[KaAy98] E. Karasan and E. Ayanoglu, "Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 186-196, Apr. 1998.

[KaKL00] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2566-2579, Dec. 2000.

[KaKL03] K. Kar, M. Kodialam, and T. V. Lakshman, "Routing restorable bandwidth guaranteed connections using maximum 2-route flows," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 772 – 781, Oct. 2003.

[KaKY03] D. Katz, K. Kompella, and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2," Internet Engineering Task Force, Request for Comments (RFC) 3630, Sep. 2003.

[KaSG04] G. S. Kanter, A. K. Samal, and A. Gandhi, "Electronic dispersion compensation for extended reach," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper TuG1.

[KBBE04] D. C. Kilper, R. Bach, D. J. Blumenthal, D. Einstein, T. Landolsi, L. Ostar, M. Preiss, and A. E. Willner, "Optical performance monitoring," *Journal of Lightwave Technology*, vol. 22, no. 1, pp. 294-304, Jan. 2004.

[KiLu03] S. Kim and S. Lumetta, "Evaluation of protection reconfiguration for multiple failures in WDM mesh networks," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper TuI7.

[KKKV04] F. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem, "Performance evaluation of constraint-based path selection algorithms," *IEEE Network*, vol. 18, no. 5, pp. 16-23, Sep./Oct. 2004.

[KNSZ04] P. M. Krummrich, R. E. Neuhauser, H.-J. Schmidtke, H. Zech, and M. Birk, "Compensation of Raman transients in optical networks," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper MF82.

[KoGr05] A. Kodian and W. D. Grover, "Failure-independent path-protecting p-cycles: Efficient and simple fully preconnected optical-path protection," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3241-3259, Oct. 2005.

[KoKr01] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," *Proceedings, IEEE INFOCOM 2001*, Anchorage, AK, Apr. 22-26, 2001, vol. 2, pp. 834–843.

[KoLa00] M. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," *Proceedings, IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 26-30, 2000, vol. 2, pp. 902-911.

[KoMB81] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141-145, Jun. 1981.

[KTMT05] P. Kulkarni, A. Tzanakaki, C. M. Machuka, and I. Tomkos, "Benefits of Q-factor based routing in WDM metro networks," *Proceedings, European Conference on Optical Communication (ECOC'05)*, Glasgow, Scotland, Sep. 25-29, 2005, vol. 4, pp. 981-982.

[Kurt93] C. Kurtzke, "Suppression of fiber nonlinearities by appropriate dispersion management," *IEEE Photonics Technology Letters*, vol. 5, no. 10, pp. 1250-1253, Oct. 1993.

[KYJH06] V. Kaman, S. Yuan, O. Jerphagnon, R. J. Helkey, and J. E. Bowers, "Comparison of wavelength-selective cross-connect architectures for reconfigurable all-optical networks," *Proceedings, International Conference on Photonics in Switching*, Crete, Greece, Oct. 16-18, 2006, pp. 1-3.

[KZJP04] V. Kaman, X. Zheng, O. Jerphagnon, C. Pusarla, R. J. Helkey, and J. E. Bowers, "A cyclic MUX–DMUX photonic cross-connect architecture for transparent waveband optical networks," *IEEE Photonics Technology Letters*, vol. 16, no. 2, pp. 638-640, Feb. 2004.

[LaAa87] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Boston, MA: D. Reidel Publishing Co., 1987.

[LaKe91] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, Second Edition, New York, NY: McGraw-Hill, Inc., 1991.

[LeJC04] J. Leuthold, J. Jaques, and S. Cabot, "All-optical wavelength conversion and regeneration," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper WN1.

[LiCS05] G. Li, A. L. Chiu, and J. Strand, "Failure recovery in all-optical ULH networks," *5th International Workshop on Design of Reliable Communication Networks (DRCN'05)*, Island of Ischia, Italy, Oct. 16-19, 2005.

[Lin06] C. Lin, Editor, *Broadband Optical Access Networks and Fiber-to-the-Home: Systems Technologies and Deployment Strategies*, West Sussex, England, John Wiley & Sons, 2006.

[LiRa97] C.-S. Li and R. Ramaswami, "Automatic fault detection, isolation, and recovery in transparent all-optical networks," *Journal of Lightwave Technology*, vol. 15, no. 10, pp. 1784-1793, Oct. 1997.

