Study on performance efficient trees for multicast communication

Yong, Kelvin Keen Mun

2009

Yong, K. K. M. (2009). Study on performance efficient trees for multicast communication.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20611>

<https://doi.org/10.32657/10356/20611>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**A STUDY ON PERFORMANCE EFFICIENT TREES
FOR MULTICAST COMMUNICATION**

YONG KEEN MUN KELVIN

School of Electrical and Electronics Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2009

Statement of Originality

I hereby declare that the content of this thesis is the result of original work done by me and has not been submitted for a higher degree to any other University or Institution.

YONG KEEN MUN KELVIN

01-MAY-2009

Nanyang Technological University

2009

Abstract

The usage of multicasting over the Internet has increased over the years due to its more efficient delivery of traffic from a single source to multiple destinations via delivery trees. The performance of the delivery tree for each multicast session has a large and direct influence on the overall performance offered by the multicast services. In this thesis, we study the development of performance efficient trees for Internet multicast communication for both IP and Optical domains.

For the first part of our work, we focus on constructing non-delay and delay-constrained trees for multiple multicast sessions from multiple groups using a single shared-tree as the basis of the construction. We propose to first construct an efficient spanning shared-tree and then perform pruning on it to obtain the required delivery tree for a multicast session spanning only to its destinations. Since there is no need to re-construct each session's tree from scratch, a significant amount of tree's ready time can be reduced, leading to more efficient utilization of bandwidth resources and minimizing of race condition. We also propose a new method to "refine" the cost-efficiency of the tree. We then proceed to develop delay-constrained multicast trees for dynamic membership multicasting where members of a particular multicast group can join or leave the group. Rearrangement of the tree after either a destination leaves or joins the group is critical because rearranging the tree for the updated destination lists will achieve optimal cost but incur significant disruptions to the remaining destinations. In the past, only time inefficient reactive rearrangements with no disruptions control have been proposed. Hence, we

propose proactive rearrangements with disruption controls to ensure that such rearrangements can be carried out within a shorter time frame and with a controlled number of disruptions.

Subsequently, we extend the tree's development to optical domain. We propose a new algorithm to quickly place wavelength conversion nodes (WCC) and multicast-capable (MC) nodes in an optimal fashion such that the lowest possible blocking probability for delivering the traffic from an incoming set of multicast sessions can be achieved with a limited number of WCC and MC nodes. Since splitters are difficult and expensive to fabricate, and wavelength converters are expensive as well, optical multicasting can be supported in a more efficient and less complex manner by limiting both WCC and MC nodes required in the network. We also investigate the dynamic routing and wavelength assignment problem for delay-constrained optical multicasting. Currently, tree routing and wavelength assignment are considered two sub-problems and solved independently. Since they are closely related to each other, solving the two sub-problems independently will not achieve optimal results in terms of routing and wavelength usage. Also, the number of wavelength conversions required is best to be kept at a low level. Hence, we propose the use of a generic auxiliary graph consisting of multiple layers reflecting the presence and absence of MC and WCC nodes to solve the two sub-problems concurrently to achieve better efficiency in blocking, wavelength usage and average delay from source.

Acknowledgement

First and foremost, I would like to thank and express my sincere and deep gratitude to my supervisors, Associate Professor Cheng Tee Hiang and Associate Professor Poo Gee Swee. Their intensive knowledge and expertise in the field of communications have inspired me to do well for my PhD studies. Their strong support and patient guidance are critical to the successful completion of my journey towards a PhD degree; and I have gained immensely from their great mentorship and professionalism.

Next, I wished to thank my family and my wife, Gean Chu for their love, understanding, strong encouragement and support. They have indeed helped me to brave through the countless challenges and uncertainties faced during my studies, and it will be unthinkable without their encouragement.

I wished to thank my research colleagues, Dr. Yin Yongning, Dr. Wang Haibo, Dr. Zhou Yinzhu, Mr. Chen Sheng and Mr. He Xuzhou for their friendship, and encouragement which make my studies a more enjoyable one.

Last but not least, I am grateful to Nanyang Technological University (NTU) for the scholarship and an enjoyable environment to study. NTU has been like a second parent to me, and has nurtured me to be a confident and knowledgeable person. Also, I would like to thank all personnel from Network Technology Research Centre (NTRC) for their help during my studies.

Yong Keen Mun Kelvin

01-May-2009

Contents

Abstract	iii
Acknowledgements	v
List of Figures	xi
List of Tables	xiv
Acronyms	xv
Chapter 1: Introduction	1
1.1 Background	1
1.1.1 Internet Multicast	1
1.1.2 Optical Internet Multicast	7
1.1.3 Types of Multicast Membership	10
1.1.4 Delay Constrained Multicast	12
1.1.5 Performance Metrics used in Multicast	13
1.1.6 Performance Efficient Trees in Multicast	14
1.2 Motivations	15
1.3 Objectives	18
1.4 Contributions of the Thesis	20
1.5 Organization of the Thesis	24
Chapter 2: Literatures Review	26
2.1 Overview of Internet Multicast	26
2.2 Minimum Cost Tree Construction in Multicast	28
2.2.1 Minimum Cost Tree Multicast	28
2.2.2 Minimum Spanning Tree Algorithms	29
2.2.3 Shortest Path Algorithm	30
2.2.4 Minimum Cost Tree Algorithms using Single Tree Construction	31
2.2.5 Minimum Cost Tree Algorithms using Shared-Tree Construction	37
2.3 Minimum Cost Tree Construction in Dynamic Membership Multicast	39
2.3.1 Dynamic Membership Multicast	39
2.3.2 Performance Metrics used for Dynamic Membership Multicast	39
2.3.3 Algorithms for Minimum Cost Tree Construction in Dynamic Membership Multicast	41
2.3.3.1 Non-Rearrangeable Algorithms	42
2.3.3.2 Rearrangeable Algorithms	45
2.4 Multicast Capable (MC) and Wavelength Conversion Capable (WCC) Nodes Placement in Optical Multicast.....	54
2.4.1 Performance Metrics used for MC and WCC Nodes Placement Problems	54
2.4.2 Placement of Multicast Capable (MC) Nodes	55
2.4.3 Placement of Wavelength Conversion Capable (WCC) Nodes	57

2.4.4	Placement of Multicast Capable (MC) and Wavelength Conversion Capable (WCC) Nodes	62
2.5	Dynamic Multicast Routing and Wavelength Assignment (MC-RWA) in Optical Multicast	63
2.5.1	Algorithms for MC-RWA Problem	64
2.6	Summary	73
Chapter 3: Efficient Heuristics for the Construction of Minimum Cost Trees in Multi-Groups Multicast		76
3.1	Introduction	76
3.2	Background	79
3.2.1	Existing heuristics based on multiple single-tree construction model (MSi-T)	79
3.2.2	Existing heuristics based on multiple shared-tree construction model (MSh-T)	82
3.2.3	Proposed heuristics based on common shared-tree construction model (CSh-T)	83
3.3	Details of Proposed Algorithms based on CSh-T	84
3.3.1	Problem Formulation	84
3.3.1.1	Network Model	84
3.3.1.2	Problem Specification	84
3.3.2	Description of Proposed Algorithms MGST-T / DC-MGST-T	86
3.3.2.1	Stage 1: Cost-Efficient Spanning Shared-Tree Construction	87
3.3.2.2	Stage 2: Resultant Tree Construction	92
3.3.2.3	Stage 2 Improvement: “Trace back” on Resultant Tree Construction	98
3.3.2.4	Stage 3: Connection of all receiver nodes not connected in Stage 2	101
3.4	Simulation Results and Discussion	107
3.4.1	Performance Metrics	107
3.4.1.1	Session Blocking Probability (<i>SBP</i>)	107
3.4.1.2	Cost Competitiveness (<i>CC</i>)	107
3.4.1.3	Overheads Required per Unblocked Session (<i>K</i>)	108
3.4.1.4	Tree’s Ready Time per Unblocked Session (<i>T_i</i>)	109
3.4.2	Simulation Model	109
3.4.2.1	Random Graphs Generation	109
3.4.2.2	Simulation Parameters	110
3.4.3	Discussion of Results	112
3.4.3.1	Comparison between DC-MGST-T and STAR in terms of <i>SBP</i>	113
3.4.3.2	Comparison between DC-MGST-T and STAR in terms of <i>CC</i>	115
3.4.3.3	Comparison between DC-MGST-T and STAR in terms of <i>K</i>	116
3.4.3.4	Effect of Uniform DC and Membership Size	119

3.4.3.5	Effect of Non-Uniform DC	125
3.5	Summary	130
Chapter 4: Proactive Rearrangement in Delay Constrained Dynamic Membership Multicast		133
4.1	Introduction	133
4.2	Overview of dynamic membership multicast in Internet	136
4.2.1	Dynamic Membership's Scenarios	136
4.2.1.1	Existing-Tree (last updated tree of a session) Interior Node Added as Member	136
4.2.1.2	Node not Part of the Existing-Tree Added as Member	137
4.2.1.3	Non Leaf member node deletion	137
4.2.1.4	Leaf member node deletion	137
4.2.2	Effective existing reactive rearrangement control algorithm: CRCDM	137
4.3	Proposed PRDCDMM Algorithm	140
4.3.1	Problem formulation	140
4.3.1.1	Network Model	140
4.3.1.2	Cost and Delay Model	140
4.3.1.3	Update Requests Model	141
4.3.1.4	Disruption Model	142
4.3.1.5	Problem Specification	143
4.3.2	Algorithm Description	144
4.3.3	Algorithm Details	146
4.3.3.1	Traffic-Ready Link Graph (TRLG) Computation	146
4.3.3.2	Node Addition Algorithm	147
4.3.3.3	Node Removal Algorithm	147
4.3.3.4	Hybrid Cost Tree Rearrangement-Full Rearrangement (HCTR-F)	147
4.3.3.5	Definition of Rearrangement-sub-tree (R-sub-tree)	152
4.3.3.6	R-sub-tree rearranged region and parameters computation	154
4.3.3.7	Definition of Disruptions Limit per update	154
4.3.3.8	Hybrid Cost Tree Rearrangement-partial rearrangement (HCTR-P)	155
4.3.3.9	Reservation channels assignment mode	156
4.3.3.10	Non uniform channels assignment mode	157
4.3.3.11	Modification to reduce total disruptions in PRDCDMM	158
4.4	Simulation results and discussions	160
4.4.1	Performance metrics for disruptions and tree costs	160
4.4.1.1	Cost Competitiveness (CC)	160
4.4.1.2	Total Disruptions (TD)	161
4.4.1.3	Maximum Disruptions (MD)	161
4.4.1.4	Disruptions Competitiveness (DC)	162
4.4.1.5	CC-DC Performance Index	162
4.4.2	Performance metric for resource utilizations	163

4.4.2.1	Redundant traffic-ready Links (RL)	163
4.4.2.2	Average Maximum Effective traffic-ready Link Utilization (AMELU)	163
4.4.3	Simulation Model	164
4.4.3.1	Random network generation	164
4.4.3.2	Update request generation	164
4.4.3.3	Delay constraint parameters	165
4.4.3.4	Simulation parameters	165
4.4.4	Simulation results and discussions	166
4.5	Summary	184
Chapter 5: Optimal Placement of MC and WCC nodes in M-OVPN 188		
5.1	Introduction	188
5.1.1	Our results	190
5.1.2	Previous work	191
5.1.3	Chapter's outline	193
5.2	M-OVPN architecture	194
5.3	Approximate blocking model for sparse multicast and sparse conversion network	196
5.3.1	Analytical model	197
5.3.2	Numerical method	204
5.4	Proposed Placement Heuristics	207
5.4.1	MC-Removal heuristic	207
5.4.2	WCC-removal heuristic	208
5.4.3	MWC-Removal heuristic (hybrid-mode)	209
5.4.4	WMC-Removal heuristic (hybrid-mode)	209
5.4.5	Dual-Removal heuristic (dual-mode)	209
5.5	Experiments and results	211
5.6	Summary	226
Chapter 6: Dynamic Multicast Routing and Wavelength Assignment with Minimal Conversions in Delay Constrained WDM Network 228		
6.1	Introduction	228
6.2	Problem Formulation	230
6.2.1	Network Model	230
6.2.2	Problem Description	231
6.3	Proposed Algorithm	232
6.3.1	Generic Layered Auxiliary Graph Model	232
6.3.2	Proposed Algorithm: Multi-Layers Tree (ML)	233
6.3.2.1	Stage 1: Minimum Weight Tree construction	234
6.3.2.2	Stage 2: Pruning	235
6.3.2.3	Stage 3: Trace Back	236
6.3.3	Contribution	238
6.4	Simulation results and discussion	240
6.4.1	Random network generation	240
6.4.2	Performance Metrics	240
6.4.2.1	Session blocking probability (SBP)	240
6.4.2.2	Session Wavelength Conversions (SWC)	241
6.4.2.3	Average Session's Receiver Delay (ASRD)	241

6.4.3 Comparison algorithms	241
6.4.3.1 Minimum delay tree (MDT)	242
6.4.3.2 Maximum capacity residue tree (MCRT)	243
6.4.3.3 Minimum wavelength conversions tree (MWCT)	243
6.4.3.4 Algorithm from Chen et al.....	244
6.4.4 Simulation Results	245
6.4.4.1 For Scenarios where all nodes are MC and WCC	245
6.4.4.2 For Scenarios where only a fraction of the nodes are MC and WCC	252
6.5 Summary	255
Chapter 7: Conclusion and Future Work	256
7.1 Conclusion	256
7.2 Future Research	263
Author's Publications	266
Bibliography	268 - 282

List of Figures

Figure 1-1: Multicast vs. Multi-Unicast and Broadcast	3
Figure 1-2: An example of MPLS	6
Figure 1-3: An example of MPLS Multicast	7
Figure 1-4: Light-Tree and Multi-Lightpaths Tree	9
Figure 3-1: An example of each step of the shared-tree construction	87
Figure 3-2: A random network of 8 nodes.....	89
Figure 3-3: Phase 1 to 4 in stage 1 of MGST-T / DC-MGST-T.....	90
Figure 3-4: Phase 5 to 8 in stage 1 of MGST-T / DC-MGST-T.....	92
Figure 3-5: For a multicast session with C as the source node, and {E, G, F, A, B} as the receiver nodes, the tree constructed by (a) MGST-T /DC-MGST-T, (b) MCT and (c) CBT.....	97
Figure 3-6: For a multicast session with C as the source node, and {F, B} as the receiver nodes, the tree constructed by (a) MGST-T /DC-MGST-T, (b) MCT and (c) CBT.....	97
Figure 3-7: Performing “Trace Back” in stage 2 of MGST-T / DC-MGST-T for multicast session with source at node C to destination node B and F.....	99
Figure 3-8: An example of appending minimum delay path to the resultant tree.....	104
Figure 3-9: The final tree after removal of duplicate edges and loop.....	104
Figure 3-10: Effect of Uniform DC and Membership Size on SBP in a 100 nodes network.....	120
Figure 3-11: Effect of Uniform DC and Membership Size on CC in a 100 nodes network.....	121
Figure 3-12: Effect of Uniform DC and Membership Size on K in a 100 nodes network.....	123
Figure 3-13: Effect of Uniform DC and Membership Size on ATi in a 100 nodes network.....	124
Figure 4-1: PRDCDMM Blocks Diagram.....	145
Figure 4-2: An example of three R-sub-trees.....	153
Figure 4-3: Cost comparison (uniform delay constraint =50).....	167
Figure 4-4: Tree’s Setup Time comparison (uniform delay constraint =50).....	168
Figure 4-5: Cost comparison under different uniform delay constraint in a network of 100 nodes.....	169
Figure 4-6: Cost comparison under different network size when delay constraint is non-uniform at 50 with range of 30.....	170
Figure 4-7: CC comparison between UA PRDCDMM and CRCDDM under uniform delay constraint of 50.....	171

List of Figures

Figure 4-8: Disruptions comparison between UA PRDCDM and CRCDM under uniform delay constraint of 50.....	172
Figure 4-9: UA PRDCDM and CRCDM comparison in maximum disruptions under uniform delay constraint of 50.....	172
Figure 4-10: UA PRDCDM and CRCDM comparison in CC-DC Performance Index under uniform delay constraint of 50.....	174
Figure 4-11: Average Tree's Setup Time Results.....	174
Figure 4-12: Maximum Disruptions Comparison between I-CRCDM and UA PRDCDM while achieving the difference in CC-DC Performance Index between them less than 0.3%.....	177
Figure 4-13: Average tree's rearrangement processing time (tree's setup time) between I-CRCDM and UA PRDCDM while achieving the difference in CC-DC Performance Index between them less than 0.3%.....	178
Figure 4-14: Non-UA and UA PRDCDM comparison in terms of CC-DC Performance Index for 100 nodes network under uniform delay constraint of 50.....	179
Figure 4-15: Non-UA and UA PRDCDM comparison in terms of RL for 100 nodes network under uniform delay constraint of 50.....	179
Figure 4-16: Non-UA and UA PRDCDM comparison in terms of AMELU for 100 nodes network under uniform delay constraint of 50.....	180
Figure 4-17: Effect of network size on CC-DC performance index comparison.....	180
Figure 4-18: Effect of network size on redundant links produced between Non-UA and UA PRDCDM.....	181
Figure 4-19: Effect of network size on AMELU between Non-UA and UA PRDCDM.....	181
Figure 5-1: An Example of M-OVPN.....	195
Figure 5-2: An Example of Multi-Segments in a Tree/Path.....	203
Figure 5-3: Pseudo Code of <i>Relax-BP_{avg}</i>	206
Figure 5-4: Pseudo Code of all Proposed "Removal".....	210
Figure 5-5: NSFNET. Each of the nodes can be a provider's edge in M-OVPN.....	211
Figure 5-6: MC-Removal Heuristic for Optimal MC nodes Placement (Using $e_R = 0.05$).....	214
Figure 5-7: Complexity Comparison between "Removal" and "Add" heuristic.....	216
Figure 5-8: WCC-Removal Heuristic for Optimal WCC nodes Placement (Using $e_R = 0.05$).....	218
Figure 5-9: MC-Removal Heuristic with varying Load.....	220
Figure 5-10: WCC-Removal Heuristic with varying Load.....	221
Figure 5-11: Blocking Performance of WMC-Removal Heuristic vs. k for different c (using $e_R = 0.05$).....	222

List of Figures

Figure 5-12: Blocking Performance of MWC-Removal Heuristic vs. c for different k (using $e_R = 0.05$).....	223
Figure 5-13: Blocking Performance vs. number of Dual-nodes used for different heuristics (using $e_R = 0.05$).....	226
Figure 6-1: An example of GLAGM. Note that the above GLAGM shows that wavelength channel 2 is not available on link CB, DA, BC and BA.....	232
Figure 6-2: (a) Original constructed tree. (b) Tree after performing “trace back”	237
Figure 6-3: (a) Routing tree using existing algorithms. (b) One possible configuration of splitters and converters in node C. (c) Another possible configuration of splitters and converters in node C. (d) Routing tree using our proposed algorithm.....	239
Figure 6-4: Average SBP vs. Traffic Load ($W = 3, \Delta = 20$).....	245
Figure 6-5: Average SWC vs. Traffic Load ($W = 3, \Delta = 20$).....	246
Figure 6-6: Average ASRD vs. Traffic Load ($W = 3, \Delta = 20$).....	247
Figure 6-7: Average SBP vs. W ($L = 20 \text{ Er}, \Delta = 20$).....	248
Figure 6-8: Average SWC vs. W ($L = 20 \text{ Er}, \Delta = 20$).....	249
Figure 6-9: Average ASRD vs. W ($L = 20 \text{ Er}, \Delta = 20$).....	249
Figure 6-10: Average SBP vs. Δ ($L = 20 \text{ Er}, W = 3$).....	250
Figure 6-11: Average SWC vs. Δ ($L = 20 \text{ Er}, W = 3$).....	251
Figure 6-12: Average ASRD vs. Δ ($L = 20 \text{ Er}, W = 3$).....	252
Figure 6-13: Average SBP vs. MC and WCC nodes in Network ($L = 20 \text{ Er}, \Delta = 25, W = 3$).....	253
Figure 6-14: Average SWC vs. MC and WCC nodes in Network ($L = 20 \text{ Er}, \Delta = 25, W = 3$).....	254
Figure 6-15: Average ASRD vs. MC and WCC nodes in Network ($L = 20 \text{ Er}, \Delta = 25, W = 3$).....	254

List of Tables

Table 2-1: Summary of existing reviewed heuristics.....	72
Table 3-1: MCPs of the network shown in figure 3.2 using Dijkstra algorithm.....	90
Table 3-2: SBP for STAR (Membership size: 50% of network size).....	114
Table 3-3: SBP for DC-MGST-T(Membership size: 50% of network size).....	114
Table 3-4: SBP_{STAR} - $SBP_{DC-MGST-T}$ (Membership size: 50% of network size).....	114
Table 3-5: CC_{STAR} - $CC_{DC-MGST-T}$ (Membership size: 50% of network size).....	116
Table 3-6: K_{STAR} - $K_{DC-MGST-T}$ ($\log_{10}$) (Membership size: 50% of network size).....	118
Table 3-7: ATi_{STAR} - $ATi_{DC-MGST-T}$ (Membership size: 50% of network size).....	118
Table 3-8: Effect of Non-Uniform DC on SBP in 100 nodes network.....	127
Table 3-9: Effect of Non-Uniform DC on CC in 100 nodes network.....	127
Table 3-10: Effect of Non-Uniform DC on K in 100 nodes network.....	128
Table 3-11: Effect of Non-Uniform DC on ATi in 100 nodes network.....	129
Table 4-1: Simulation Parameters.....	166
Table 4-2: Number of Traffic-Ready Links Assigned with Channels.....	183
Table 4-3: Performance in Non-uniform Delay Constraint Environment with delay range of 10 and network size of 100 nodes.....	183
Table 4-4: Performance under Non-uniform Delay Constraint Environment with delay range of 20 and network size of 100 nodes.....	184
Table 5-1: List of Notations used in this Chapter.....	196
Table 5-2: Sample of 10 Multicast Sessions with Corresponding Routing Trees.....	212
Table 5-3: MC-Criticality Weight of each Node in Backbone.....	213
Table 5-4: MC Nodes Placement Configuration for All Possible Numbers of Allowed MC Nodes in Backbone.....	213
Table 5-5: WCC-Criticality Weight of each Node in Backbone.....	216
Table 5-6: WCC Nodes Placement Configuration for All Possible Number of Allowed WCC Nodes in Backbone.....	217
Table 5-7: MC and WCC Nodes Placement Configuration with varying Traffic Loads. (Node with the largest weight will be placed with MC or WCC node first).....	219
Table 5-8: Dual-Criticality Weight of Each Node in Backbone.....	224

Acronyms

Acronyms

AMELU	Maximum Effective Traffic-Ready Link Utilization
ARIES	A Rearrangeable Inexpensive Edge-Based On-Line Steiner algorithm
ARP	All Nodes Reroute to Parent
ASRD	Average Session's Receiver Delay
ATM	Asynchronous Transfer Mode
BSMA	Bounded Shortest Multicast Algorithm
CBT	Core Based Tree
CC	Cost Competitiveness
CE	Client's Equipment
CKMB	Constrained KMB
CMCT	Centralized Minimum Cost Tree Heuristic
CR	Cost Reduction
CRCDM	Controlled Rearrangement for Constrained Dynamic Multicast
CRMR	Constrained Rearrangement Multicast Routing
CS	Cost Saving
CSh-T	Common Shared-Tree Construction Model
CSR	Cost to Service Ratio
DCMCP	Delay Constrained Minimum Cost Path
DCMCT	Delay Constrained Minimum Cost Tree
DC-MGST	Delay-constrained MGST
DC-MGST-T	DC-MGST with Trace-Back
DCDMM	Delay Constrained Dynamic Membership Multicast
DDBMA	Dynamic Delay Bounded Multicast Algorithm

Acronyms

DL	Disruption Limit per update
DMCT	Distributed Minimum Cost Tree Heuristic
DMR-DC	Delay Multicast Routing Problem under Delay-Constraints
DVMRP	Distance Vector Multicast Routing Protocol
DRNN	Destinations Reroute to Nearest Node
EBA	Edge-Bounded Algorithm
FAR	Fix Alternate Routing
FDDI	Fiber Distributed Data Interface
FEC	Forward Equivalence Class
FF	First-Fit
GLAGM	Generic Layered Auxiliary Graph Model
GMPLS	Generalized Multiprotocol Label Switching Protocols
GSDM	Geographic Spread Dynamic Multicast
HCTR-P	Hybrid Cost Tree Partial Rearrangement
HCTR-F	Hybrid Cost Tree Full-Rearrangement
HDTV	High Definition Television
I-CRCDM	Improved CRCDM
ILDR	Immediate Least Delay Replacement
IGMP	Internet Group Management Protocol
ILP	Integer Linear Programming
IP	Internet Protocol
KMB	A heuristic from [KMB81]
LCW	Link Criticality Weight
LDR	Least Delay Replacement
LLR	Least Load Routing

Acronyms

LRA	Lagrangian-Relaxation-Based Algorithm
LSP	Label Switch Path
LSR	Label Switch Router
MBPF	Minimum Blocking Probability First
MC	Multicast Capable
MCP	Minimum Cost Path
MCT	Minimum Cost Tree
MCRT	Maximum Capacity Residue Tree
MC-RWA	Multicast Routing and Wavelength Assignment
MD	Maximum Disruptions
MDP	Minimum Delay Path
MDT	Minimum Delay Tree
MGST	Multi-Groups Shared Tree algorithm
MGST-T	MGST with Trace-Back
MHCP	Minimum Hybrid Cost Path
MI	Multicast Incapable
MILP	Mixed Integer Linear Programming
ML	Multi-Layers Tree Algorithm
MPH	Minimum Path Heuristic
MPLS	Multiprotocol Label Switching Protocols
M-OVPN	Multicast Optical Virtual Private Network
MOSPF	Multicast Open Shortest Path First
MST	Minimum Spanning Tree
MSi-T	Single-Tree Construction Model
MSh-T	Multiple Shared-Tree Construction Model

Acronyms

MVPN	Multicast Virtual Private Network
MWCT	Minimum Wavelength Conversion Tree
Non-UA	Non-Uniform Channels Assignment Mode
NMCF	Nearest MC Node First
NWC	Non-Wavelength-Continuous
O-E-O	Optical-Electronics-Optical
OTMCF	On Tree MC Node First
OSPF	Open Shortest Path First
PE	Provider's Equipment
PIM	Protocol Independent Multicast
PRDCDMM	Proactive Rearrangement for Delay-constrained Dynamic Membership Multicast
PWC	Partial-Wavelength-Continuous
QF	Quality Factor
QoS	Quality of Service
RBP	Receiver Blocking Probability
RD	Resultant Disruption
RL	Redundant Traffic-Ready Link
RWA	Routing and Wavelength Assignment
RTP	Resultant Tree Path
SBP	Session Blocking Probability
SPH	Shortest Path Heuristic
SF	Selection Function
SLIM	Delay-constrained, Low-cost, Inexpensive Multicasting
SPT	Shortest Path Tree
STAR	Segment, Trim and Reconnect algorithm

Acronyms

SWC	Session Wavelength Conversions
TD	Total Disruptions
TOTU	Total outgoing Traffic Unicast
TRLG	Traffic-Ready Link Graph
UA	Uniform Channels Assignment Mode
VOD	Video on Demand
VOIP	Voice over IP
VPN	Virtual Private Network
VT	Virtual Trunk
VTDM	Virtual Trunk Dynamic Multicast
WCC	Wavelength Conversion Capable
WCP	Wavelength Converter Placement
WDM	Wavelength Division Multiplexing
WLR	Weight Limit Ratio

Chapter 1

Introduction

This thesis investigates the development of performance efficient traffic delivery trees for multicast communication in the Internet. In this chapter, the background, motivation, objectives, contributions and organization of the thesis are provided.

1.1 Background

1.1.1 Internet Multicast

In recent years, Internet [For06] has become an integral part of our lives. It is widely used both at work and at home, for information exchanges, learning, and leisure etc. When used properly, it will help us to improve productivity and shorten the distance and time. For example, an email sent via the Internet will be able to reach the intended recipient(s) within seconds compared to a postal (or snail) mail which will take at least several days to deliver. Another example is that online paperless applications can be processed automatically and almost instantly compared to traditional paper forms which will take longer time to be delivered and processed. A third example is that, with the advent of Internet technology, Voice over IP (VOIP) [RC99] via the Internet has become a much cheaper way to contact people afar and encourage more to engage in long distant voice communications. It is clear that Internet has revolutionized communication in our daily lives in the positive note.

Chapter 1

Communication within the Internet can be classified as unicast, broadcast and multicast communication. Unicast communication [For06] is most straightforward and is concerned with the communication between two nodes across the Internet. Broadcast communication [ME96] involves the communication between a single node and all the other nodes in the network. Although implementation is easy and straight-forward, broadcasting delivers traffic to all nodes in the network even if some of them are not the intended recipients. Multicast communication [Will99] is a subset of broadcasting. The difference compared to broadcasting is that multicasting delivers traffic via a traffic delivery tree only to the intended receiving nodes. For convenience, we will henceforth refer to traffic delivery tree simply as tree unless specified otherwise.

Although multi-unicast can be applied to send data from a node to multiple nodes, redundant traffic may incur due to some unicast paths sharing the same links when delivering traffic. As observed in figure 1-1, when source node A sends data to receiver node C and E, multicast requires only one packet to be sent on each link in the tree with packet duplication at node D. On the other hand, multi-unicast requires two packets to be sent on each link, one for each receiver node. Hence, multi-unicast incurs an extra packet on each link. This is undesirable as the bandwidth on that link for the extra packet can be used to carry other traffic to other receiver nodes. In other words, the links' bandwidth utilization is not optimized. From figure 1-1(c), broadcast sends packets to all nodes including non-receiver node F and G, and performs packet duplication at node B and D. This incurs extra packets in link (B,F) and link (B,G) since node

Chapter 1

F and G do not need to receive any data packets. Thus, it can be concluded from this simple example that multicast is more bandwidth efficient for node to multi-nodes communication compared to multi-unicast and broadcast.

Figure 1-1: Multicast vs. Multi-Unicast and Broadcast

Multicast is often implemented at the network layer because the Internet consists of different media types at the data-link layer, such as Asynchronous Transfer Mode (ATM), Fiber Distributed Data Interface (FDDI), Frame Relay, Ethernet and other networking technologies. In order to support multicast communication at the network layer, there is a need to define addressing, multicast routing and dynamic registration of the receiver nodes. For addressing, an address is required to communicate with a group of receiver

Chapter 1

nodes. As for multicast routing, a delivery tree is required to route the traffic from the source node to the receiver nodes. Finally, for dynamic registration of receivers, an efficient method should be put in place to register dynamically each receiver node to a multicast group so that traffic can be sent to the intended receiver nodes correctly.

Internet Protocol (IP) multicast [Will99] performs multicasting at the network layer where some of the routers duplicate the incoming traffic and send it out to the respective outgoing links in the delivery tree. In addition, IP multicast uses the dynamically allocated Class D addresses for addressing purposes and adopts Internet Group Management Protocol (IGMP) [Will99] to register receiver nodes dynamically. For multicast routing in IP multicast, some of the popular standards are Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF) and Protocol Independent Multicast (PIM).

DVMRP [Will99] adopts the Reverse Path Forwarding technique to send traffic to the receiver nodes. Upon receiving the traffic packet, each router will duplicate and flood the packet on all paths except the one from which the packet is received. Prune message is sent from the router up to the delivery tree if the sub-tree rooted at the router does not contain any receiver nodes. DVMRP will re-flood periodically to send the traffic to any new receiver nodes joining the multicast group. DVMRP can only work with its own DVMRP unicast routing protocol which is based purely on hop counts.

Chapter 1

MOSPF [Will99] is an extension of Open Shortest Path First (OSPF) unicast routing protocol. It makes use of multicast information stored in OSPF link state advertisements to build a delivery tree for each multicast session with a source-receivers pair. This tree has to be re-built if the link state changes or the tree state in the cache has expired. MOSPF can only be used in a network that employs OSPF as the unicast routing protocol.

PIM [Will99] consists of dense and sparse mode. Dense PIM uses Reverse Path Forwarding technique, which is similar to DVMRP except that it can work with whatever unicast protocol being used. Sparse PIM resembles Core Based Tree (CBT) [BFC93] where a rendezvous point is defined. The sender will send data to the rendezvous point for the latter to send out to the receiver nodes whenever it has something to send. The receiver nodes need to register with the rendezvous point to express its desire to receive a particular traffic stream.

Network-layer multicast routing protocols such as PIM, DVMRP and MOSPF are normally preferred for a datagram network such as the Internet; however, some multicast applications like Multicast Virtual Private Network (MVPN) [PGA03] and Optical Internet Multicast favor the virtual circuit approach. In the latter case, the virtual circuit can be seen as forming an overlay network over the datagram-based Internet. For the virtual circuit overlay network, a minimum cost tree could be constructed based on the Steiner-tree model.

To provide virtual circuit service in the Internet, Multiprotocol Label Switching Protocols (MPLS) [PGA03] is the preferred mean as it could support the unicast

Chapter 1

traffic of circuit-switched and packet-based switching applications. MPLS uses labels to forward traffic between the source node and the receiver node via a path of label switched routers (LSRs) known as label switched path (LSP). MPLS requires a routing protocol to determine the LSPs, and a signaling protocol to inform the switches which labels and links to be used for each LSP. As shown in figure 1-2, LSR #1 first determines the next hop and initial label (label 2) from the destination IP address of the packet and attaches the label to the packet before forwarding it to the next hop LSR. When the packet reaches the next hop LSR, LSR #2 will use the label in the packet to identify the next LSP, from which it determines the next hop and label (label 8). This continues until the packet reaches LSR #3. The label will then be stripped off from the packet before the packet is forwarded to the host.

Figure 1-2: An example of MPLS

As MPLS is designed for unicast communication, it has to be extended for multicast communication. In [AC05], the authors propose a protocol to extend MPLS for multicast communication by assuming that multicast group members are connected directly to edge-LSRs via LSPs and that branching nodes of the delivery tree are confined to these edge-LSRs. This implies that the tree will consist of an LSP connecting the source node and the branching node, and a few LSPs connecting the branching node to each receiver node as shown in an

Chapter 1

example in figure 1-3. LSR #2 in figure 1-3 duplicates packets according to the routing table and assigned them the corresponding MPLS labels.

Figure 1-3: An example of MPLS Multicast

1.1.2 Optical Internet Multicast

Due to the increasing bandwidth demand from multimedia applications, an optical Internet may be deployed in the foreseeable future to ensure that there is sufficient bandwidth to maintain reasonable QoS to the users. There are many possible ways to implement an optical Internet, which is still a topic of intense research. [Muk06], provides a good overview on this topic. One possible realization of an optical Internet is to employ optical fibers with large bandwidth capacity at the physical layer to route and switch IP packets belonging to the same Forward Equivalence Class (FEC)¹ in different optical wavelengths using wavelength division multiplexing (WDM). The assigned wavelength may be converted to another wavelength at any node along the route if the same wavelength has to be used to carry other traffic in the next hop. The path to be followed by the IP packets for a particular flow (or FEC) is collectively known as a light-path. Beside unicast, multicast communication

¹ FEC is an MPLS terminology which refers to a set of packets with similar and / or identical characteristics which may be forwarded the same way within the network.

Chapter 1

can be supported at the wavelength level in the optical Internet. As shown in figure 1-4, multicast traffic can be sent in the optical Internet either via a homogeneous or heterogeneous wavelength light-tree depending on wavelength availability and presence of wavelength converters or Optical-Electronics-Optical (O-E-O) [Mai08] conversions. A homogenous wavelength light-tree carries traffic to the receiver nodes using only a single wavelength throughout while a non-homogeneous one uses multiple wavelengths. If a link in the tree does not have any wavelength available or the wavelength for a traffic flow from the preceding link cannot be converted to the required wavelength for this link, the flow will be blocked and not sent to the receiver nodes. This suggests that efficient bandwidth utilization and the existence of wavelength conversion are essential to obtain low blocking probability for multicast traffic in the optical Internet.

In a transparent optical Internet or all-optical Internet [Mai08], multicasting takes place at the physical layer where incoming optical waveform arriving at a multicast-capable (MC) node is split and transmitted to its outgoing links in the tree. Any wavelength conversion is carried out directly at the physical optical layer in wavelength conversion capable (WCC) node. Thus, to support a transparent optical Internet efficiently, power splitters and wavelength converters are required. Power splitters are required to split the incoming signals before transmitting out to the receiver nodes. As the power intensity will be split in the process, the split signals need to be amplified to ensure that the signals arriving at the receiver nodes will be strong enough to be recognized and converted back to the original data. Taking into consideration the splitting

Chapter 1

loss and amplification requirement, power splitters become expensive as a result. In addition, wavelength converters which convert incoming signals from one wavelength to another are also expensive. Therefore, one immediate concern pertaining to deploying a transparent optical Internet is the number of power splitters and wavelength converters required to support an acceptable quality of service (QoS).