[LiRa01] G. Liu and K. G. Ramakrishnan, "A*Prune: An algorithm for finding K shortest paths subject to multiple constraints," *Proceedings, IEEE INFOCOM 2001*, Anchorage, AK, Apr. 22-26, 2001, vol. 2, pp. 743-749.

[LiVe02] R. Lingampalli and P. Vengalam, "Effect of wavelength and waveband grooming on all-optical networks with single layer photonic switching," *Proceedings, Optical Fiber Communication (OFC'02)*, Anaheim, CA, Mar. 17-22, 2002, Paper ThP4.

[LiWM07] W. Lin, R. S. Wolff, and B. Mumey, "A markov-based reservation algorithm for wavelength assignment in all-optical networks," *Journal of Lightwave Technology*, vol. 25, no. 7, pp. 1676-1683, Jul. 2007.

[LLBB03] O. Leclerc, B. Lavigne, E. Balmefrezol, P. Brindel, L. Pierre, D. Rouvillain, and F. Seguineau, "Optical regeneration at 40 Gb/s and beyond," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2779-2790, Nov. 2003.

[LSTN05] M.-J. Li, M. J. Soulliere, D. J. Tebben, L. Nederlof, M. D. Vaughn, and R. E. Wagner, "Transparent optical protection ring architectures and applications," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3388-3403, Oct. 2005.

[LWYD07] G. Li, D. Wang, J. Yates, R. Doverspike, and C. Kalmanek, "IP over optical cross-connect architectures," *IEEE Communications Magazine*, vol. 45, no. 2, pp. 34-39, Feb. 2007.

[MaLe03] B. Manseur and J. Leung, "Comparative analysis of network reliability and optical reach," *National Fiber Optic Engineers Conference (NFOEC'03),* Orlando, FL, Sep. 7–11, 2003.

[Mann04] E. Mannie, Editor, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," Internet Engineering Task Force, Request for Comments (RFC) 3945, Oct. 04.

[Maro05] D. M. Marom, et al., "Wavelength-selective 1xK switches using free-space optics and MEMS micromirrors: Theory, design, and implementation," *Journal of Lightwave Technology*, vol. 23, no. 4, pp. 1620-1630, Apr. 2005.

[MaTo03] C. M. Machuca and I. Tomkos, "Failure detection for secure optical networks," *Proceedings, International Conference on Transparent Optical Networks (ICTON'03)*, Warsaw, Poland, Jun. 29-Jul. 3, 2003, pp. 70-75.

[Mats05] M. Matsumoto, "Regeneration of RZ-DPSK signals by fiber-based all-optical regenerators," *IEEE Photonics Technology Letters*, vol. 17, no. 5, pp. 1055-1057, May 2005.

[MaTT05] C. Mas, I. Tomkos, and O. K. Tonguz, "Failure location algorithm for transparent optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 8, pp. 1508-1519, Aug. 2005.

[MDLP05] S. Melle, R. Dodd, C. Liou, D. Perkins, M. Sosa, and M. Yin, "Network planning and economic analysis of an innovative new optical transport architecture: The digital optical network," *National Fiber Optic Engineers Conference (NFOEC'05)*, Anaheim, CA, Mar. 6-11, 2005, Paper NTuA1.

[MeCS98] M. Médard, S. R. Chinn, and P. Saengudomlert, "Attack detection in all-optical networks," *Proceedings, Optical Fiber Communication (OFC'98)*, San Jose, CA, Feb. 22-27, 1998, Paper ThD4.

[MFBG99] M. Médard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Transactions on Networking,* vol. 7, no. 5, pp. 641-652, Oct. 1999.

[MMBF97] M. Médard, D. Marquis, R. A. Barry, and S. G. Finn, "Security issues in all-optical networks," *IEEE Network*, vol. 11, no. 3, pp. 42-48, May-Jun. 1997.

[MMMT03] S. Mechels, L. Muller, G. D. Morley, and D. Tillett, "1D MEMS-based wavelength switching subsystem," *IEEE Communications Magazine*, vol. 41, no. 3, pp. 88-94, Mar. 2003.