Figure 1-4: Light-Tree and Multi-Lightpaths Tree

In an opaque optical Internet [Mai08], multicasting takes place at the network layer with O-E-O conversions where incoming optical waveform arriving at a node is converted to electronics packet, and duplicated before converted back to optical waveforms and transmitted to all the outgoing paths. Likewise, for wavelength conversion, the optical waveform is converted electronically first before converted back to optical waveform tuned to another wavelength. The number of O-E-O conversions required is of concern in the opaque optical Internet. For completeness, one should understand that an optical Internet may

Chapter 1

adopt a combination of both transparent and opaque sub-networks where some nodes perform all-optical processing while others perform O-E-O conversions. Such a hybrid configuration is normally known as a translucent optical Internet [Mai08].

In a unicast optical network, Generalized Multiprotocol Label Switching Protocols (GMPLS) [FB05] is normally used as the data-carrying mechanism. GMPLS is an extension of MPLS for optical networks where the labels are not only restricted to numbers (up to 32 bits); they can be generalized in the form of wavelengths, frequency or any information which can distinctly differentiate each traffic flow. This increases scalability since more combination of labels can be formulated. Since the difference between GMPLS and MPLS is mainly in the labels and their values, and independent of the routing mechanism, the extension of MPLS for multicast as described in [AC05] can be applied for multicast communication using GMPLS in the Optical Internet.

1.1.3 Types of Multicast Membership

In multicast communication, there are two modes of membership. Membership here refers to the node's registration as member to a particular multicast group. A receiver node for a multicast session must be registered as a member of the multicast group which the session originated from. The two kinds of membership are static [Witt00] and dynamic membership [Witt00]. In static membership, the group of nodes registered as members for a multicast group will remain fixed whereas in dynamic membership, the group of nodes registered as members for a multicast group will change depending on users'

Chapter 1

connections to the application represented by that particular multicast group. Hence, in dynamic membership multicast, the receiver nodes composition for a multicast-session can change while no such changes will occur for static membership multicast.

Static multicast membership is able to construct cost optimal delivery trees using more direct and simple heuristics because of its fixed membership. Some examples of such heuristics can be found in [KMB81], [Ram96] and [YY05]. On the contrary, cost optimal delivery trees for dynamic membership multicast is far more difficult and complex to construct because the group of receivers is always changing for each multicast session. This means that in order to have a cost optimal tree for every multicast session, the delivery tree will need to be constructed for that particular group of receivers of the multicast session rather than using just one tree for the multicast group with a fix group of receivers for all sessions. The re-construction of the tree for each session will cause traffic disruption to the other receiver nodes which is highly undesirable. The duration of the disruption and the number of receiver nodes affected by the disruption are vital to the quality of the traffic delivered to the receiver nodes. Hence, a balance must be struck between achieving efficient performance tree and reasonable disruption. Some studies have been done in [BV97], [Jia98], [HLP98], [HZ98b] and [MN99] to construct cost optimal trees for dynamic membership multicast.

Due to the increasingly dynamic nature of applications where user is allowed to leave or join a multicast application, dynamic membership multicast is an

Chapter 1

important aspect to be considered in our studies of performance efficient multicast.

1.1.4 Delay Constrained Multicast

Beside dynamic membership, delay-constraint between the source node and each receiver node of a particular multicast session is also introduced for applications where time-sensitivity is of importance. Such applications may include real-time video conferencing, online gaming etc. The delay to deliver the traffic from the source node to each receiver node for a multicast session has to satisfy the specified delay-constraint. Such delay-constraint is usually determined by the different premium paid for the different quality classifications of multicast service. For example, a higher quality service will require the traffic delivery to be faster and hence has a stricter (smaller) delay-constraint to be satisfied when compared to a lower quality service.

There are many studies on delay constrained multicast, for example, [Jia98], [Xue99], [RMM99], [FZG01], [UZ02], [LMZ04], [LA05] and [YY05]. Basically, the delay-constraint ensures that the constructed delivery tree is able to deliver traffic to each receiver node within a certain delay. Usually the delay considered is only the propagation delay from the source to the receiver node since propagation delay contributes a major part of the overall delay compared to other types of delay within each node in the transmission path between the source and the receiver node. Moreover, the delay in each node is almost constant whereas the propagation delay depends largely on the selection of routing paths. Hence, propagation delay is generally considered in selecting the

Chapter 1

appropriate routing paths for the delivery tree to ensure that the delay to each receiver node is within the delay-constraint specified.

1.1.5 Performance Metrics used in Multicast

The performance metrics for determining the performance of the algorithms or heuristics proposed in multicast communication normally consist of cost scalability, bandwidth efficiency, delivery reliability in terms of blocking probability, and time delay. The cost of multicast is normally measured in terms of the delivery tree cost, and such cost may consist of the link cost or the number of devices -- power splitters and wavelength converters etc – required for optical multicast. Link cost is the transmission cost in a single link and can be a function of lease cost, fibers used, maintenance cost and other related cost. The service providers will dictate which cost metric they want to optimize and apply the cost function to each link in the network as the transmission cost.

Bandwidth efficiency is concerned with whether the bandwidth of individual links is used effectively. Hence, metrics related to such bandwidth usage as utilization and occupancy rate are usually selected to determine the bandwidth efficiency of the proposed algorithm. For delivery tree reliability, the blocking probability is used as the metric. The definition of blocking probability is in terms of either session blocking probability (SBP) or receiver blocking probability (RBP). In SBP, a particular multicast session is blocked as long as one of its receiver nodes failed to receive the traffic due to insufficient bandwidth or not able to satisfy some constraints such as the delay-constraint. In other words, the reliability in SBP is measured in terms of the number of

Chapter 1

blocked multicast sessions. For RBP, a particular multicast request is only blocked if all the receiver nodes failed to receive traffic. Hence, the reliability in RBP is measured in terms of the number of receiver nodes blocked rather than the number of sessions blocked.

For time efficiency, the objective is to perform a particular task within the shortest time interval. Hence the time metric is usually used to determine the proposed algorithm's execution complexity, and lower time taken by the algorithm indicates lesser complexity to execute the algorithm and thus able to perform more tasks within the same time interval. For QoS, the metrics used depend on the classification of the quality of service adopted. For example, if the QoS is expressed in terms of the delay which the receiver node receives traffic from the source node; the metric used will be the propagation delay between the source and the receiver node. If the QoS is in terms of the frequency in which receiver nodes are able to receive traffic, then the metric used will be the blocking probability which the traffic can be delivered to the receiver nodes.

1.1.6 Performance Efficient Trees in Multicast

Just like any other communication, the performance of multicast communication needs to be efficient in order to be supported economically. Efficient performance in this thesis is defined as achieving results not too far off from the optimal results of one or more performance metrics such as cost, blocking probability etc given any constraints or conditions imposed. The achieved results may not need to be the optimal results but must not be too far

Chapter 1

off from the optimal. The optimal or near optimal results obtained by the existing optimal algorithms and heuristics will be used as the benchmark for all the proposed algorithms where possible. As the configuration of each traffic delivery tree has direct impact on the multicasting performance, performance efficient trees need to be developed in order to achieve efficient performance multicast.

The purpose of achieving efficient performance multicast is to ensure that multicast can be supported economically or with minimal resource. The traffic delivery tree for each multicast request has to be constructed and updated fast so that good performance can be achieved without any side-effects from taking place and inefficient bandwidth utilization due to slow update of the tree can be avoided [RMM99]. In other words, the process to find and establish performance efficient trees in multicasting must be fast and not overly complex. Since the construction of optimal trees is known to be NP-complete, heuristic algorithms are preferred in order to develop performance efficient trees fast. We shall refer heuristic algorithms as algorithms or heuristic henceforth for convenience.

1.2 Motivations

The advantages of multicasting can only be fully exploited if it can be supported economically without any degradation of QoS. Although many works dealt with different areas of developing performance efficient trees in multicast communication, to our best knowledge the following four areas have not studied and are investigated in this thesis. The four areas are:-

Chapter 1

- Construction of cost-optimal Steiner-tree is known to be NP-complete.

Hence, the construction of a single cost-optimal Steiner-tree is time consuming and not desirable for cases in which the tree is required to be constructed and rearranged fast or when multiple trees are required to be constructed for multi-groups multicast. Shared-tree has been used for the construction of cost-optimal peer-to-peer paths, but what if a shared-tree is used for the construction of cost-optimal trees for multi-groups multicast? The resulting advantages will include faster construction of trees, and more robust and reliable services in multiple-groups multicast, especially so with dynamic membership multicast.

- In dynamic membership multicast, rearrangement of trees is vital to ensure that the multicast cost is kept minimal. If no rearrangement takes place, the tree configuration will be outdated leading to degrading performance in terms of cost which is undesirable. Existing algorithms adopt the reactive approach where rearrangement procedures take place if and only if an update request in the memberships arrives. The corresponding resulting disruptions caused from the existing rearrangement algorithms cannot be controlled directly. As certain amount of disruptions will still be allowed in most situations, an algorithm which can control the level of disruptions will be able to maximize the rearrangement while not compromising the performance. In addition, the reactive model will take a lengthy time to rearrange

Chapter 1

trees which might cause race conditions². Hence, a new model with disruptions control to rearrange the trees fast will provide better efficiency and more reliable services for multicast communication with dynamic membership support.

- In optical Internet multicast, the optimal placement of a limited number of wavelength-conversion-capable (WCC) nodes based on the assumption that all nodes are multicast-capable (MC) has been investigated. Also has been investigated is the optimal placement of a limited number of MC nodes based on the assumption that all nodes are WCC. It is interesting to explore if an algorithm can be devised to place both WCC and MC nodes optimally together since both are preferred to be deployed in limited numbers? Existing algorithms do not consider the impact of MC nodes placement on WCC placement and vice versa. Since the MC and WCC nodes both contribute to the blocking performance, a new methodology is required to consider the impact of MC nodes placement on WCC placement and vice versa for better accuracy and higher efficiency in limiting their numbers.
- In Optical Internet multicast, the multicast routing and wavelength assignment (MC-RWA) problem pertaining to obtaining efficient cost trees as well as efficient assignment of wavelengths has been intensively investigated. Efficient cost trees ensure that optical multicast can be supported economically while wavelength assignment efficiency

² A **race condition** is a flaw in a system or process whereby the output and/or result of the process are unexpectedly and critically dependent on the sequence or timing of other events.

Chapter 1

minimizes the blocking probability due to insufficient wavelength channels to deliver the traffic. Existing algorithms consider multicast routing and wavelength assignment independently and this will not achieve optimal performance in terms of cost and wavelength assignment. This is because wavelength assignment efficiency depends largely on the routed multicast tree, and the tree cost efficiency depends largely on the availability of the link. The availability of a link here refers to whether a link has any free wavelength channels or a particular wavelength channel to deliver the traffic. Hence, new algorithms are needed to perform both multicast routing and wavelength assignment concurrently to achieve efficient performance in optical Internet multicast.

1.3 Objective

The overall objective is to study the development of performance efficient trees in order to achieve efficient performance multicast in the Internet. Specifically, this thesis covers the following scopes to show that multicast in the Internet can be supported with efficient performance in one or more metrics in the presence of certain constraints.

1. Study the minimum cost tree construction problem for multi-groups multicast in both non-delay-constraint and delay-constraint scenarios, and develop efficient heuristic algorithms based on the shared-tree concept to construct minimum cost tree and delay constrained minimum

Chapter 1

cost tree for non-delay-constraint and delay-constraint scenarios respectively.

2. Study the minimum cost tree rearrangement problem in dynamic membership environment, and develop an efficient heuristic algorithm to perform proactive rearrangement in delay-constraint multicast Internet with dynamic membership support.
3. Study the MC and WCC placement problem, and develop optimal heuristic algorithms to place both MC and WCC nodes to limit their numbers in the optical Internet without comprising blocking probability. The proposed heuristics must be flexible enough to achieve optimal placement for all kinds of physical-constraint (in terms of numbers of MC and WCC node allowed) configurations without degrading the blocking performance. This is made possible by considering the impact of MC nodes placement on WCC placement and vice versa.
4. Study the multicast routing and wavelength assignment problem in delay constrained WDM multicast networks, and develop heuristic algorithms to perform both multicast routing and wavelength assignment concurrently for the first time to achieve even better performance efficiency than the existing algorithms which consider the two processes independently.

Chapter 1

Recently, overlay multicast [FK07] is intensively studied as a possible replacement for the conventional IP Multicast. This is motivated by easier implementation and lack of multicast deployment in the public Internet. However, this comes at a cost of much lower bandwidth utilization efficiency which will inflict a much higher cost for larger scale multicast applications in the Internet. With the likely increase of multicast deployment in the Internet in the near future, more protocols which facilitate IP Multicast will be developed, leading to easier implementation. With easier implementation and better bandwidth utilization efficiency, IP Multicast will be a plausible solution for cost-efficient multicast Internet in the near future. Hence, this thesis focuses in the conventional IP Multicast for both without and with optical support (IP over WDM) at the physical backbone of the network.

1.4 Contributions of the Thesis

The contributions of this thesis are as follow:-

1. Efficient minimum cost tree construction heuristic algorithms are proposed in this thesis for non-delay constrained and delay constrained multi-groups multicast Internet. The algorithms make use of shared-tree strategy to construct minimum cost trees and delay constrained minimum cost trees from this shared-tree for different multicast sessions in non-delay-constraint and delay-constraint scenarios respectively. Since only one-time computation of the shared-tree is required, the construction time is drastically reduced. Results show that beside reducing the complexity and construction timing, the proposed

Chapter 1

algorithms are able to construct near optimal cost trees from the shared-tree. Although the shared-tree concept is widely used for unicast communications, the idea of using single shared-tree for multiple trees construction for all multicast groups has not been studied before. This thesis presents the proposed heuristic algorithms using shared-tree concept for multiple trees construction in multi-groups multicast Internet and show that the shared-tree concept can also be used for multicasting problem.

2. The minimum cost tree rearrangement problem for dynamic membership multicast is studied and a more efficient heuristic algorithm adopting the proactive concept is proposed in this thesis. The results show that besides constructing minimum cost trees and reducing the rearrangement timing drastically, the side-effect of proactive strategy such as redundant provisioning is also drastically reduced. In addition, the proposed algorithm also allows the total number of disruption from the rearrangement to be controlled. Such control is not found in the existing reactive algorithms and this advantage will enable the rearrangement to be carried out for maximum cost benefits with all the given requirements satisfied.
3. The placement problem for MC and WCC nodes is studied to limit their numbers. A mathematical model based on relaxation method is derived to compute the blocking probability of a multicast session in an optical Internet with limited MC and WCC nodes (i.e. sparse splitting and

Chapter 1

sparse wavelength conversion). Such model which considers both sparse splitting and wavelength conversion together is the first so far as existing models only consider them separately and independently. Traffic-dependent optimal placement algorithms based on the “removal” method are devised and the results show that they can optimally place a number of MC and WCC nodes to achieve near optimal-cost performance, and are much faster than the existing “add” method for placing MC nodes.

4. The multicast routing and wavelength assignment problem in delay constrained optical multicast Internet is studied, and heuristic algorithms which perform both multicast routing and wavelength assignment concurrently for the first time are developed. The results show that our proposed algorithms are able to achieve better performance efficiency in terms of number of wavelength conversions, blocking probability and delays between source and receiver node compared to the existing algorithms which consider the two processes separately. The proposed algorithms are also extended for the limited MC and WCC nodes environment to show the flexibility of the proposed algorithms in any environment.

Each of the studies is carried out with both mathematical and simulation analysis. Simulation was carried out on a Pentium IV PC running Windows XP. The simulator program is programmed with C++ [Str00] and Matlab [HH00] codings. The main reason why C++ is used for the simulation programming is

Chapter 1

because of its flexibility to program all kinds of network simulation scenarios. Due to this reason, existing commercial software like Matlab, OPNET and even NS2 use C codes in their programs. The main advantage of these software packages is that they have ready-made functions which can be used immediately for simulations mostly for unicast or wireless communications. Unfortunately, due to the complexity of my proposed algorithms, and the complexity of tree constructions in multicasting, it is more viable to use C++ programming for the simulators. Thus, the existing commercial software was not used in the simulations for this thesis. This is consistent with most of the existing high-profile related studies where simulations programmed by C++ codes are considered sufficient and used to obtain results with reasonable accuracy. One of the main objectives of my proposed algorithms is to be computationally less intensive and efficient in processing. Thus, the simulation time based on my simulators is not a major problem. In my simulations, the average simulation time for a 100 nodes network with 10000 multicast sessions for a proposed algorithm is only at most a couple of minutes.

It is fair to judge how realistic an algorithm can be by using real experiment. The real experiment can be straight-forward for application layer multicast algorithms where I can just perform the real experiment in the Internet using a number of computers in the laboratory. However, switches and routers are involved for all my proposed multicasting algorithms in my studies. Thus, a testbed with OXC switches and multicast routers will need to be set up first. This will require lots of resources in terms of cost and time, which is not really possible at this stage. In addition, the simulation parameters, networks and

Chapter 1

scenarios used in this thesis are closely related to those commonly used by high-profile studies in [SSM06], [Ali02], [CLT06], [LMZ04] and [RMM99] just to name a few. Also, the objective of the simulation results in this thesis is to compare my proposed algorithms with the existing ones. By using simulations or real experiments, the proposed algorithms will be assessed based on the same inputs and conditions as the existing algorithms, and thus, the difference in results between the algorithms will be fairly consistent regardless of whether simulations or real experiments are conducted. Although an algorithm should realistically have mathematical, simulation and real experiment results to exhibit its efficiency for the real-world scenarios, the real experiment requires too much complexity, cost and time; and since the mathematical and simulation results are sufficient to show the difference in results between the compared algorithms, almost all the related studies in the literature till date according to my knowledge provide only mathematical and simulation results. Hence, the mathematical and simulation results achieved in the thesis which exhibits the advantages of my proposed algorithms over the existing ones should also be fairly consistent with real experimental results.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows. Literature reviews on the development of performance efficient trees in multicast Internet are presented in Chapter 2. Chapter 3 presents the efficient heuristic algorithms for constructing minimum cost tree and delay constrained minimum cost tree in multi-groups Internet multicast for non-delay-constraint and delay-constraint scenarios respectively. Chapter 4 presents the proposed proactive tree

Chapter 1

rearrangement strategy for delay-constraint multicast with dynamic membership support in the Internet. Chapter 5 studies the placement problem of MC and WCC nodes for multicast in an optical Internet, and presents the proposed optimal placement heuristic algorithms. Chapter 6 presents algorithms to perform both multicast routing and wavelength assignment concurrently in optical multicast Internet. Finally, the thesis is concluded with recommendation for future work in Chapter 7.

Chapter 2

Literature Review

Research on the performance of efficient trees for multicasting in the Internet has been ongoing for years and will continue to be a vital research topic for many years to come due to the increasing usage of multicast applications over the Internet. In this chapter, a brief recap of Internet multicast is first presented; followed by comprehensive literature review on existing related works which form the background for the research work of this thesis.