[MoAz98] A. Mokhtar and M. Azizoglu, "Adaptive wavelength routing in all-optical networks," *IEEE/ACM Transactions on Networking,* vol. 6, no. 2, pp. 197-206, Apr. 1998.

[MoBB04] A. Mokhtar, L. Benmohamed, and M. Bortz, "OXC port dimensioning strategies in optical networks – a nodal perspective," *IEEE Communications Letters*, vol. 8, no. 5, pp. 283-285, May 2004.

[MoLS02] R. Monnard, H. K. Lee, and A. Srivastava, "Suppressing amplifier transients in lightwave systems," *Proceedings, IEEE/LEOS Summer Topicals*, Mont Tremblant, Quebec, Jul. 15-17, 2002, Paper WE3.

[MORC05] J. McNicol, M. O'Sullivan, K. Roberts, A. Comeau, D. McGhan, and L. Strawczynski, "Electrical domain compensation of optical dispersion", *Proceedings, Optical Fiber Communication (OFC'05)*, Anaheim, CA, Mar. 6-11, 2005, Paper OThJ3.

[MSSD03] X. Masip-Bruin, S. Sànchez-López, J. Solé-Pareta, J. Domingo-Pascual, and D. Colle, "Routing and wavelength assignment under inaccurate routing information in networks with sparse and limited wavelength conversion," *Proceedings, IEEE Global Telecommunications Conference* (*Globecom'03)*, San Francisco, CA, Dec. 1-5, 2003, vol. 5, pp. 2575-2579.

[Mukh06] B. Mukherjee, *Optical WDM Networks*, New York, NY: Springer, 2006.

[NoVD01] L. Noirie, M. Vigoureux, and E. Dotaro "Impact of intermediate traffic grouping on the dimensioning of multi-granularity optical networks," *Proceedings, Optical Fiber Communication (OFC'01)*, Anaheim, CA, Mar. 17-22, 2001, Paper TuG3.

[OIF04] Optical Internetworking Forum, "User Network Interface (UNI) 1.0 Signaling Specification, Release 2," Feb. 27, 2004.

[Okam98] K. Okamoto, "Tutorial: Fundamentals, technology and applications of AWG's," *Proceedings, European Conference on Optical Communication (ECOC'98)*, Madrid, Spain, Sep. 20-24, 1998, pp. 35–37.

[OuMu05] C. Ou and B. Mukherjee, *Survivable Optical WDM Networks*, New York, NY: Springer, 2005.

[OzPJ03] T. Ozugur, M.-A. Park, and J. P. Jue, "Label prioritization in GMPLS-centric all-optical networks," *Proceedings, IEEE International Conference on Communications (ICC'03)*, Anchorage, AK, May 11-15, 2003, vol. 2 , pp. 1283-1287.

[OZSZ04] C. Ou, H. Zang, N. K. Singhal, K. Zhu, L. H. Sahasrabuddhe, R. A. MacDonald, and B. Mukherjee, "Subpath protection for scalability and fast recovery in optical WDM mesh networks," *IEEE Journal on Selected Areas in Communications,* vol. 22, no. 9, pp.1859-1875, Nov. 2004.

[OZZS03] C. Ou, K. Zhu, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "Traffic grooming for survivable WDM networks - shared protection," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 9, pp. 1367-1383, Nov. 2003.

[PaPP03] G. I. Papadimitriou, C. Papazoglou, and A. S. Pomportsis, "Optical switching: Switch fabrics, techniques, and architectures," *Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384-405, Feb. 2003.

[PCHN03] A. R. Pratt, B. Charbonnier, P. Harper, D. Nesset, B. K. Nayar, and N. J. Doran, "40 x 10.7 Gbit/s DWDM transmission over a meshed ULH network with dynamically reconfigurable optical crossconnects," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper PD9.

[PDCS06] M. Pickavet, P. Demeester, D. Colle, D. Staessens, B. Puype, L. Depré, and I. Lievens, "Recovery in multilayer optical networks," *Journal of Lightwave Technology*, vol. 24, no. 1, pp. 122-134, Jan. 2006.

[PrAS05] R. G. Prinz, A. Autenrieth, and D. A. Schupke, "Dual failure protection in multilayer networks based on overlay or augmented model," *5th International Workshop on Design of Reliable Communication Networks (DRCN'05)*, Island of Ischia, Italy, Oct. 16-19, 2005.

[QiXu02] C. Qiao and D. Xu, "Distributed partial information management (DPIM) schemes for survivable networks - Part I," *Proceedings, IEEE INFOCOM 2002*, New York, NY, Jun. 23-27, 2002, vol. 1, pp. 302–311.

[QiYo99] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) – A new paradigm for an optical internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69-84, Jan. 1999.

[RaEl05] T. Rahman and G. Ellinas, "Protection of Multicast Sessions in WDM Mesh Optical Networks," *Proceedings, Optical Fiber Communication (OFC'05)*, Anaheim, CA, Mar. 6-11, 2005, Paper OTuK5.

[RaSi01] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd Edition, San Francisco, CA: Morgan Kaufmann Publishers, 2001.

[ReLG06] R. Rejeb, M. S. Leeson, and R. J. Green, "Fault and attack management in all-optical networks," *IEEE Communications Magazine,* vol. 44, no. 11, pp. 79-86, Nov. 2006.

[RLAC03] R. Ramamurthy, J.-F. Labourdette, A. Akyamac, and S. Chaudhuri,, "Limited sharing on protection channels in mesh optical networks," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper TuI3.

[RoSt02] K. Rottwitt and A. Stentz, "Raman amplification in lightwave communication systems," in *Optical Fiber Telecommunications IV A*, I. Kaminow and T. Li, Editors, San Diego: Academic Press, 2002, pp. 213-258.

[Sala02] D. Y. Al-Salameh, "Optical switching in transport networks: Applications, requirements, architectures, technologies and solutions," in *Optical Fiber Telecommunications IV A*, I. Kaminow and T. Li, Editors, San Diego: Academic Press, 2002, pp. 295-373.

[Sale98a] A. A. M. Saleh, "Islands of transparency – an emerging reality in multiwavelength optical networking," *Proceedings, IEEE/LEOS Summer Topical Meeting on Broadband Optical Networks and Technologies*, Monterey, CA, Jul. 20-24, 1998, p. 36.

[Sale98b] A. A. M. Saleh, "Short- and long-term options for broadband access to homes and businesses," *Conference on the Internet: Next Generation and Beyond*, Cambridge, MA, Nov. 1-2, 1998.

[Sale00] A. A. M. Saleh, "Transparent optical networking in backbone networks," *Proceedings, Optical Fiber Communication (OFC'00)*, Baltimore, MD, Mar. 7-10, 2000, Paper ThD7.

[Sale03] A. A. M. Saleh, "Defining all-optical networking and assessing its benefits in metro, regional and backbone networks," *Proceedings, Optical Fiber Communication (OFC'03)*, Atlanta, GA, Mar. 23-28, 2003, Paper WQ1.

[SaMu00] L. H. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network*, vol. 14, no. 1, pp. 90-102, Jan./Feb. 2000.

[SaSi99] A. A. M. Saleh and J. M. Simmons, "Architectural principles of optical regional and metropolitan access networks," *Journal of Lightwave Technology*, vol. 17, no. 12, pp. 2431-2448, Dec. 1999.

[SaSi06] A. A. M. Saleh and J. M. Simmons, "Evolution toward the next-generation core optical network," *Journal of Lightwave Technology*, vol. 24, no. 9, pp. 3303-3321, Sep. 2006.

[SaSR05] H. P. Sardesai, Y. Shen, and R. Ranganathan, "Optimal WDM layer partitioning and transmission reach in optical networks," *Proceedings, Optical Fiber Communication (OFC'05)*, Anaheim, CA, Mar. 6-11, 2005, Paper OTuP4.

[ScAF01] D. A. Schupke, A. Autenrieth, and T. Fischer, "Survivability of multiple fiber duct failures," *Proceedings, Third International Workshop on the Design of Reliable Communication Networks (DRCN)*, Budapest, Hungary, Oct. 7-10, 2001, pp. 213-219.

[SDIR04] G. Swallow, J. Drake, H. Ishimatsu, and Y. Rekhter "Generalize Multiprotocol Label Switching (GMPLS) User-Network Interface (UNI): Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model," draft-ietf-ccamp-gmpls-overlay-05, Internet Engineering Task Force, Work In Progress, Oct. 2004.