2.1 Overview of Internet Multicast

The Internet is a public network comprises of many sub-networks supported by various service providers. Internet multicast allows data to be sent efficiently from a single node to a group of receiver (destination) nodes in the Internet via a delivery tree. A delivery tree is a non-looping tree that can be used to transmit the traffic from the source to the receiver nodes across the network. In Internet multicast, construction of the tree to deliver traffic is dependent on the kind of routing protocols adopted for each application. Packet-based multicast data traffic is growing rapidly in today's telecommunication and has overtaken circuit-switched traffic. Hence, packet-switching is preferred over circuit-switching for many reasons with efficient bandwidth utilization as one of them. In packet-switching, data can either be routed in the form of a datagram (connectionless oriented) where routing tables are maintained at each node and

Chapter 2

each packet carries sufficient information to reach the receiver, or in the form of virtual circuits (connection oriented) where paths are provisioned with bandwidth to send packets to their receiver nodes without having the packets carrying too much routing overheads. For the connectionless oriented platform, multicast data packets are routed by means of the routing table at each node and each data must have sufficient information to find its destination. Hence, the overheads required in such platform are very important. For connection oriented platform, paths are already provisioned and set up from the source to the receiver nodes, and packets are transmitted via these paths without carrying too much routing information. Thus, the transmission cost in this case is more relevant than overheads. As virtual circuits allow QoS to be provided via traffic engineering and are more suitable for higher level and more reliable services, this thesis focuses on Internet multicast employing packet-switching via virtual circuits. For the subsequent sections, we shall review some existing algorithms that can be used to construct performance efficient trees for virtual-circuited packet-switching multicast.

First, the minimum cost tree construction for multicast is studied and its existing algorithms reviewed. After which, the minimum cost tree construction for dynamic membership multicast is studied and its existing algorithms reviewed. Next, the placement problem of multicast and wavelength conversion nodes in transparent optical multicast networks is studied and its existing algorithms reviewed. Finally, the multicast routing and wavelength assignment (MC-RWA) problem is studied and its existing algorithms reviewed.

2.2 Minimum Cost Tree Construction in Multicast

2.2.1 Minimum Cost Tree Multicast

In minimum cost tree (MCT) multicast, minimum cost delivery trees are constructed to deliver traffic from the source node to the receiver nodes. It is also known as minimum weight tree multicast when a weight is given to each link to represent link cost, link bandwidth or link delay and the objective is to minimize the total link weight inside the tree. For simplicity, we shall use the more popular link cost metric as the weight on each link in our studies although it should be noted that any other metrics can be selected instead, and that all the existing and proposed heuristics to minimize the overall link cost in the constructed tree are applicable to other metrics being used as the weight for each link.

It was established in [Win87] and [HR92] that constructing an optimal minimum cost tree for static membership multicast group can be modeled as the Steiner tree problem, which is NP-complete. Hence, heuristics need to be developed to construct near-optimal minimum cost tree within a much faster period. In this section, we shall first introduce the well known minimum spanning tree (MST) algorithms such as the Prim's and Kruskal's algorithm, and the popular shortest path (in terms of cost) algorithm such as the Dijkstra's algorithm as they are used by some of the existing minimum cost tree heuristics to construct the delivery trees. A MST constructed from a graph that makes up of the entire or part of the network is the minimum cost or weight tree which spans from the source node to all other nodes in the graph. After introducing the MST algorithms, the existing minimum cost tree heuristics for both delay and

Chapter 2

non-delay constrained are reviewed. The heuristics are grouped into single tree construction where each minimum cost tree is constructed from scratch or shared tree construction where each minimum cost tree is constructed from a pre-constructed shared tree.

2.2.2 Minimum Spanning Tree Algorithms

Prim's algorithm [Prim57] is a well known greedy algorithm used to compute a MST from a graph G . We shall refer to the tree constructed so far during each step of Prim's algorithm as active tree and the completed tree as the final tree. For each step of Prim's algorithm, a node will be selected to connect to the active tree. The selection of such node depends on the link cost (weight) between the active tree and those unconnected nodes adjacent to the active tree. The unconnected node adjacent to the active tree and has the least link cost (weight) to the active tree is selected to be connected, and this process continues until all the nodes are connected to obtain the final tree. The initial starting node in Prim's algorithm is selected arbitrarily. Prim's algorithm runs in $O(n \cdot m)$ in its most native implementation and $O(n^2)$ for searching at most n edges for each step. The time complexity is improved to $O(m \cdot \log n)$ if binary heaps in conjunction with adjacency lists are used and $O(m + n \cdot \log n)$ if Fibonacci heaps [FT87] in conjunction with adjacency lists are used.

Kruskal's algorithm [Kru56] is another well known greedy algorithm that can be used to compute MST. For each step of Kruskal's algorithm, the minimum cost link in graph G which has not been selected to form any trees is selected to connect to any active trees computed. If the selected link has its ingress and

Chapter 2

egress nodes belonging to a single active tree, a cycle will be created. Hence, such a link will not be considered again for selection to form any trees. If the ingress and egress nodes of the selected link belong to two different active trees, the two active trees will be connected by means of this link. This continues until all the nodes in the network are connected to a single tree. The time complexity of Kruskal's algorithm is the time complexity to sort all the links in the graph in increasing link cost order which is expressed as $O(m \cdot \log m)$.

2.2.3 Shortest Path Algorithm

Dijkstra's algorithm [Dijk59] is a well known algorithm that can be used to construct the shortest path (in terms of cost) between any two nodes in the network. To compute the shortest path between node s to all other nodes $u \in V$ where V is the set of all vertices in the graph G , a shortest path tree (SPT) has to be computed using Dijkstra's algorithm. To compute the SPT from node s to all other nodes $u \in V$, Dijkstra's algorithm first has to have two set of vertices. Let V' be the set of vertices whose shortest path from s has been determined. Hence, $V - V'$ is the set of remaining vertices yet to have their shortest paths from s determined. Each node u in V' has two parameters. They are the shortest path from s to u given by $P_{s,u}$ and its cost, $C(P_{s,u})$. To determine the shortest path from s to the next node $v \in V - V'$, the sum of $C(P_{s,u})$ and link cost from u to v , expressed as $C(u,v)$, is considered. Corresponding to each $C(P_{s,u}) + C(u,v)$ is a path $P_{s,v} = P_{s,u} \cup e_{u,v}$. The path with the minimum $C(P_{s,u}) + C(u,v)$ will then be selected as the shortest path from s to v and

Chapter 2

$V' = v \cup V$. This is repeated until $V - V' = \emptyset$. Hence, the shortest paths from s to all nodes in $V - s$ are computed in the form of a SPT.

2.2.4 Minimum Cost Tree Algorithms using Single Tree Construction

The heuristic proposed in [KMB81] is a classical heuristic solution for the minimum cost Steiner tree problem which is NP-complete. It can be remodeled as the minimum cost multicast tree problem by treating the source s and receiver nodes $Dest = \{d_1, \dots, d_x\}$ as the Steiner points S (i.e. $S = s \cup Dest$), and the objective is to construct a non-directional tree spanning to all the Steiner points so that finally a directional tree can be constructed from the Steiner tree by directing the traffic from the source to the receiver nodes via the Steiner tree edges. We shall refer to the heuristic as KMB for convenience. KMB is used widely even in the recent papers as the benchmark for constructing trees with cost near to the optimal minimum. To obtain the minimum cost Steiner tree in KMB, the distance graph G' consisting of only the Steiner points is first computed from the original graph $G = (V, E)$ where V and E are the set of nodes and edges in the graph respectively. A distance graph is a graph in which each edge between two nodes is mapped to the corresponding minimum cost path between the two nodes computed from the original graph using some well known algorithms such as Dijkstra's algorithm. After the distance graph is computed, a minimum spanning tree T' is constructed from the distance graph. The subgraph of G given by G_s is then computed by replacing each edge in T' to the corresponding minimum cost path in G . The minimum spanning tree from G_s given by T_s is then computed. Finally, the Steiner tree T_H is obtained

Chapter 2

by deleting edges in T_s such that all the leaves are the Steiner points. To obtain the minimum cost tree from the Steiner tree, the source node can be set as the starting point to direct the traffic to other Steiner points which are the receiver nodes since $S = s \cup Dest$ via the Steiner tree edges. It has been shown in [KMB81] that KMB has a worst time complexity of $O(|S||V|^2)$ and the tree cost obtained is no more than $2(1 - \frac{1}{L})$ times that of the optimal minimum cost tree where L is the number of leaves in the optimal minimum cost tree. KMB is applicable only to non-delay constrained multicast. A heuristic modified from KMB, known as CKMB, was proposed in [SL98] for constructing the minimum cost tree in delay constrained multicast. Basically KMB is modified such that the constructed tree is delay bounded.

A centralized approach to construct the minimum cost tree in delay constrained multicast known as CMCT was proposed in [KPP93a]. In CMCT, a closure graph consisting of all source and receiver nodes and constrained minimum cost edge between any two of them is first computed. This is somewhat similar to the approach in [KMB81] except that each edge between two nodes in the closure graph in CMCT corresponds to the delay constrained minimum cost path between the two nodes in the original graph. The delay constrained minimum cost path between two nodes is computed using the dynamic programming approach similar to the shortest path algorithm proposed in [Flo62]. After computing the closure graph, a minimum spanning tree rooted at the source node and spanning to all other nodes in the close graph, which are the receiver nodes, is to be constructed. In [KPP93a], two selection functions

Chapter 2

f_C and f_{CD} are proposed to select the edge to connect between the active tree and the unconnected node during each iteration of the minimum spanning tree construction. CMCT is known as CMCT_C if f_C is used for the selection function and CMCT_{CD} if f_{CD} is used for the selection function. For CMCT_C, the delay constrained minimum cost edge adjacent to the active tree in the closure graph is selected to connect the next unconnected node. The selection function of an edge $e = (u, v)$ in CMCT_C is given by $f_C(u, v)$

$$= \begin{cases} C(u, v), & \text{if } D(P_T(u)) + D(u, v) < \Delta \\ \infty, & \text{if otherwise} \end{cases} \quad \text{where } C(u, v), D(u, v), D(P_T(u)) \text{ and } \Delta$$

represent the cost of edge e , delay of edge e , delay of active tree path from source to v and the delay constraint respectively. Unlike CMCT_C where only edge cost is considered, CMCT_{CD} considers both the cost and delay of each edge adjacent to the active tree. The selection function of an edge $e = (u, v)$ in CMCT_{CD} is given by,

$$f_{CD}(u, v) = \begin{cases} \frac{C(u, v)}{\Delta - D(P_T(u)) - D(u, v)}, & \text{if } D(P_T(u)) + D(u, v) < \Delta \\ \infty, & \text{if otherwise} \end{cases}.$$

Hence, it can be easily observed that CMCT_{CD} is able to connect more nodes in the event of stricter delay-constraint compared to CMCT_C because it pays attention to the delay residue when selecting the next edge to connect new nodes during tree construction. However, this is achieved at the expense of less optimal cost compared to CMCT_C since only the cost component is considered during tree construction in CMCT_C; naturally it should achieve better cost performance. After the minimum spanning tree is computed from the closure graph, each edge in the spanning tree is replaced with the corresponding delay

Chapter 2

constrained minimum cost path in the original graph. Pruning is then performed to ensure that all the leaves are receiver or source nodes

It has been established in [KPP93b] that the centralized approach proposed in [KPP93a] required the broadcasting of network information to maintain the consistent view of the network status and while this process is possible for small networks, it is less reliable, less accurate and more expensive for larger networks. Hence, a distributed algorithm known as DMCT was proposed in [KPP93b]. The two selection functions f_c and f_{cd} similar to that in CMCT were proposed to select the next delay constrained edge to connect to the active tree during each step of the tree construction process. The difference between DMCT and CMCT is that no closure graph is constructed. The tree in DMCT is constructed by considering the next suitable adjacent edge to the active tree for connection using either f_c or f_{cd} until all the receiver nodes are connected in the tree. Obviously, because DMCT connects the next suitable delay constrained adjacent edge instead of next suitable delay constrained adjacent minimum cost path to the active tree, CMCT will yield better cost performance than DMCT but DMCT is more scalable to larger networks. Hence, CMCT is used as the benchmark for comparison with DMCT in [KPP93b].

Another centralized algorithm for constructing delay constrained minimum cost tree is BSMA proposed in [PZG98]. BSMA consists of two phases. The first phase is to construct an initial tree with minimum delay between the source and each receiver node (i.e. minimum delay tree) using Dijkstra's shortest path algorithm. The second phase is to iteratively minimize the cost of the tree

Chapter 2

obtained in the first phase while keeping it bounded. For the second phase, the higher cost relay paths known as “Superpaths” are replaced with paths of lower costs iteratively until no more paths need to be replaced. The replacement paths are constructed using the *k-shortest path algorithm* [Yen71]. Each relay path is characterized by two properties. The first property is that all nodes within the relay path must not be the source or receiver nodes and must be of exactly degree two (relay node). The second property is that the ingress and egress node of the relay path must either be the source or a receiver node. The strategy of BSMA is to ensure that a delay bounded (constrained) tree is possible first before trying to tweaking the relay paths to minimize the cost of the delay bounded tree.

In contrast to BSMA, a centralized approach, known as SLIM, is proposed in [AZ01] to construct a delay constrained minimum cost tree using a set of minimum cost paths (MCPs) and minimum delay paths (MDPs). The delay constrained path with the least cost among the MCPs and MDPs will be selected to connect the next receiver node to the active tree during each step of tree construction in SLIM until all receiver nodes are connected via a tree. In other words, the least cost MCP which can satisfy the delay-constraint is selected for connecting the next receiver node while the next least cost MCP or MDP will be selected in cases where the least cost MCP cannot satisfy the delay constraint. It was stated in [AZ01] that in cases where MDP is selected to connect a node, the path may not be optimal. Hence, SLIM further performs the cost improvement process on the constructed tree similar to that in BSMA by iteratively replace the higher cost relay paths with lower cost relay paths.

Chapter 2

However, the pool of candidate paths for replacement consists of only the MCPs and MDPs; hence, SLIM will not be able to find an optimal cost tree if a non-optimal MDP is selected. Thus, K-SLIM was proposed to improve SLIM by computing delay constrained minimum cost paths using the *k-shortest path algorithm* and include them in the selection pool for replacing the higher cost relay paths. Since K-SLIM constructs a better optimized initial tree at the beginning by considering delay constrained least cost MCPs first, it will take much lesser cost optimization steps than BSMA where a minimum delay initial tree is constructed.

For the centralized algorithms reviewed so far, although they are able to compute near cost optimized delay constrained trees, they did it at the expenses of higher computational complexity and hence incur longer computational time. The STAR algorithm proposed in [LMZ04] is another centralized algorithm aims at constructing near cost optimized delay constrained trees in multicast networks. However, unlike previous centralized approach algorithms reviewed, STAR aims to reduce the complexity by computing only the necessary paths for connecting receiver nodes to obtain the minimum cost delay constrained tree. STAR basically involves 3 steps. The first step is to compute the delay constrained (bounded) minimum cost path (DCMCP) originated from all the receiver nodes using the same concept as Dijkstra's algorithm. The second step is to construct the initial tree by connecting the next shortest delay constrained receiver node to the active tree by selecting the least cost DCMCP until all the receiver nodes are connected by the tree. The final step is to improve the tree cost by rearranging the sub-tree rooted at each trim node by means of the

Chapter 2

DCMCP. Trim nodes are discovered by doing a search in the tree starting from the source node to all the leaf nodes and placed in a stack before any improvement processes start. A trim node is the child node of either the source node or a node with node degree greater than 2. The last trim node to be placed in the stack will be popped out and considered first followed by the next in the stack. The new sub-tree will be retained only if the improvement in cost exceeds a certain level, and when that happens, a new set of trim nodes from the new tree will be computed and the whole process will start all over again. If the cost improvement is below the specified level, the next trim node will be considered. The entire improvement step stops when no more sub-trees can be improved. The main differences between STAR and previous centralized algorithms is that STAR reduces tree cost by optimizing sub-trees instead of relay paths, and that it only computes the necessary paths instead of all paths between nodes. Due to these differences, STAR was shown to be able to compute lower minimum cost delay constrained trees with a much faster computational time and using much less space to store path information compared to CMCT, BSMA, SLIM and K-SLIM.

2.2.5 Minimum Cost Tree Algorithm using Shared-Tree Construction

So far, all the reviewed algorithms for minimum cost tree construction are based on single tree construction. This means that for each incoming multicast session request, the delivery tree has to be constructed from scratch. Although the single tree construction technique will most likely construct near optimal cost trees, the computational complexity is extremely high. This is more so for those algorithms which involve cost reduction phase where minimum cost paths

Chapter 2

have to be reconstructed to replace the higher cost paths to improve the existing tree cost. Hence, another technique known as the shared-tree construction is introduced to reduce the complexity and overheads involved at the expense of less cost optimal trees.

The core-based tree (CBT) algorithm proposed in [BFC93] is a classic algorithm which employs a single shared tree to construct the delivery tree for all multicast sessions originated from the same multicast groups. A multicast group is usually assigned to a multicast service where multicasting takes place between the nodes inside the group. Requests can be made to send traffic from any node to the rest in the group. Hence, each of the requests is represented as a multicast session with a source and receiver nodes pair. For single tree construction or Steiner tree technique, multicast sessions need not be grouped according to the multicast group they originated from, because each request is uniquely handled by the algorithms. For shared-tree construction, multicast sessions from the same multicast group are grouped and uniquely handled. Hence, a single shared-tree for each group is constructed to handle all the multicast sessions from the group for the shared-tree construction technique; while a single tree is constructed from scratch for each multicast session for the single tree construction technique. To construct the shared-tree for a group in CBT, a core node is first selected. Members of the group are then connected to the core node via minimum cost paths one by one until all are connected. When an incoming multicast session from the group arrives, the shared tree will be used to direct the traffic from the source to the receiver nodes. In terms of overheads and computational time, CBT performs better than single tree

Chapter 2

construction algorithms; but in terms of cost performance, CBT is inferior to the single tree construction algorithms. Recently, [WQY06] studied the feasibility of employing CBT in WDM multicast networks and focused on the sub-problem of routing data from sources to the core based on the argument that the other sub-problem of routing data from core to the receivers is strictly a multicast routing problem. This shows that CBT is useful for cases in which overheads and computational time are important considerations.