[SeKS03] S. Sengupta, V. Kumar, and D. Saha, "Switched optical backbone for cost-effective scalable core IP networks," *IEEE Communications*, vol. 41, no. 6, pp. 60-70, Jun. 2003.

[ShGr04] G. Shen and W. D. Grover, "Segment-based approaches to survivable translucent network design under various ultra-long-haul system reach capabilities," *Journal of Optical Networking,* vol. 3, no. 1, pp. 1-24, Jan. 2004.

[ShWi06] F. B. Shepherd and P. J. Winzer, "Selective randomized load balancing and mesh networks with changing demands," *Journal of Optical Networking*, vol. 5, no. 5, pp. 320-339, May 2006.

[SiGS98] J. M. Simmons, E. L. Goldstein, and A. A. M. Saleh, "On the value of wavelength-add/drop in WDM rings with uniform traffic," *Proceedings, Optical Fiber Communication (OFC'98)*, San Jose, CA, Feb. 22-27, 1998, Paper ThU3.

[Simm99] J. M. Simmons, "Hierarchical restoration in a backbone network," *Proceedings, Optical Fiber Communication (OFC'99)*, San Diego, CA, Feb. 21-26, 1999, Paper TuL2.

[Simm02] J. M. Simmons, "Analysis of wavelength conversion in all-optical express backbone networks," *Proceedings, Optical Fiber Communication (OFC'02)*, Anaheim, CA, Mar. 17-22, 2002, Paper TuG2.

[Simm04] J. M. Simmons, "An introduction to optical network design and planning," *Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Short Course 216.

[Simm05] J. M. Simmons, "On determining the optimal optical reach for a long-haul network," *Journal of Lightwave Technology*, vol. 23, no. 3, pp. 1039-1048, Mar. 2005.

[Simm06] J. M. Simmons, "Network design in realistic 'all-optical' backbone networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 88-94, Nov. 2006.

[Simm07] J. M. Simmons, "Cost vs. capacity tradeoff with shared mesh protection in optical-bypass-enabled backbone networks," *Proceedings, Optical Fiber Communication/National Fiber Optic Engineers Conference (OFC/NFOEC'07)*, Anaheim, CA, Mar. 25-29, 2007, Paper NThC2.

[SiSa99] J. M. Simmons and A. A. M. Saleh, "The value of optical bypass in reducing router size in gigabit networks," *Proceedings, IEEE International Conference on Communications (ICC'99)*, Vancouver, British Columbia, Jun. 6-10, 1999, vol. 1, pp. 591-596.

[SiSa07] J. M. Simmons and A. A. M Saleh, "Network agility through flexible transponders," *IEEE Photonics Technology Letters*, vol. 19, no. 5, pp. 309-311, Mar. 1, 2007.

[SiSB01] J. M. Simmons, A. A. M. Saleh, and L. Benmohamed, "Extending Generalized Multi-Protocol Label Switching to configurable all-optical networks," *Proceedings, National Fiber Optic Engineers Conference (NFOEC'01)*, Baltimore, MD, Jul. 8-12, 2001, pp. 14-23.

[SiSM03] N. K. Singhal, L. H. Sahasrabuddhe, and B. Mukherjee, "Provisioning of survivable multicast sessions against single link failures in optical WDM mesh networks," *Journal of Lightwave Technology,* vol. 21, no. 11, pp. 2587-2594, Nov. 2003.

[SMCS05] A. G. Striegler, M. Meissner, K. Cveček, K. Sponsel, G. Leuchs, and B. Schmauss, "NOLM-based RZ-DPSK signal regeneration," *IEEE Photonics Technology Letters*, vol. 17, no. 3, pp. 639-641, Mar. 2005.

[SoPe02] H. Soliman and C. Peyton, "An efficient routing algorithm for all-optical networks with turn constraints," *IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, Fort Worth, TX, Oct. 12-16, 2002, pp. 161-166.

[SrSS02] M. Sridharan, R. Srinivasan, and A. K. Somani, "Dynamic routing with partial information in mesh-restorable optical networks," *Proceedings, Sixth Working Conference on Optical Networks Design and Modelling (ONDM)*, Torino, Italy, Feb. 4-6, 2002.

[StBa99] T. E. Stern and K. Bala, *Multiwavelength Optical Networks*, Reading, MA: Addison-Wesley Longman Inc,1999.