2.3 Minimum Cost Tree Construction in Dynamic Membership Multicast

2.3.1 Dynamic Membership Multicast

As mentioned in chapter 1, multicast membership can be dynamic to allow member nodes of a multicast group to join or leave the group. Hence, the list of receiver nodes of a multicast session from the multicast group is not fixed and may change over time. Each request for a member node to join or leave a group is referred to as an update. Dynamic membership multicast is also known as on-line multicast. Due to the dynamic nature of the multicast applications over the Internet in recent years, it is imperative that any performance efficient multicast algorithms should support dynamic membership multicast. The performance of a multicast algorithm is determined by how well the tree is constructed for the delivery of the traffic. In order to achieve good performance, the delivery tree has to be constructed in such a way that cost or blocking probability is minimal for each multicast session's update. This can be easily achieved for static membership multicast. Several well-known heuristics are proposed in [KMB81], [Ram96] and [YY05].

Chapter 2

On the other hand, to construct a performance efficient multicast tree for dynamic membership multicast is not as straight forward. This is because the membership of a multicast group can be changed and in order to obtain a performance efficient tree after each update, the tree has to be rearranged for the updated membership. In other words, performance efficient tree construction in dynamic membership multicast can be viewed as multiple performance efficient trees construction in static membership multicast. Such rearrangements will affect the services to the remaining members of the multicast group. Hence, member's service disruption duration, numbers of disrupted members, and performance have to be balanced in order to achieve the desirable services economically in dynamic membership multicast. As mentioned earlier, performance efficiency refers to achieving near-optimal results for one or more performance metrics, such as cost, reliability or algorithm complexity with or without some constraints imposed. This is in line with the real-world practices whereby there are usually more than one performance metrics to achieve near-optimal results for the desirable services. For example, multicast cost and the number of disrupted members must be kept reasonable low because too high a cost will cause scalability issue, and too many members disrupted will cause service quality issues. The near-optimal results from existing well known optimal heuristics are usually used as the benchmark to determine if a particular heuristic is able to achieve near optimal result for each metric. As each member who joins or leaves a multicast group cannot be the source node of an ongoing multicast session from the group and can only be a receiver node of the ongoing multicast session, the joining and leaving of a member for a multicast group is the same as the joining and leaving

Chapter 2

of a receiver node for the multicast session of the multicast group. Hence, the term “member” and “receiver” can be used interchangeably unless stated otherwise. Since we are concerned with the cost of the tree for each multicast session, to avoid any confusion, we shall refer to the dynamic membership in the form of joining and leaving of receiver nodes in the context of a multicast session. However, readers should note that this is only possible if there are joining and leaving of member nodes from the multicast group from which the multicast session originates.

2.3.2 Performance Metrics used for Dynamic Membership Multicast

In order to support dynamic membership multicast economically and efficiently, the cost of each tree, which is the summation of the cost of all links in the tree, has to be considered. As mentioned in chapter 1, link cost is defined as the transmission cost in a single link and can be a function of lease cost, fibers used, maintenance cost or other related cost. Beside tree cost, duration at which a member's service is disrupted due to the rearrangement of each tree must not take too long as it will affect the quality of service. In addition, the total number of members which have their services disrupted by the rearrangement must be kept small in order to achieve the desired performance. To prevent any race conditions where the rearrangement of a tree for an update is yet to finish before the next update arrives, the rearrangement duration must be kept as low as possible. Hence, it is easy to see that for non-rearrangement algorithms, tree cost is the only metric to be minimized whereas tree cost, member's disruption duration, number of members disrupted and

Chapter 2

rearrangement duration have to be kept minimal for the desirable dynamic membership supported multicast services.

2.3.3 Algorithms for Minimum Cost Tree Construction in Dynamic Membership Multicast

Algorithms for MCT construction in dynamic membership multicast are divided into non-rearrangeable and rearrangeable algorithms. For non-rearrangeable algorithms, delivery tree for each update of a multicast session request will remain the same, while rearrangeable algorithms will rearrange the tree for each update. As mentioned earlier, rearrangement is necessary to ensure that the performance of the tree for each update is not degraded. Hence, reviews in this chapter will focus more on rearrangeable algorithms. However, non-rearrangeable algorithms will also be reviewed briefly for completeness of this review.

2.3.3.1 Non-Rearrangeable Algorithms

The development of performance efficient trees in non-delay constrained dynamic membership multicast was first introduced in [Wax88] using the non-rearrangeable greedy approach with the aim to minimize any disruptions to the existing members (receivers). In [Wax88], a greedy heuristic was proposed to consider all possible minimum cost paths to connect the joining member to the existing tree. The least minimum cost path will then be selected to connect the new member. If the new member is already a node in the existing tree, only the status of that node will be changed to be a receiver while the structure of the tree remains unchanged. For a member leaving the multicast request, the

Chapter 2

redundant path to the leaving member will be pruned if the leaving member is a leaf-node. The ingress node of the redundant path has to be part of the existing tree used to deliver traffic to other member nodes. Hence, the ingress node is either a source node or a node with out-degree greater than 1. If the leaving member is a non-leaf node, only the status of the node will be changed to be a non-receiver while the structure of the tree remains unchanged because the node is still needed to deliver traffic to the rest of the receivers. The greedy heuristic in [Wax88] is simple and straight-forward. However, it does not consider tree rearrangement when an existing tree node is added as member or when a non-leaf member node leaves the tree. This will most likely degrade the performance of the tree because the tree structure is not re-constructed based on the updated set of receivers. Such performance degradation increases with the increase in updates over time.

Another non-rearrangeable algorithm for non-delay constrained dynamic membership multicast, known as Virtual Trunk Dynamic Multicast (VTDM), was proposed in [LL98]. For VTDM, a virtual trunk (VT) which is a tree in the underlying network is first computed. Then, to add a member node which is a non-existing tree node, the least minimum cost path between the added member node to the VT will be selected to connect the added member node. This VT is computed using the algorithm in [KMB81] to connect the source node to a selected set of nodes. The set of nodes is selected by selecting the first k most weighted nodes. Each node is weighted according to the number of shortest paths between any two nodes passing through it. Hence, the concept of VTDM is to make use of frequently used nodes and links in the optimal tree

Chapter 2

construction in static multicast algorithm to connect new member nodes which are not part of the existing tree so that low-cost trees matching the cost-performance of the static multicast algorithm can be constructed. For the addition of existing tree nodes as member nodes and deletion of leaf and non-leaf member nodes, VTDM handles them in the same way as described in [Wax88]. Since VTDM is a non-rearrangeable algorithm, it has the same weakness as the greedy heuristic in [Wax88].

In contrast to [Wax88] and [LL98], a delay-constrained environment was considered in [Jia98]. For the construction of each static multicast tree in [Jia98], a table showing the cost between the existing tree and the receivers, and a local table for each existing tree node showing the cost to the rest of the unconnected receivers are created. Using the source node as the starting node in the existing tree, the receiver node which can be connected to the existing tree using the least cost delay constrained path will be selected and appended to the tree, and this process continues until all receivers are added to the tree. The algorithm in [Jia98] is then extended for dynamic membership support by applying the same technique for adding a non-existing tree node as member node. The deletion of non-leaf and leaf member nodes; and the addition of existing tree nodes as member nodes are handled in the same way as in [Wax88] and [LL98]. As [Jia98] is a non-rearrangeable algorithm, it has the same weaknesses as those algorithms proposed in [Wax88] and [LL98].

Another non-rearrangeable heuristic for delay-constrained dynamic membership multicast, known as Dynamic Delay Bounded Multicast Algorithm

Chapter 2

(DDBMA), was proposed in [HZ98b]. DDBMA first constructs a minimum delay Steiner tree using Dijkstra's shortest path algorithm with respect to the multicast's source node. It then performs refinement iteratively on the current tree for lower cost until no further refinement is possible. The refinement is carried out by replacing the superpath with the delay constrained shortest path. A superpath is the longest path in the current tree whereby all its nodes other than the two end nodes are relay nodes which connect two tree edges. By removing the superpath, disjoint sub-trees are created and the objective is to join the sub-trees with the delay constrained shortest path, which is the least cost path with the tree satisfying the delay-constraint condition. The delay constrained shortest path is computed using Dijkstra's shortest path algorithm. If the first shortest path found is not delay bounded, the m^{th} path found to be delay bounded will be selected to join the sub-trees. For adding a non-existing tree node as member node, the delay constrained shortest path computed from Dijkstra's shortest path algorithm is selected to connect the new member node. DDBMA uses the same technique as described in [Wax88] and [LL98] for adding an existing tree node as a new member, and deleting a leaf or non-leaf member node. Thus, DDBMA has the same weakness as those algorithms proposed in [Wax88] and [LL98].

2.3.3.2 Rearrangeable Algorithms

The development of performance efficient trees in non-delay constrained dynamic membership multicast using rearrangeable algorithm was first introduced in [IW91]. In [IW91], a heuristic known as *Edge-Bounded Algorithm (EBA)* was proposed to perform rearrangement if the distance

Chapter 2

between nodes in the tree after each membership change exceeds a set bound δ .

EBA worked on the basis of distance graph where each edge between two nodes in the graph is the shortest path between the two nodes. Rearrangement in EBA is partial in such a way that when a new member node is added, the least-cost edge in the distance graph $\hat{G}$ corresponding to network graph G between the new member node v and the existing tree node u is first selected to be appended to the existing tree T and such tree is defined as the extended tree T' . The corresponding distance graph for T' is $\hat{T}'$. The maximum-cost edge among the paths between v and the rest of the nodes in T' is selected and listed as $\text{edge}(u, v)$. This maximum-cost edge is then compared with the corresponding least-cost edge between u and v in the distance graph, and if the cost difference is more than δ times, the former edge between u and v will be replaced by the edge between u and v in the distance graph. This ensures that the cost of the rearranged tree is bounded by 4δ times the cost of a Steiner tree found by pruning a minimum spanning tree of the graph G . From the algorithm of EBA, it is clearly seen that there will be no change if the added member node is an existing tree node. For the deletion of member nodes, EBA adopted the same handling as the greedy heuristic proposed in [Wax88]. Although the trees produced by EBA are at most 4δ times that of the optimal level, this is only for the case in which updates are about adding non-existing tree nodes as member nodes. No rearrangement is proposed for adding existing tree nodes as member nodes or for deletion of leaf and non-leaf member nodes. Hence, EBA will inherit the same weaknesses exhibited by the non-rearrangeable algorithms where member nodes deletion and existing tree nodes added as members are concerned.

Chapter 2

Another rearrangeable heuristic known as Geographic Spread Dynamic Multicast (GSDM) was proposed in [KK95]. GSDM came from the idea of geographic spread. If a non-existing tree node A is added as a member node, three key nodes are to be involved in the attachment of A to the existing tree. First node B is the nearest node in the existing tree to A . The other two nodes are C and D which are the two nearest nodes to A and neighbors on either side of B in the existing tree. GSDM then compares the cost of four possible configurations of connections involving A, B, C and D , and select the one with the least cost to connect the new member node. The connection which will result in greater geographic spread will be preferred if there is a tie in the selection. Again, GSDM does not consider any rearrangement when an existing tree node is added as new member or when leaf and non-leaf member nodes are deleted. GSDM handles existing tree node addition as member and deletion of non-leaf and leaf member nodes in the same way as EBA. Hence, the same weaknesses in EBA are inherited in GSDM.

So far, the aforementioned rearrangeable algorithms only perform partial rearrangement when non-existing tree nodes are added as member nodes, and do nothing when existing tree nodes are added as new members or when leaf and non-leaf member nodes are deleted. These algorithms will inherit the same weakness as those found in non-rearrangeable algorithms. To overcome this, ARIES was proposed in [BV97] to construct better cost trees in non-delay constrained dynamic membership multicast. Unlike EBA in [IW91] and GSDM in [KK95], ARIES performs rearrangement for a region of the current tree if and only if the accumulated damage of that region due to a number of updates

Chapter 2

exceeds a specific threshold. In this way, the rearrangement is triggered by the damages done by all cases of member joining and leaving updates, and should yield a better cost tree than EBA and GSDM since full-rearrangement is carried out in ARIES when triggered. The damage of each region is measured by the number of modified (former or new member) nodes in the region. Although ARIES performs full-rearrangement and hence does not inherit the weaknesses of non-rearrangeable and partial rearrangeable algorithms, the rearrangement triggering function which takes into account only modified nodes in each region of the tree is not suffice for delay constrained multicast networks.

Unlike ARIES which is suitable only for non-delay-constrained multicast, the Lagrangian-Relaxation-Based Algorithm (LRA) proposed in [HLP98] can be used to construct delay bounded trees for delay-constrained dynamic membership multicast. LRA employs the Lagrangian-Relaxation method frequently used to solve NP problems to connect non-existing tree nodes as new members but function like those non-rearrangeable algorithms described in [Jia98] and [HZ98b] to keep the current tree unchanged for adding existing tree nodes as new members, deleting non-leaf member nodes, and pruning the redundant paths for deleting leaf member nodes. Hence, LRA also inherits the weaknesses found in [Jia98] and [HZ98b] although it employs a more cost-optimal method to connect new non-existing tree nodes as member nodes.

The concept employed in ARIES whereby rearrangement is triggered when the damage of a region in the current tree exceeds a threshold was further extended for delay-constrained multicast in [RMM99]. The heuristic proposed in

Chapter 2

[RMM99] is known as *Controlled Rearrangement for Constrained Dynamic Multicast* (CRCDM). In CRCDM, a new triggering function known as the *Quality Factor (QF)* was introduced to determine if each M-region of the current tree after updates require any rearrangement. M-region is defined as a region in the current tree which contains at least one M-node. An M-node is defined as a former non-leaf member node or the ingress node of a previously pruned path whose egress node is a leaving leaf-node member. It is the equivalent of a damage node in ARIES. The QF of an M-region R is given by the expression $QF(R) = \frac{|desc_T(R)|}{|Mnodes_T(R)|}$, where $desc_T(R)$ and $Mnodes_T(R)$ is the set of descendent receivers and set of M-nodes respectively of M-region R . QF measured the usefulness of a region in the current tree. The usefulness of a region decreases if the number of M-nodes in the region increases. The usefulness of a region will determine if the region requires any rearrangement. If the usefulness is high, no rearrangement should be carried out. To prevent excessive disruptions, the usefulness of a region is consider high if it transmits traffic to many receiver nodes which reside either within the region, or are descendents to the region, and should not be rearranged. M-node is defined as a former receiver node which has left the multicast group, but remains in the tree because it now acts as a relay node. Thus, if the number of M-nodes in the region increases, it means that a large number of receivers which were in the region have since been removed from the group leading to a decrease in the usefulness of the region. This means that there are an increased number of deleted member nodes in the region. Also, the usefulness of a region increases with the number of receivers receiving traffic via the region. Hence, it is

Chapter 2

obvious that a high QF region indicates a great utility region to the current tree. Thus, rearrangement for the M-region R occurred if $QF(R) < \rho$ where ρ is the threshold which indicates the minimum usefulness a region should have without any rearrangement. The check for rearrangement occurs after each update of member node addition or deletion.

For adding an existing-tree node as member, deleting non-leaf and leaf member nodes, CRCDM handles them in the same way as that of ARIES. This means the current tree remains the same and only the status of node changes, either to start receiving traffic for adding existing-tree node as member or to stop receiving traffic for deleting non-leaf member node. For leaf member node removal, the path between the removed member node and the nearest current tree node still relaying traffic for the remaining member nodes is pruned. For adding non-existing tree node as member, the process is more complex. First, the delay constrained minimum cost path DCMCP between two nodes will be computed. To reduce the total computation time, [RMM99] suggests only computing DCMCP between the new member node and the first κ existing tree nodes in the priority list. The existing tree nodes in the list are ranked according to their delay to the source with the smallest delay having the top rank. The idea is that a smaller delay node to the source will have greater delay residue to connect incoming new member nodes. After computing the DCMCPs, the selection function SF of each DCMCP is computed to determine which DCMCP is selected to connect the new member node. The selection

function of a DCMCP, P , is expressed as $SF(P) = \frac{C(P)}{\Delta - D_T(s, a) - D(P)}$ where

Chapter 2

$C(P)$ is the cost of P , Δ is the delay-constraint, $D(P)$ is the delay of P and $D_T(s, a)$ is the delay of the current tree path between the source node and the ingress node of P . The DCMCP with the minimum SF is then selected to connect the new member node. The aim is to minimize tree cost and maximize delay residue so that more member nodes can be connected in future updates. If the ingress node of the selected P is an M-node, the ingress node will change its status from M-node to any ordinary intermediate tree node since it is now useful again to relay traffic to the new member node. Hence the configuration of M-nodes and M-regions changes with the incoming updates and check needs to be performed to determine if any rearrangement is needed after each update.

During rearrangement, the M-region to be rearranged will first be removed from the current tree. Fragments of the current tree will be created due to the removal. Starting from the fragment with the source node, DCMCPs are again constructed to connect the fragments. Each fragment consists of a fragment leader whereby the delay between the leader and the rest of the nodes in the fragment is the maximum possible. This is to ensure that the DCMCP selected to connect each fragment at its leader node is able to deliver traffic from source node to each node in the fragment within the delay-constraint specified. [RMM99] claimed that CRCDM outperformed ARIES and LRA for dynamic membership multicast scenarios, and performed extremely well when compared to static Steiner heuristic for static membership multicast scenarios.

The concept in CRCDM was further modified in [LLL04] and [RM06]. In [LLL04], a heuristic known as *Constrained Rearrangement Multicast Routing*

Chapter 2

(CRMR) was proposed to include the cost change of each M-region in the triggering function for M-region's rearrangement. The triggering function for rearrangement in CRMR is given by the expression

$$TF(R) = A1 \frac{|desc_T(R)|}{|Mnodes_T(R)|} + A2 \frac{1}{CostC} \quad \text{where } A1 \text{ and } A2 \text{ are adjustable}$$

parameters approximated by numerical results for different density networks, and $CostC$ is the cost change in region R . $CostC$ is denoted as the sum of cost differences between current connection and optimal connection of each unique node pair in the region. Similar to CRCDM, rearrangement in CRMR occurs if and only if $TF(R) < \rho$ where ρ is the threshold which indicates the minimum usefulness a region should have without any rearrangement. [LLL04] also introduced a bandwidth indication function to improve the well-known static membership multicast constrained algorithm proposed in [KPP93a] for constructing the initial trees. It was shown in [LLL04] that CRMR achieved better cost performance than CRCDM proposed in [RMM99]. However, nothing was mentioned regarding the resulting members' disruption duration and the number of disrupted members produced for the better cost performance in CRMR.

In [RM06], a rearrangeable heuristic using a modified concept from CRCDM was proposed to solve the dynamic membership multicast problem in WDM networks although it is also applicable for any general networks. Similar to CRMR, the heuristic algorithm proposed in [RM06] takes into consideration the cost contribution of a particular M-region to determine if that M-region requires any rearrangement. The difference is that instead of including cost change of

Chapter 2

M-region in the triggering function, the heuristic algorithm proposed in [RM06] includes the cost contribution of the M-region directly with respect to the total cost of the tree. The cost contribution of an M-region R is expressed as $Cost - Contribution(R) = \frac{C(R)}{C(T)}$ where $C(R)$ and $C(T)$ are the cost of M-region and tree respectively. The heuristic algorithm proposed in [RM06] also differs from CRCDM in the triggering function in such a way that the service contribution of M-region is considered instead of the ratio between the number of descendent member nodes of R and the number of M-nodes in R . Service consideration of an M-region is expressed as

$$Service - Contribution(R) = \frac{|desc_T(R)|}{|Dest_T|} \text{ where } Dest_T \text{ is the set of all destination (member) nodes.}$$

Due to the consideration of both cost and service contribution of the M-region, the triggering function for the rearrangement of an M-region R in [RM06] is known as the *Cost-to-Service Ratio (CSR)* and is expressed as

$$CSR(R) = \frac{Cost - Contribution(R)}{Service - Contribution(R)}. \text{ Hence, the CSR of an M-region is}$$

expected to be small as it means that the M-region has small cost contribution and big service contribution to the tree, and should not be rearranged. On the contrary, if the CSR of an M-region is big, it means the M-region has a big cost contribution and small service contribution to the tree and hence should be rearranged to achieve smaller overall tree cost with the number of disrupted members kept reasonably small. [RM06] claimed that its proposed heuristic is able to reduce the number of disrupted members by approximately 20-85% while achieving the same cost performance as CRCDM proposed in [RMM99].

Chapter 2

Although the rearrangeable algorithms are able to produce good results, they are mainly reactive and hence may take considerable time to rearrange the required tree for each update. Efficient proactive rearrangement algorithms should be considered to increase the rate at which tree is rearranged so that potential race conditions and other problems faced by time-sensitive applications can be prevented.

2.4 Multicast Capable (MC) and Wavelength Conversion Capable (WCC) Nodes Placement in Optical Multicast

As mentioned earlier, due to the overwhelming capacity advantage of the optical fibers over the traditional coaxial cables and the increasing multimedia traffic volume over the Internet, it is logical to extend multicasting over the optical platform. It was mentioned in chapter 1 that the number of multicast and conversion capable nodes in transparent optical networks are of utmost importance since they affect the feasibility to support such a network. It was shown in many studies that there is no need to have all the nodes in the network to be multicast or wavelength conversion capable in order to obtain optimal reliable performance. We review some of the existing placement algorithms in this sub-section.

2.4.1 Performance Metrics used for MC and WCC Nodes Placement Problems

In optical WDM networks, traffic is carried across by means of wavelength channels. As there is a limit to the number of channels per fiber and a wavelength channel can only uniquely carry one unit of traffic at a time, the

Chapter 2

reliability issue in terms of blocking probability to the receiver nodes is of importance. Hence, one of the performance metrics that can be used to gauge if the tree is performance efficient is the average blocking probability in terms of the number of multicast sessions blocked or in terms of number of receiver nodes blocked. Also, as mentioned earlier, the number of optical devices, such as power splitters and wavelength converters, required is also a way to gauge if the constructed tree is efficient. We assume that a multicast capable (MC) node has the required number of power splitters to operate optimally and a wavelength conversion capable (WCC) node has the required number of converters to operate optimally. Thus, the numbers of multicast and wavelength conversion capable nodes are of interest here rather than individual numbers of optical devices although power-efficient design space can reduce the number of splitters required [AD00].

2.4.2 Placement of Multicast Capable (MC) Nodes

In MC nodes, incoming signal is split and sent to the output port of outgoing links in the tree. There are three kinds of splitting configuration in optical networks, namely, full, limited and sparse splitting. In full splitting, all the nodes in the network are MC and there is no restriction in the output degree for the splitters. Hence, achieving optimal cost in such configurations is no difference from that of the Steiner tree problem. The objective for this kind of configuration is to construct the optimal tree without any restriction and the trees are usually used as the benchmark for comparison with other configurations yielded by other algorithms. In limited splitting, the output degree of splitters at each multicast capable node is limited to a certain number.

Chapter 2

The purpose of this is to limit power loss after each splitting and reduce the amplification requirement. The lesser amplification in turn requires smaller number of amplifiers to be installed within each multicast node and in turn reduces the complexity and cost.

In sparse splitting, only a portion of the nodes in the network are MC and each MC node has splitting capability limited to a certain threshold with the upper bound decided by the physical topology of the network. Since each MC nodes requires a substantial number of splitters for optimal splitting, minimizing the total required MC nodes will reduce the total number of required splitters compared to just consider minimizing the total number of splitters. Therefore, each MC node is assumed to have the maximum splitting capability. Having no further restriction in the output degree of splitters, the output signal will need to be amplified along the way to the receiver node so that it can be recognized without any error. It was shown in [ZWQ00], [Ali02], [CY05], [TS06] and [HL07] that there is no need to have all the nodes in the network to be multicast capable to obtain near optimal performance. Hence, many studies on multicast routing problem have been done with the assumption of sparse splitting (see [MZQ98], [YAJ01], [WWY01], [TS06], [SSM06] and [CTL06]). However, since only a portion of the network nodes are required to be multicast capable for near optimal performance, the biggest challenge in sparse splitting configuration is the placement of the multicast capable (MC) nodes.

[Ali02] proposed the “Add” heuristic to optimally place MC nodes while assuming all the nodes have no restriction in wavelength conversion. The idea

Chapter 2

of the “Add” heuristic is to iteratively identify a node, which yields the best blocking performance when it is MC, for a set of incoming multicast sessions. All nodes in the network are assumed to be multicast-incapable (MI) initially. For each step of the “Add” heuristic, each of the MI nodes needs to compute the blocking performance for a set of incoming multicast sessions if it is a MC node. [Ali02] employed minimum path heuristic (MPH) to construct minimum path trees and proposed an analytical model to compute the average blocking probability among the receiver nodes for all multicast sessions. The MI node with the best performance is then selected to be a MC node and the MI nodes configuration in the network is updated. The process is repeated using the updated network and continues until the specified number of MC nodes have been placed.

2.4.3 Placement of Wavelength Conversion Capable (WCC) Nodes

Beside multicast capable nodes, transparent optical networks require wavelength conversion capable (WCC) nodes to minimize blocking. For each WCC node, the incoming wavelength channel carrying a unit of traffic can perform wavelength conversion to output the traffic using another wavelength channel. As each WCC node only performs wavelength conversion, the WCC node placement problem is not limited to only multicast networks. It is also applicable in unicast networks. Similar to power splitting, there are three kinds of wavelength conversion configuration, namely, full, limited and sparse wavelength conversion. In full wavelength conversion, there is no restriction in the conversion. This means that an incoming wavelength channel to any nodes in the network can be converted to any available outgoing wavelength channels

Chapter 2

to relay the traffic. Thus, in full wavelength conversion configuration, traffic to a receiver node is blocked if and only if at least one of the links along the multicast tree path has no free wavelength channels available for transmission.

In limited wavelength conversion, the range of wavelength channels which can be converted at the output is restricted. This will increase blocking compared to full wavelength conversion since not all output wavelength channels can be used to carry traffic to the receiver nodes. However, limited wavelength conversion will require fewer converters and hence incur lower cost compared to full wavelength conversion. [CCBLS00] proposed an analytical approach to study the blocking performance of WDM networks employing limited wavelength conversion while [VSK99] studied the performance between full and limited wavelength conversion and proposed a heuristic to optimally place limited wavelength converters in the network.

In sparse wavelength conversion, a portion of the network nodes are set to be WCC. It was shown in [SAS99], [VSK99], [AS00], [CLX00], [WLW01] and [CLC03] that similar to MC nodes, only a portion of the network nodes need to be WCC in order to achieve near optimal performance. Thus, the placement of WCC nodes is as vital as the placement of MC nodes for supporting transparent optical multicasting economically and efficiently. Each WCC node can have unlimited or limited conversion capability. However, as pointed out in [VSK99] that in transparent optical networks, wavelength converters are generally capable of only limited range of wavelength conversion. This is because the distortion of the optical signal increases with the increase in the range of

Chapter 2

wavelength conversion in wavelength converters. Thus, many existing studies consider only a fraction of the total wavelength channels available to be used for transmission for each set of incoming traffic to reduce or prevent any potential large distortion. Hence, for simplicity and without any loss of generalization, each WCC node is defined as a node which can convert incoming wavelength traffic to another wavelength within the range specified. Thus, the range of convertible wavelengths for a WCC node will be the full wavelength channel capacity of each fiber link for unlimited conversion. The total number of wavelength converters required for optimal performance can be reduced by either considering the coarse-grain or fine-grain converter placement problem or both. Most studies on coarse-grain converter placement problem assume that the converters required in each WCC node for optimal performance is more or less the same and the objective is to reduce the overall number of WCC nodes for optimal performance which will in turn reduce the total number of converters. The fine-grain converter placement problem is aimed at reducing the total number of converters regardless of the number of WCC nodes used. For fine-grain placement problem, besides employing efficient tree routing algorithms, complex and efficient WCC node interior designs are also required to achieve optimal placement for the converters. Hence, coarse-grain placement problem is normally studied since it is able to reduce the total number of wavelength converters significantly without having to go in depth into the WCC node's interior design. We shall henceforth refer to wavelength converter placement as WCC node placement unless specified otherwise.

Chapter 2

The WCC node placement problem for unicast communication was studied and addressed in [VSK99], [SAS99], [WLW01] and [CLC03]. Since each WCC node is only concerned with converting an incoming wavelength to another outgoing wavelength, it is considered a unicast operation and hence the algorithms proposed by these papers can also be applied in those multicast communication which employs full splitting configuration by considering the lightpaths within each light-tree used to route the traffic. [VSK99], [SAS99] and [CLC03] proposed optimal placement algorithms for WCC nodes with the aim to achieve minimum overall average blocking probability. [VSK99] proposed placing WCC nodes at those most congested nodes, those on long lightpaths and with high nodal degree and those where conversion is extremely high. The performance of the proposed algorithm for different range of possible wavelength conversion for each WCC node is also studied in [VSK99]. [SAS99] showed that dynamic programming is another alternative method to find optimal location for WCC nodes in ring and bus topologies to obtain minimum blocking probability possible.

In [CLC03], the efficiency of the WCC node placement algorithm was argued to be linked to the routing and wavelength assignment (RWA) problem strategy used. For *fix alternate routing and first-fit wavelength assignment* (FAR-FF RWA) strategy, [CLC03] proposed the MBPF algorithm. In MBPF, all nodes are assumed to be non-WCC node. Each non-WCC node will then be considered as a WCC node and have the routing paths and wavelength assignment performed using FAR-FF RWA algorithm. The average blocking probability to deliver traffic to each receiver node is then computed for a fix set

Chapter 2

of incoming traffic. The one with the least blocking will be placed as a WCC node permanently and this continues until the given numbers of WCC nodes are placed. For the least load routing and first-fit wavelength assignment (LLR-FF RWA) strategy, [CLC03] proposed the WMSL algorithm. In WMSL, WCC node is placed with the aim to minimize the sum of maximum segment length over the entire network to reduce the blocking probability due to wavelength continuity constraint. [CLC03] claimed and showed that their proposed placement algorithms together with the corresponding RWA strategy used outperformed the existing algorithms in blocking performance.

On the contrary to [VSK99], [SAS99], and [CLC03], optimal placement algorithm known as WCP was proposed in [WLW01] with the aim of minimizing the number of wavelengths used. One reason is that by minimizing the number of wavelengths used, more wavelengths can be available for use by other traffic requests and hence the overall blocking performance can be improved. In WCP, the lower bound number of wavelengths used is first computed by assuming all the nodes are WCC. All the nodes are then assumed to be non-WCC. Each non-WCC node with the largest transit load (largest number of lightpaths passing the node) is then considered iteratively if it requires to be WCC to converge the number of wavelengths used to the lower bound value computed earlier. It was shown in [WLW01] that WCP is able to achieve the lower bound number of wavelengths used by just placing a small number of WCC nodes at some key nodes in the network.

2.4.4 Placement of Multicast Capable (MC) and Wavelength Conversion Capable (WCC) Nodes

Although MC and WCC nodes placement problems can be solved separately, they are not mutually independent since MC and WCC nodes contribute to the overall performance. Hence, it was stressed in [YL03] and [CY05] that it was more desirable to solve the MC and WCC nodes problems jointly. The first comprehensive study on locating the optimal placement of MC and WCC nodes jointly was reported in [YL03]. The objective in [YL03] was to design light-tree based logical topology to minimize the number of wavelength channels used by solving the optimal routing and wavelength assignment problem for delay bounded light-trees using MILP. The optimal placement of MC and WCC nodes are then found as part of the solution obtained from the MILP. Although this approach is able to find optimal placements, it was only in the context of the logical topology design; there was no investigation in the direct relationship between the MC and WCC nodes placements. [CY05] proposed solving the MC and WCC node placement problems directly using ILP with the objective to minimize the number of wavelength channels used. [CY05] showed that MC and WCC nodes placements indeed have a close relationship and reinforced the earlier claim that it was more desirable to solve MC and WCC nodes placement problems jointly. [CY05] also claimed that their model is applicable to any optical network topology with any multicast configuration such as limited or sparse splitting.

Most of the existing heuristics reviewed considered MC and WCC node placement problems independently. But it was shown in [YL03] and [CY05]

Chapter 2

that they are not mutually independent since both MC and WCC nodes contribute to the overall performance. However, there are enough evidences in existing works that showed that the difference in performance is not significant when the two sub-problems are considered independently. In addition, although [YL03] and [CY05] considered the placement problems jointly, their approach is to find the optimal results using MILP and ILP which is very time consuming and may have problems for larger networks. From the surveys, it is clear that research on the impact of sparse-splitting (partial network nodes are MC) and sparse-wavelength conversion (partial network nodes are WCC) in blocking performance is lacking. Hence, fast and flexible new heuristics have to be developed to show that the near optimal performance can still be achieved with efficient placement of MC and/or WCC nodes in the optical network to cater for all kinds of physical constraint (in terms of numbers of MC node and WCC node allowed). This is only possible if the impact of MC node placements on WCC placements and vice versa are taken into consideration in the placement heuristics.

2.5 Dynamic Multicast Routing and Wavelength Assignment (MC-RWA) in Optical Multicast

The multicast routing and wavelength assignment (MC-RWA) problem is another important area which contributes greatly to the construction of performance efficient trees in optical multicast networks. The existing algorithms divided the MC-RWA problem into multicast routing and wavelength assignment sub-problems. As the MC-RWA problem is the combination of the multicast routing and wavelength assignment sub-problems,

Chapter 2

the performance metrics used to gauge the efficiency of an algorithm for the MC-RWA problem are usually the number of wavelength channels used, the number of wavelength conversions and the session blocking probability. This is because the fewer the wavelength channels used, the lower is the blocking probability due to lack of free channels. In addition, fewer wavelength channels used will mean less resource to be reserved and it helps to lower the overall cost. Also, smaller number of wavelength conversions will require fewer converters in the network. However, as session blocking can be due to the absence of wavelength converters or power splitters, minimizing average session blocking probability is a more direct way to measure the reliability performance of an algorithm for MC-RWA problem. The multicast routing sub-problem in MC-RWA problem for optical multicast when considered independently is the same as the minimum cost tree problem mentioned earlier and can be modeled as the minimum cost Steiner tree problem. The additional physical constraints imposed, which may exist, are the absence and presence of MC and WCC nodes in the optical network.

2.5.1 Algorithms for MC-RWA Problem

[JDH01] solved the MC-RWA problem without any consideration of the physical constraints and focused on the optimal heuristic algorithms for the two sub-problems. This means that the splitting capability of each node is bounded by its physical topology and wavelength conversion is possible if required. [JDH01] aimed to minimize cost and the total number of wavelengths used, and hence assumed no limit in the link capacity (i.e. number of wavelength channels in each link). However, to cater for delay sensitive applications, the trees

Chapter 2

constructed in [JDH01] are delay constrained. For the multicast routing heuristic, MST is constructed from a graph consisting of source and receiver nodes connected to each other by means of minimum cost path in the original graph. The delay from the source to each receiver node is then checked to see if it is bounded. Those which are not bounded will be connected to the source using the minimum delay path. The edges in MST will then be expanded to the corresponding minimum cost or delay paths and pruning is carried out to remove any redundant links and nodes. Two methods based on the greedy algorithm were proposed for the wavelength assignment heuristics, which only assign a new wavelength when it is needed. The first method, known as single-phase algorithm, considers existing used wavelengths for the assignment of the new tree. This is made possible by removing any links which do not have a particular used wavelength available during the routing tree phase. The decision pertaining to which used wavelength to be considered first is based on some simple policies. The second method, known as two-phase algorithm, is an extension of the single-phase algorithm with the aim to reduce the network load. This is based on the fact that the required number of wavelengths is at least able to handle the network load. Hence, reducing network load will reduce the required number of wavelengths. For the first phase of the two-phase algorithm, the routing tree with the least network load is searched. The tree is then assigned a wavelength using the single-phase algorithm. In this case, network load is defined as the maximum number of routing trees which traverse a link among all links in the network. Although [JDH01] proposed an effective algorithm to achieve low number of wavelengths used for constructing delay constrained low cost trees, the authors of [JDH01] simplify their algorithms by

Chapter 2

assuming non-blocking scenarios which is not realistic unless enormous bandwidth is catered for the multicast application.

[ZAM02] proposed three heuristics to construct least cost delay constrained trees which assumed all the nodes to be MC and WCC. The cost considered included both the wavelength usage cost and the wavelength conversion cost while the delay considered is the wavelength conversion delay. The wavelength assignment strategy is indirectly enforced by the objective to minimize cost which consists of the wavelength conversion cost component. The first heuristic, known as *Least Delay Replacement* (LDR), first constructs the least cost tree using SPH and replaced those nodes whose delay-constraint cannot be satisfied with least cost delay bounded paths. The second heuristic, known as ILDR, was almost similar to SPH except that in each step to connect a receiver node, the delay bound condition is checked first and a least cost bounded path will be constructed if a delay violation existed. The third heuristic, known as LDF, was similar to ILDR except that the least cost connection between the active tree and the receiver node with the least delay is selected to be connected to the tree. [ZAM02] compared LDR and ILDR with some of the existing minimum cost tree heuristics such as BSMA and KPP and showed that they performed reasonably well.