[SuAS96] S. Subramaniam, M. Azizoglu, and A. K. Somani, "All-optical networks with sparse wavelength conversion," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 544-557, Aug. 1996.

[SuTa84] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325-336, 1984.

[Suur74] J. W. Suurballe, "Disjoint paths in a network," *Networks,* vol. 4, pp. 125–145, 1974.

[TCFG95] R. W. Tkach, A. R. Chraplyvy, F. Forghieri, A. H. Gnauck, and R. M. Derosier, "Four-photon mixing and high-speed WDM systems," *Journal of Lightwave Technology*, vol. 13, no. 5, pp. 841-849, May 1995.

[Tekt01] Tektronix, *SONET Telecommunications Standard Primer*, 2001, www.tek.com/Measurement/App_Notes/SONET.

[Telc05] Telcordia, *Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria*, GR–253–CORE, Issue 4, Dec. 2005.

[TeRo03] J. Teng and G. N. Rouskas, "A comparison of the JIT, JET, and Horizon wavelength reservation schemes on a single OBS node," *Proceedings, The First International Workshop on Optical Burst Switching (WOBS)*, Dallas, TX, Oct. 16, 2003.

[ThSo02] S. Thiagarajan and A. K. Somani, "Traffic grooming for survivable WDM mesh networks," *Optical Networks Magazine*, vol. 3, no. 3, pp. 88-98, May/Jun. 2002.

[TkCh94] R. W. Tkach and A. R. Chraplyvy, "Dispersion and nonlinear effects in lightwave systems," *Proceedings, 7th Annual Meeting of the IEEE LEOS*, Boston, MA, Oct. 31-Nov. 3, 1994, vol. 1, pp. 192-193.

[Tomk02] I. Tomkos, et al., "Ultra-long-haul DWDM network with 320x320 wavelength-port 'Broadcast & Select' OXCs," *Proceedings, European Conference on Optical Communication (ECOC'02)*, Copenhagen, Denmark, Sep. 8-12, 2002, vol. 5, pp. 1-2.

[TsMT05] I. Tsirilakis, C. Mas, and I. Tomkos, "Cost comparison of IP/WDM vs. IP/OTN for European backbone networks," *Proceedings, International Conference on Transparent Optical Networks (ICTON'05)*, Barcelona, Spain, Jul. 3-7, 2005, pp. 46-49.

[TzZT03] A. Tzanakaki, I. Zacharopoulos and I. Tomkos, "Optical add/drop multiplexers and optical cross-connects for wavelength routed networks," *Proceedings, International Conference on Transparent Optical Networks (ICTON'03)*, Warsaw, Poland, Jun. 29-Jul. 3, 2003, pp. 41-46.

[VAAD01] W. Van Parys, P. Arijs, O. Antonis, and P. Demeester, "Quantifying the benefits of selective wavelength regeneration in ultra long-haul WDM networks," *Proceedings, Optical Fiber Communication (OFC'01)*, Anaheim, CA, Mar. 19-22, 2001, Paper TuT4.

[VaPD04] J. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*, San Francisco, CA: Morgan Kaufmann, 2004.

[Verb05] S. Verbrugge, et al., "Modeling operational expenditures for telecom operators," *Proceedings, Conference on Optical Network Design and Modeling (ONDM'05),* Milan, Italy, Feb. 7-9, 2005, pp. 455-466.

[Voss92] S. Voss, "Steiner's problem in graphs: Heuristic methods," *Discrete Applied Mathematics*, vol. 40, no. 1, 1992, pp. 45-72.

[WASG96] R. E. Wagner, R. C. Alferness, A. A. M. Saleh, and M. S. Goodman, "MONET: Multiwavelength optical networking," *Journal of Lightwave Technology,* vol. 14, no. 6, pp. 1349-1355, Jun. 1996.

[Waxm88] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications,* vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[WBSK03] S. T. Wilkinson, E. B. Basch, V. Shukla, P. Kubat, S. Raguram, and P. Limaye, "SONET mesh network architecture," *Proceedings, National Fiber Optic Engineers Conference (NFOEC'03)*, Orlando, FL, Sep. 7–11, 2003, pp. 293-302.

[WeCZ05] Y. Wen, V. W. S. Chan, and L. Zheng, "Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3358-3371, Oct. 2005.