It was mentioned earlier that not all nodes in the network are required to be MC or WCC for good performance. [ZWQ00] proposed to use some well-known heuristics to solve the MC-RWA problem under sparse splitting scenario in which only a fraction of the network nodes are MC. To make the studies more

Chapter 2

complete, other scenarios corresponding to full, sparse and no wavelength conversion were also considered in [ZWQ00]. In the sparse splitting scenario, since not all nodes are MC, multiple light-trees, known as light-forest, are required to deliver traffic for each multicast request. For the multicast routing sub-problem in MC-RWA, it was mentioned in [ZWQ00] that constructing a light-forest is a straight-forward way to modify a multicast tree. A light-forest can be constructed using any existing Steiner tree algorithm, through rerouting certain parts where the MI nodes may be treated as branching nodes. In order to prevent any splitting violation in any nodes, the links to those excess children of the violated node are cut and rerouted. Hence, two tree rerouting algorithms, namely, *Re-route-to-source* and *Re-route-to-any* were proposed in [ZWQ00]. The main difference between the two is that the *re-route-to-source* algorithm will connect each cutoff child to the first MC node in the reverse direction of the path between the source and the violated node while the *re-route-to-any* algorithm will connect the cutoff child to any part of the light-forest as long as a node's splitting constraint is not violated.

[ZWQ00] also proposed two algorithms, namely, *member-first* and *member-only*, for constructing light-forest from scratch. For *member-first*, light-trees are constructed one at a time. During each step in the construction of a light-tree, the unconnected node that has the shortest distance to the source in terms of hop count is selected to connect the tree via the link between the tree and the node. If a tie exists in the selection between a receiver node and a non-receiver node, the former will be preferred for selection. For all other cases of a tie, the selection will be done arbitrarily. The *member-only* algorithm is similar to the

Chapter 2

member-first algorithm except that only receiver nodes are considered for connection via minimum hop path in each step of the tree construction process. [ZWQ00] adopted the first-fit wavelength assignment strategy to assign wavelengths to edges in trees constructed based on the tree routing heuristic. Each segment in the trees is defined as a path in which all links are assigned with the same wavelength. The creation of segments is due to the need for the constructed tree to obey the wavelength continuity constraint. Although good performance can be achieved in the algorithms proposed in [ZWQ00], only non-blocking scenarios are considered there.

[YAJ01], [HM06] and [CLT06] focused on the multicast routing sub-problems for sparse splitting scenario and assumed no wavelength conversions at all. In [YAJ01], a heuristic based on tabu-search was proposed to construct minimum edge trees. Tabu search basically involves two main steps. The first step is to obtain an initial solution and the second step is to iteratively improve the solution until no further improvement is possible. It was shown that the tabu search heuristic is able to achieve almost similar results as that obtained by ILP. However, [YAJ01] basically solved the well-known minimum cost Steiner tree problem and ignore the wavelength assignment problem which has significant impact on cost and blocking performances especially for dynamic traffic in networks with limited capacity. [HM06] considered the splitting limit of each node and proposed a heuristic that aimed to achieve load balancing while minimizing the number of requests blocked. [HM06] employed the *member-only* approach to overcome the physical constraint brought about by the sparse splitting condition and adopted the first-fit wavelength assignment strategy for

Chapter 2

the wavelength assignment sub-problem. To achieve the objectives of load balancing and minimizing request blocking probability, the selection of each path to connect a receiver node during each step of the tree construction process is based on the minimum residue bandwidth of all links in the path and the path distance is considered next if there is a tie. [HM06] showed that its proposed heuristic outperformed the *member-only* approach proposed in [ZWQ00] in terms of request blocking probability for low and average loads.

[CLT06] proposed a heuristic known as delay multicast routing problem under delay-constraints (DMR-DC) for dynamic multicast routing in sparse splitting condition without wavelength conversion. Like in [YAJ01], the authors of [CLT06] did not consider the important wavelength assignment sub-problem but focused solely on the multicast routing sub-problem. The difference compared to [YAJ01] is that [CLT06] constructs delay constrained trees while [YAJ01] only constructs non-delay-constrained trees. There are three phases in DMR-DC. The first phase is to generate the initial minimum delay tree by merging all the minimum delay lightpaths from the source to the receiver nodes by means of Prim's algorithm. No solution exists if at least one of the paths is not delay constrained. The second phase is to iteratively reduce the cost of the tree by rerouting the path (for a node-pair) in the tree obtained from the first phase that gives the largest cost reduction. The selection of node-pair for rerouting is based on the rerouting scheme adopted. [CLT06] presented three rerouting methods. The first method is known as *all nodes reroute to parent* (ARP) where the node-pair must be within the path from source to any of the receiver nodes. The second method is known as *destinations reroute to nearest*

Chapter 2

node (DRNN) where either one of the node-pair must be a receiver node. The third method is *all nodes reroute to nearest node* (ARNN) where the node-pair can consist of two nodes from any part of the tree. After the first and second phases, some of the nodes may experience splitting violation due to the sparse splitting constraint. Hence light-forest is constructed in the third phase with the preference to minimize wavelength consumption (number of wavelengths used). It was shown that DMR-DC is able to achieve better cost performance than that the algorithms proposed in [KMB81] and [JDH01]. However, it should be noted that [CLT06] considered only single request routing and without any potential blocking scenarios.

[WWY01] argued that MC nodes which use optical amplifiers to compensate power loss due to splitting are expensive, and hence resort to using MC nodes without optical amplifiers and aim to minimize power loss from splitting. Sparse splitting and sparse wavelength conversion were considered to further reduce the number of MC and WCC nodes required in the network. [WWY01] assumed that each node has equal chance to be MC or WCC or both, and considered non-blocking scenario where there is no limit in the number of wavelength channels per link. As a result of only some nodes are MC, light-forest instead of a single tree is required for each multicast session. The heuristic known as *centralized-splitting algorithm* was proposed in [WWY01] to construct the light-forest based on a set of design considerations that reduce power loss. The first consideration is that if concatenated splitting in a particular tree exists, it is better to replace the concatenated splitting by one MC node. The second consideration is that when the MC node is located as far from

Chapter 2

the source node as possible, the power loss is smaller. The last consideration is that a node where more splitting occurs is preferred to assign as a MC node to reduce the number of lightpaths required to connect between the source node and destination node due to lack of splitting capabilities in any of the transition nodes in the routing tree. Hence, the proposed heuristic consisted of two phases. The first phase is to construct minimum hop light-forest using the *member-only* heuristic proposed in [ZWQ00]. The second phase is to modify each tree in the light-forest such that centralized splitting is preferred. Readers are encouraged to refer to [WWY01] for details. First-fit wavelength assignment is then used for the wavelength assignment sub-problem. It was shown in [WWY01] that the *centralized-splitting algorithm* is able to achieve lesser power loss than the *member-only* heuristic while not degrading any performance in terms of bandwidth (number of edges or hops in trees), number of wavelengths used and average delay between source and receivers. Again, the algorithm is simplified by not considering the blocking scenario where the number of wavelength channels per link is limited and blocking in the multicast session occurs when at least one of the links in the tree has all its channels occupied.

Recently, [HL07] proposed a heuristic for handling the multicast routing problem in WDM network assuming sparse splitting but ignoring rerouting. [HL07] suggested that the resultant tree should comprise of a MC-tree and a set of lightpaths connecting receiver nodes not in the MC-tree. Each MC node is assumed to have both splitting and wavelength conversion capabilities; hence, the total cost is the summation of the MC-tree cost and the cost of all lightpaths. The construction of an MC-tree is made simple by applying an existing

Chapter 2

minimum cost Steiner tree heuristics, such as the one proposed in [KMB81]. The question considered is whether the selection of MC nodes needs to be included in the MC-tree construction. Two methods of selecting MC nodes are proposed. The first, known as *on tree MC node first*, (OTMCF) aimed at minimizing the cost of MC-tree and the second, known as *nearest MC node first* (NMCF), aimed at minimizing the cost of lightpaths. The major difference between heuristic proposed in [HL07] and other popular rerouting scheme is that, in the former, the node's splitting constraint violation is checked during each step of tree construction. It was claimed and shown in [HL07] that its proposed heuristic performs better than the *member-only* approach proposed in [ZWQ00] in terms of wavelength channel cost and wavelength channels used per link. Again, blocking scenarios which will make the proposed heuristic more complex are not considered in [HL07].

Table 2-1: Summary of existing reviewed heuristics.

Heuristic	Splitting Configuration	Wavelength Conversion Configuration	Consider blocking scenario?	Multicast Tree Routing and Wavelength Assignment considered as two sub-problems individually or considered concurrently?
ZWQ00	Sparse	Full, Sparse, None	No	Two sub-problems individually
JDH01	Full	Full	No	Two sub-problems individually
YAJ01	Sparse	None	No	Two sub-problems individually
WWY01	Sparse	Sparse	No	Two sub-problems individually
ZAM02	Full	Full	No	Two sub-problems individually
HM06	Sparse	None	Yes	Two sub-problems individually
CLT06	Sparse	None	No	Two sub-problems individually
HL07	Sparse	Sparse	No	Two sub-problems individually

Table 2-1 lists all the heuristics for solving MC-RWA problem in optical multicast that have been reviewed in this chapter. Although existing heuristics are able to construct trees with minimize cost, wavelength usage per link, number of wavelengths and bandwidth, and take into considerations various

Chapter 2

MC and WCC node configurations, few of them consider how the proposed algorithms could be modified for use in the more complex blocking scenario. Also, all the existing heuristics consider the multicast routing and wavelength assignment as two individual sub-problems. These two sub-problems are inter-dependent on when network performance needs to be optimized; hence heuristics which consider both two sub-problems concurrently for the more complex blocking scenario under various MC and WCC node configurations will most likely yield better and more meaningful results.

2.6 Summary

In this chapter, we presented detailed reviews on heuristic algorithms for the construction of performance efficient trees for multicast networks. The fundamental problem of constructing minimum cost tree was first looked into and the proposed heuristics for solving the problem are classified into two groups -- based on single tree construction and based on shared tree construction. We reviewed the heuristics for both groups and examined their pros and cons. Although many useful heuristics have been proposed based on single tree construction, there are not many that are based on shared tree construction.

Then, we extended our reviews to those heuristics for constructing minimum cost trees with dynamic membership support and grouped them into rearrangeable and non-rearrangeable algorithms. We explained that rearrangeable algorithms are more useful and relevant in real worlds due to the need to balance cost and member disruption. We also presented important

Chapter 2

performance metrics that can be used to gauge the usefulness of any proposed heuristic algorithms for tackling the dynamic membership multicast problem. From the reviews, we noticed that all the rearrangeable algorithms are reactive in nature and suggested a need for proactive algorithms to prevent any potential race conditions.

We also reviewed existing heuristic algorithms for constructing performance efficient trees for optical multicast. We explained the benefits of employing multicasting in transparent optical networks and examined the problem of limiting the number of expensive and complex MC and WCC nodes through optimal placement. Comprehensive reviews on placement heuristics for MC node only, WCC node only, and both MC and WCC nodes are presented. We noticed that most of the proposed heuristics consider the MC and the WCC node placement sub-problems independently, and only a few did consider the sub-problems jointly. In general, these algorithms made use of such optimal algorithms as MILP and ILP which are not time-efficient and there is a need to develop fast and efficient heuristics to consider the two placement sub-problems jointly for better result.

Lastly, we surveyed the literature on the MC-RWA problem. The proposed heuristics for the MC-RWA problem for different MC and WCC node configurations are reviewed. We explained the lack of dependency in the existing heuristics to solve the multicast routing and wavelength assignment sub-problems jointly. We also stressed that it was important to consider the blocking scenarios to ensure that the proposed heuristics are able to solve the

Chapter 2

MC-RWA problem efficiently in the more realistic environment where bandwidth is limited and blocking occurs due to lack of bandwidth or physical constraints (lack of MC or WCC node). To the best of our knowledge, this is the first time that the performance efficient tree construction problem for both IP and optical multicast networks are reviewed extensively in such a detailed manner.

Chapter 3

Efficient Heuristics for the Construction of Minimum Cost Trees for Multi-Groups Multicast

3.1 Introduction

Multicasting is a way to transmit traffic efficiently from a single source to multiple destinations. We refer to the tree used to deliver traffic for each multicast session as the delivery tree. The cost of each delivery tree must be kept minimal such that multicasting can be supported in a cost-efficient manner. Such a problem is known as the minimum cost tree (MCT) problem and can be modelled as the NP-complete Steiner tree problem [HR92] for networks where the Steiner points are the receivers. Since a considerable amount of time is required to solve NP-complete problems, there is a need to develop heuristic algorithms to find a sub-optimal solution with much less time.

It is observed that the algorithms for the construction of delivery trees can be grouped into two approaches. The first approach, known as single-tree construction model (MSi-T), constructs the delivery tree for each multicast session from scratch without relying on pre-constructed trees other than those minimum cost or delay paths required in tree construction. For MSi-T, the total number of trees constructed from scratch is equal to the number of incoming multicast sessions. The second approach, known as multiple shared-tree

Chapter 3

construction model (MSh-T), constructs delivery trees belonging to each multicast group using a pre-constructed shared-tree for that group. For MSh-T, the number of shared-tree construction is equal to the number of multicast groups involved. Note that each incoming multicast session is originated from one of the multicast groups involved. A multicast group is normally assigned to an application over the Internet. All hosts using the same application are required to register as a member to the multicast group assigned to the application. If a request to send data is made from one member to the other members in the multicast group, a multicast session for this request will be generated to deliver data via the delivery tree constructed for the multicast session. MSh-T focuses on reducing overheads and tree construction time while MSi-T focuses on reducing the tree cost. The core-based tree (CBT) algorithm [BFC93] is an example of a minimum cost tree construction algorithm based on MSh-T.

Although algorithms based on MSi-T are able to construct trees with much lower cost, the overheads and tree construction time incurred are too substantial to be ignored. This is especially true for multi-group multicast where the multicast sessions from the same multicast groups will use the same cost-efficient tree edges to deliver the data most of the time. Hence, algorithms based on MSh-T to construct delivery trees will reduce the overheads and tree construction time drastically. This serves well for all kinds of applications since delivery trees can be quickly set up and data quickly delivered. In addition, it also prevents race conditions [RMM99] from taking place. Hence, we propose a new algorithm, known as Multi-Groups Shared Tree algorithm (MGST) for

Chapter 3

constructing delivery trees from a pre-constructed spanning shared-tree for all multicast groups. The key idea is to construct trees for multicast sessions of all multicast groups using a common shared-tree. This is in contrast to the existing MSh-T heuristics where only multicast sessions from the same multicast group use a unique shared-tree to construct their delivery trees. Since MGST requires only one common spanning shared-tree to be constructed from scratch, the overheads and tree construction time will be extremely low compared to algorithms based on the MSi-T and MSh-T approaches. Due to the need to construct only a common spanning shared-tree, we refer to our proposed algorithm as the common shared-tree construction model (CSh-T). For convenience, we use the term “spanning shared-tree” and “shared-tree” interchangeably in this dissertation. It should be noted that the spanning shared-tree in CSh-T spans to all nodes in the network, while the spanning shared-tree in MSh-T spans only to all the member nodes in the multicast group assigned to that shared-tree. We also improve MGST for delay-constraint scenarios. We refer to the delay constrained version of MGST as DC-MGST.

The rest of the chapter is organized as follows. In section 3.2, we describe some of the existing heuristics in greater details. The problem specification and the proposed heuristics are described in section 3.3. Also, we prove in section 3.3 that DC-MGST guarantees a solution if the delay constraint is always greater than the maximum delay of the minimum delay paths. Section 3.4 presents the simulation results and discussion of the proposed heuristics. Finally, section 3.5 concludes the chapter and highlights the advantages of the proposed heuristics.

3.2 Background

3.2.1 Existing heuristics based on multiple single-tree construction model (MSi-T)

The classical solution for the MCT problem using MSi-T without any delay-constraint is the KMB heuristic proposed in [KMB81]. KMB first constructs the minimum cost path between any two nodes in the network. The source and the receiver nodes are then connected in a distance graph by merging all those minimum cost paths between the source and receiver nodes. After the distance graph is computed, Prim's algorithm is performed on it to obtain the spanning tree from the source to the receiver nodes. The edges of the obtained spanning tree are then converted back to the corresponding minimum cost paths in the physical graph. Pruning is finally carried out to remove any unnecessary nodes and edges from the spanning tree. It was shown that KMB has a worst time complexity of $O(|S_e||V|^2)$ where S_e and V are the set of Steiner points and vertices respectively, and the tree cost obtained is no more than $2(1 - \frac{1}{L})$ times that of the optimal minimum cost tree, where L is the number of leaves in the optimal minimum cost tree. Hence, KMB is commonly used as the benchmark for evaluating the tree cost efficiency of other tree-construction algorithms. The CKMB algorithm reported in [SL98] is an extension of KMB for constructing delay constrained minimum cost trees.

The CMCT algorithm proposed in [KPP93a] adopts a centralized approach based on heuristic for constructing a delay constrained minimum cost tree. CMCT is quite similar to KMB except that each edge in the distance graph now

Chapter 3

represents a delay constrained minimum cost path (DCMCP) instead of minimum cost path (MCP). In addition, the function for selecting the next new node to connect to the active tree during tree construction is made more complex compared to non-delay constrained situations. One selection function focuses on cost reduction while the other focuses on cost reduction and delay residue maximization. It was shown that the selection function which focuses on cost reduction and delay residue maximization is able to connect more receiver nodes within the specified delay-constraint even in the event of very small or strict delay-constraint but at the expense of slightly higher tree cost.

The centralized approach proposed in [KPP93a] requires the broadcasting of network information which is suitable for small networks but less accurate, reliable and more expensive for larger networks. Hence, a distributed heuristic called DMCT was introduced in [KPP93b]. DMCT adopts the two selection functions proposed in [KPP93a] but the difference is that the selection function selects the next ‘best’ adjacent edge to connect a new node to the tree instead of selecting the next ‘best’ path during tree construction. We refer to the adjacent edge of the active tree as an edge directed to a new neighbour node of the active tree. DMCT is more scalable than CMCT for larger networks but the scalability is provided at the expense of higher tree cost.

The BSMA algorithm proposed in [PZG98] is another heuristic algorithm based on the centralized approach for constructing delay constrained minimum cost trees. Basically there are two stages for BSMA. The first stage is to construct an initial minimum delay tree rooted at the source and spanning to each of the

Chapter 3

receiver nodes. This is done by merging all the minimum delay paths between the source and the receiver nodes. The second stage is to improve the cost of the initial minimum delay tree until no further improvement can be made. BSMA ensures that a delay constrained tree is obtained first before improvement is made to reduce the initial tree cost.

The centralized heuristic algorithm known as SLIM was proposed in [AZ01]. SLIM can be invoked to construct delay constrained minimum cost trees by selecting the delay constrained path with the least cost among all the minimum cost and minimum delay paths between any two nodes to connect the next new node during tree construction. Since some cost-inefficient minimum delay paths may be selected, the tree cost can be further improved using the same concept as in BSMA. However, such improvement is limited since the candidate paths beside the minimum cost paths for connection selection during each step of the construction are not cost optimal. Hence, K-SLIM where the k least cost DCMCP are included as the candidate paths for connection selection was also proposed in [AZ01]. The number of cost improvement steps for SLIM / K-SLIM is much lesser than BSMA due to the construction of a more optimized initial tree.