[Welc06] D. F. Welch, et al., "The realization of large-scale photonic integrated circuits and the associated impact on fiber-optic communication systems," *Journal of Lightwave Technology*, vol. 24, no. 12, pp. 4674-4683, Dec. 2006.

[Will06] A. E. Willner, "The optical network of the future: Can optical performance monitoring enable automated, intelligent and robust systems?" *Optics and Photonics News*, pp. 30-35, Mar. 2006.

[WLYK04] D. Wang, G. Li, J. Yates, and C. Kalmanek, "Efficient segment-by-segment restoration," *Proceedings, Optical Fiber Communication (OFC'04)*, Los Angeles, CA, Feb. 22-27, 2004, Paper TuP2.

[WSGM03] I. Widjaja, I. Saniee, R. Giles, and D. Mitra, "Light core and intelligent edge for a flexible, thin-layered, and cost-effective optical transport network," *IEEE Communications Magazine*, vol. 41, no. 5, pp. S30-S36, May 2003.

[WuSF06] M. C. Wu, O. Solgaard, and J. E. Ford, "Optical MEMS for lightwave communication," *Journal of Lightwave Technology*, vol. 24, no. 12, pp. 4433-4454, Dec. 2006.

[XuQX07] D. Xu, C. Qiao, and Y. Xiong, "Ultrafast potential-backup-cost (PBC)-based shared path protection schemes," *Journal of Lightwave Technology,* vol. 25, no. 8, pp. 2251-2259, Aug. 2007.

[XuXQ03] D. Xu, Y. Xiong, and C. Qiao, "Novel algorithms for shared segment protection," *IEEE Journal on Selected Areas in Communications,* vol. 21, no. 8, pp.1320-1331, Oct. 2003.

[XXQL03] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Trap avoidance and protection schemes in networks with shared risk link groups," *Journal of Lightwave Technology*, vol. 21, no. 11, pp. 2683-2693, Nov. 2003.

[YaRa05a] X. Yang and B. Ramamurthy, "Dynamic routing in translucent WDM optical networks: The intradomain case," *Journal of Lightwave Technology*, vol. 23, no. 3, pp. 955-971, Mar. 2005.

[YaRa05b] W. Yao and B. Ramamurthy, "Survivable traffic grooming with path protection at the connection level in WDM mesh networks," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 2846-2853, Oct. 2005.

[YaYo05] H. Yang and S. J. B. Yoo, "All-optical variable buffering strategies and switch fabric architectures for future all-optical data routers," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3321-3330, Oct. 2005.

[Yen71] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

[Yue91] M. Yue, "A simple proof of the inequality FFD(L) ≤ (11/9)OPT(L) + 1, for all L, for the FFD bin-packing algorithm," *Acta Mathematicae Applicatae Sinica*, vol. 7, no. 4, pp. 321–331, Oct. 1991.

[ZaJM00] H. Zang, J. P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Optical Networks Magazine*, vol. 1, no. 1, pp. 47-60, Jan. 2000.

[ZhMu03] K. Zhu and B. Mukherjee, "A review of traffic grooming in WDM optical networks: Architectures and challenges," *Optical Networks Magazine*, vol. 4, no. 3, pp. 55-64, Mar./Apr. 2003.

[ZhQi98] X. Zhang and C. Qiao, "Wavelength assignment for dynamic traffic in multi-fiber WDM networks," *Proceedings, International Conference on Computer Communications and Networks (ICCCN'98)*, Lafayette, LA, Oct. 12-15, 1998, pp. 479 – 485.

[ZhZM05] K. Zhu, H. Zhu, and B. Mukherjee, *Traffic Grooming in Optical WDM Mesh Networks*, New York, NY: Springer, 2005.

[ZhZM06] J. Zhang, K. Zhu, and B. Mukherjee, "Backup reprovisioning to remedy the effect of multiple link failures in WDM mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 57-67, Aug. 2006.

[ZTTD02] Y. Zhang, K. Taira, H. Takagi, and S. K Das, "An efficient heuristic for routing and wavelength assignment in optical WDM networks," *Proceedings, IEEE International Conference on Communications (ICC'02)*, New York, NY, Apr. 28-May 2, 2002, pp. 2734-2739.

[ZZZM03] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 285-299, Apr. 2003.

Index