Another centralized heuristic called STAR, was proposed in [LMZ04] for computing delay constrained minimum cost trees. Unlike those previously mentioned heuristics, STAR aims to reduce the algorithm's time complexity by constructing only the paths needed for tree construction. STAR consists of three stages. The first stage is the computation of DCMCPs for all receivers to all

Chapter 3

other nodes. The second stage is the construction of the initial tree. The least cost DCMCP is selected each time to connect the next new node in the tree until all the receivers are connected to obtain the initial tree. The final step is the improvement of the initial tree through rearrangements of sub-trees rooted at each trim node until no further improvements can be made. Each trim node is defined as the child of either the source node or node with degree greater than 2. As a result of constructing fewer number of paths and more efficient cost improvement, STAR is shown to achieve better results than CMCT, BSMA, SLIM and K-SLIM

3.2.2 Existing heuristics based on multiple shared-tree construction model (MSh-T)

The core-based tree (CBT) algorithm [BFC93] is a classic algorithm employing *MSh-T* to solve MCT problems. A spanning shared-tree is constructed from scratch for each multicast group. To minimize the tree cost, all the member nodes of the group will connect to the selected core node via minimum cost paths. For each incoming multicast session originated from the multicast group, the shared-tree is used to direct the traffic from the source to the receiver nodes within the group. The selection of the core node is vital to the optimality of the shared-tree. Unfortunately, the selection of core node is complex and difficult. Another potential problem faced by CBT is that congestion may occur at the core node since it is the relay node between any two nodes in the multicast group.

3.2.3 Proposed heuristics based on common shared-tree construction model (CSh-T)

Our proposed heuristic MGST aims to construct only one common spanning shared-tree and use it to construct the delivery tree for each multicast session originated from any multicast groups. As the common shared-tree is cost-effectively constructed to span from one node to all others in the network instead of just to the member nodes in a particular multicast group like in MSh-T, the pruning of redundant edges in the common shared-tree will not achieve a cost-efficient tree for delivering data to a fraction of the network nodes. Thus, there is a need to improve the tree after performing pruning of the common shared-tree. We introduce the “trace-back” feature to MGST (became known as MGST-T) to lower the cost of the tree after performing pruning from the common spanning shared-tree. MGST-T is only applicable to solve the MCT problem without any constraint. It is not sufficient to solve the MCT problem for delay-constraint scenarios. For example, since the common shared-tree is cost-effectively constructed to span from one node to all the other nodes in the network, it does not guarantee all nodes to receive data within a specified delay-constraint. Hence, we also proposed the DC-MGST-T algorithm to achieve similar good results as MGST-T for delay-constraint scenarios. In order not to introduce additional complexity and for continuity, DC-MGST-T is proposed in such a way that it can be reduced to MGST-T if no or very relax delay-constraint is specified. Likewise, MGST-T can be upgraded to DC-MGST-T if delay-constraint is specified. The details of the proposed heuristics and their simulation results will be presented in the later sections of this chapter.

3.3 Details of Proposed Algorithms based on CSh-T

3.3.1 Problem Formulation

3.3.1.1 Network Model

The network is modeled as a graph $G = (V, E)$ where V and E are the set of vertices and connecting edges respectively. Each edge e consists of the cost metric $C(e)$ and delay metric $D(e)$. The tree cost is the total cost of all edges within the tree, and can be expressed as:

$$C(T) = \sum_{e \in T} C(e)$$

The cost and delay of a path between two nodes can be expressed respectively as:

$$C(P(a, b)) = \sum_{e \in P(a, b)} C(e)$$

$$D(P(a, b)) = \sum_{e \in P(a, b)} D(e)$$

If a path is part of the delivery tree, we express the path as $P_T(a, b)$. Each multicast group ϕ_i is assumed to be assigned to a particular multicast service, and all its members $m \in M_{\phi_i}$ are registered to the group. A list of multicast session requests randomly selected from any of the multicast groups is expressed in the set $L = \{S_0, \dots, S_x\}$ and $S_j = (s, Y)$ where s and Y are the source node and the set of receiver nodes respectively, and $M_{\phi} = \{s\} \cup Y$.

3.3.1.2 Problem Specification

We consider the delay constrained minimum cost tree problem with blocking. The multicast session is considered blocked if at least one of its receiver nodes is unable to receive the data within the specified delay-constraint. We refer the

Chapter 3

probability for such blocking as session blocking probability (SBP). The delay-constraint (DC) considered in this chapter includes uniform and non-uniform constraints where the delay-constraint for each group is identical (i.e. $\Delta_\phi = \Delta_{\phi_0} = \Delta_{\phi_1} = \dots = \Delta_{\phi_z}$) and randomized respectively. The delay-constraint for a multicast group is defined as the maximum delay allowed for delivering data from the source node to each member node in the multicast group. We denote the tree construction time and overheads required in the construction as $t(T_s)$ and $K(T_s)$ respectively. The objective of our proposed algorithms is to construct the tree T_s for each incoming session request $S \in L$ whereby the following conditions are fulfilled.

- (a) T_s spans the set M_ϕ where $S \in \phi$.
- (b) $D(P_T(s, y)) < \Delta_\phi, \forall y \in Y, S = (s, Y)$.
- (c) $t(T_s)$ is minimal.
- (d) $K(T_s)$ is minimal.
- (e) $C(T_s)$ is the minimum among all the possible candidates satisfying all the above conditions.

It can be observed that DC-MGST-T requires the construction of each tree which satisfies all the above conditions while MGST-T can ignore condition (b) since no or very relax delay-constraint will be specified for MGST-T to solve the MCT problem. For delay-constraint scenarios, we assume that the delay-constraint among the members of the same multicast group is always uniform, and varies the delay-constraint of members in different multicast groups for non-uniform delay constraint. This is reasonable as each multicast group is assumed to be assigned to a particular multicast service and hence the delay

Chapter 3

requirement of all the members within the same multicast group should be the same to achieve the desirable quality of service.

3.3.2 Description of Proposed Algorithms: MGST-T / DC-MGST-T

MGST-T consists of two stages. The first stage is to construct the common cost-efficient spanning shared-tree. The second stage is to construct the delivery tree for each incoming multicast session through pruning of redundant edges in the common shared-tree from the first stage. We shall refer to the common spanning shared-tree as shared-tree and the tree constructed from the shared-tree as the resultant tree to avoid any confusion. The second stage also performs “trace back” on the resultant tree to reduce the tree cost. The final version of the tree after all the improvements is referred as the final tree.

For delay-constraint scenarios, additional stages are required. This is because the shared-tree is constructed in such a way that the resultant trees constructed from it after performing “trace back” is cost minimized but may incur large delays between the source and receiver nodes, therefore some of the receiver nodes may not be able to receive data within the specified delay-constraint using the resultant tree obtained from the second stage. To resolve this, MGST-T is upgraded to DC-MGST-T by including an additional third stage which uses cost improved minimum delay paths to deliver data to those receiver nodes. Thus, MGST-T can be upgraded easily to DC-MGST-T for delay-constraint scenarios and vice versa for non-delay-constraint scenarios.

Chapter 3

3.3.2.1 Stage 1: Cost-Efficient Spanning Shared Tree Construction

We defined active shared-tree as the shared-tree that is being constructed up to a certain time. For each step of the shared-tree construction, we consider each adjacent new node of the active shared-tree as the potential node to connect to. In order to have a cost-efficient shared-tree, the selection function (SF) which determines the adjacent new node to connect to the active shared-tree in each step of the construction must be formulated in such a way that the cost to deliver traffic via (as a transition node) or from the node (as a source node) is small. The role of a transition node is to relay data. Hence, for each step of the shared-tree construction, the SF of a particular unconnected adjacent node j should consider the average minimum cost to deliver traffic to each node from node j either as a transition node or the source node.

Figure 3-1: An example of each step of the shared-tree construction.

As illustrated in figure 3-1, each potential node j for connection to the active shared-tree needs to compute the average minimum cost to deliver traffic via node j in order to determine its SF. Hence, the SF computation consists of computing the average minimum cost from any of the nodes in the active

Chapter 3

shared-tree to node j , and the average minimum cost to any of the nodes not in the tree from node j , and vice versa. Since we assume symmetric link cost, only one direction needs to be considered. The computation of the average minimum cost to any of the nodes not in the tree from node j should be approximated by the average minimum path cost from j to the unconnected nodes. This is because those unconnected nodes are not yet connected via the shared-tree. Let B and R be the set of unconnected nodes and the set of connected nodes to the active shared-tree respectively. Denoting $e_{k,j}$ as the minimum cost link between the active shared-tree $\hat{T}'$ and node j and $C_{MCP}(P_{x,y})$ as the minimum cost path between node x and y , the selection function of a unconnected node j can be expressed as,

$$SF(j) = \left[\frac{1}{|R|} [C(\hat{T}') + C(e_{k,j})] + \frac{1}{|B|} \sum_{b \in B} C_{MCP}(P_{j,b}) \right] \quad (3.1)$$

The first term in (3.1) corresponds to the average minimum cost from any of the nodes in the active shared-tree to node j . The second term corresponds to the average minimum cost from node j to any of the nodes not in the active shared-tree. The unconnected node with the least SF value is selected to connect to the active shared-tree via the minimum cost link between the active shared-tree and the node. The process is repeated until all nodes are connected to the tree. For each step of the shared-tree construction, since the active shared-tree makes no further changes in its structure, the active shared-tree is independent from the cost contribution in future steps of the construction process. Hence, the active shared-tree can be viewed as a single entity and the first term in (3.1) can be replaced by the average minimum cost between the active shared-tree and the unconnected node. Thus, (3.1) can be simplified to

Chapter 3

$$SF(j) = \left[C(e_{k,j}) + \frac{1}{|B|} \sum_{b \in B} C_{MCP}(P_{j,b}) \right] \quad (3.2)$$

The final shared-tree obtained is the cost-efficient spanning shared-tree denoted as $\hat{T}$. For determining the first node to start the construction process, since initially no shared-tree is constructed, the first component of (3.2) is zero and the node with the least average minimum cost to all the other nodes is selected as the first node to start the construction process. Each node in the active shared-tree is required to know the up-to-date configuration of the active shared-tree so that the SF of those unconnected nodes can be computed for further tree construction. Hence, whenever a new node is connected to the shared-tree, a control packet which stores the information of the new connection is broadcast from the new node to all other nodes in the active shared-tree. The overheads required are the total number of control packets required to complete the shared-tree construction.

Figure 3-2: A random network of 8 nodes.

Chapter 3

Table 3-1: MCPs of the network shown in figure 3.2 using Dijkstra algorithm.

A→D, 1 A→G, 2 A→B, 3 A→D→C, 3 A→D→E, 4 A→G→F, 4 A→G→H, 4	B→E, 2 B→C, 2 B→A, 3 B→E→H, 3 B→C→D, 4 B→C→F, 5 B→A→G, 5	C→B, 2 C→D, 2 C→F, 3 C→D→A, 3 C→B→E, 4 C→F→G, 5 C→F→H, 5	D→A, 1 D→C, 2 D→E, 3 D→A→G, 3 D→C→B, 4 D→E→H, 4 D→C→F, 5
E→H, 1 E→B, 2 E→D, 3 E→H→G, 3 E→B→C, 4 E→H→F, 4 E→D→A, 4	F→G, 2 F→H, 3 F→C, 3 F→G→A, 4 F→H→E, 4 F→C→B, 5 F→C→D, 5	G→F, 2 G→H, 2 G→A, 2 G→H→E, 3 G→A→D, 3 G→A→B, 5 G→F→C, 3	H→E, 1 H→G, 2 H→F, 3 H→E→B, 3 H→G→A, 4 H→E→D, 5 H→E→B→C, 5

Figure 3-3: Phase 1 to 4 in stage 1 of MGST-T / DC-MGST-T.

Chapter 3

Figure 3-2 shows a random network of 8 nodes with the symmetric edge cost labeled beside each edge. The minimum cost paths (MCPs) between any two nodes in the random network are computed using Dijkstra's algorithm [Dijk59] and listed in table 3-1. The numerical value beside each path in table 3-1 indicates the cost of the path.

An example is shown in figure 3-3 for computing the spanning cost-efficient shared-tree from the network shown in figure 3-2. The value shown beside each node in figure 3-3 indicates the SF value to add that node to the active shared-tree. For the first phase shown in figure 3-3(a), each node computes their SF value using equation (3.2) without the first component. Node G is found to have the least SF value of 2.86. Hence, it is selected as the first node to add to the shared-tree. Note that at this stage the active shared-tree (shared-tree constructed so far) comprises of only node G. For the second phase which is illustrated in figure 3-3(b), since only node F, A and H are neighbors to the active shared-tree, nodes F, A and H compute the SF to yield some finite values while the rest of nodes that are not part of the active shared-tree are given an infinity SF value. Since node A has the least SF, it is selected next to be connected to the active shared-tree as shown in figure 3-3(b). If there is a tie in the SF value, the one with the least connection link cost will be selected to be connected to the shared-tree. This process is repeated for subsequent phases as shown in figure 3-3(c) to figure 3-3(d) and in figure 3-4(a) to figure 3-4(d) until all the nodes are connected to the spanning shared-tree shown in figure 3-4(d).

Chapter 3

Figure 3-4: Phase 5 to 8 in stage 1 of MGST-T / DC-MGST-T.

3.3.2.2 Stage 2: Resultant Tree Construction

For MGST, the resultant tree of an incoming multicast session is constructed by pruning the unnecessary edges and nodes in the shared-tree obtained from stage 1. Each node is assumed to contain the information of the physical graph and the shared-tree topologies. This makes sense because knowledge of the physical graph topology will allow replacement paths to be found if any fault in the existing delivery tree takes place; and knowledge of the shared-tree topology will allow data packets to be sent over the tree without incurring extra overheads. The only disadvantage of this algorithm is that the volume of information contained in each node may be too large. However, this is not a major problem since the shared-tree is constructed from the physical graph,

Chapter 3

only information on the physical topology information is required in each node with some flags to indicate the shared-tree edges. The methodology for efficient information storage per node is outside the scope of this thesis. During the resultant tree construction, the control plane will check the status of each node starting from the source node in the top to down and left to right fashion if it requires data to be transmitted out from it via the shared-tree. We refer to the resultant tree constructed so far at a particular step as the active resultant tree. When each node is checked, the sub-tree rooted at the node will be checked for any receiver nodes. If none exists, an *End-Flag* will be placed in that node indicating the end of data transmission in that node. Thus, after all the nodes in the shared-tree are checked, a resultant tree is constructed and data will be transmitted from the source to all the receiver nodes via the resultant tree. Since no data will be transmitted after those nodes with an *End-Flag*, the edges and nodes after those nodes with an *End-Flag* are not included in the resultant tree and are considered pruned.

However, for DC-MGST, since the shared-tree is constructed based on cost efficiency and not delay residue efficiency, some receiver nodes will not be able to receive data within the specified delay-constraint via the resultant tree. Hence, it should be noted that the resultant tree obtained in stage 2 for DC-MGST may not be able to connect all the receiver nodes of the multicast session. The following is a proof to show that the resultant tree obtained in stage 2 for DC-MGST connects to all receiver nodes if and only if the specified delay-constraint is always greater than the maximum delay between any two nodes in the shared-tree.

Chapter 3

Lemma 3-1: For delay-constraint scenarios, the conditions for connecting a node n from source s in the resultant tree $\vec{T}_s$ constructed from the shared-tree $\hat{T}$ are:-

$$(a) \ n \in Y \text{ or } n \in P_{\hat{T}}(s, y), \forall y \in Y.$$

$$(b) \ D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}.$$

Proof: Condition (a) states that node n needs to be connected if it is a receiver node or is part of the path which connects to a receiver node from the source. During the construction of the resultant tree in stage 2, the source node in the shared-tree transmits the data to its descendent nodes and will stop transmitting the data at those nodes with an *End-Flag*. Condition (b) states that data will only be transmitted to node n in the resultant tree if the delay to node n from the source node is less than the specified delay-constraint. Besides checking if there are any receivers in its descendent nodes, each node will also check the accumulated delay from the source node. If the delay equates or exceeds the delay-constraint Δ_{ϕ_j} for the multicast group from which the multicast session is initiated (i.e. $D(P_{\hat{T}}(s, n)) \geq \Delta_{\phi_j}$), then no data will be transmitted.

Lemma 3-2: If $D(P_{\hat{T}}(s, y)) = \max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\}$ and $\Delta_{\phi_j} > D(P_{\hat{T}}(s, y))$, then

$$D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}.$$

Proof: Assuming in the worst case that the delay between the source and a receiver node in $\hat{T}$ is the maximum delay between any two nodes in $\hat{T}$ (i.e.

$$D(P_{\hat{T}}(s, y)) = \max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\},$$

then: - $\therefore D(P_{\hat{T}}(s, n)) < D(P_{\hat{T}}(s, y)), \forall n \in \hat{T}, y \neq n,$

Chapter 3

$$\therefore \Delta_{\phi_j} > D(P_{\hat{T}}(s, y)) \Rightarrow D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}.$$

Lemma 3-3: As long as the delay from the source to each node in $\hat{T}$ is smaller than Δ_{ϕ_j} (i.e. $D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}$), $\vec{T}_S$ constructed from the shared-tree $\hat{T}$ can span to all receiver nodes from the source without delay-constraint violation.

Proof: This can be easily proved by inheriting the characteristics of node n in lemma 3-1 and extend them to all nodes in $\hat{T}$. Therefore, if $D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}$, according to lemma 3-1, the conditions to connect and hence span from the source node to each receiver node $y \in Y$ such that $n \in Y$ or $n \in P_{\hat{T}}(s, y), \forall y \in Y$ are fulfilled. Hence, $\vec{T}_S$ is able to span to all receiver nodes from the source node without any delay-constraint violation if $D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}$.

Lemma 3-4: As long as the maximum delay between any two nodes in $\hat{T}$ is smaller than Δ_{ϕ_j} (i.e. $\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} < \Delta_{\phi_j}$), $\vec{T}_S$ constructed from the shared-tree $\hat{T}$ can span to all the receiver nodes from the source node without delay-constraint violation.

Proof: The worst case occurs when the delay between the source and one of the receiver nodes is the maximum (i.e. $D(P_{\hat{T}}(s, y)) = \max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\}$). Since

$$\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} < \Delta_{\phi_j} \Rightarrow D(P_{\hat{T}}(s, y)) < \Delta_{\phi_j} \because s, y \in \hat{T}, \text{ therefore, for the}$$

worst case scenario, according to lemma 3-2, $D(P_{\hat{T}}(s, n)) < \Delta_{\phi_j}, \forall n \in \hat{T}$ when

Chapter 3

$\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} < \Delta_{\phi_j}$. Thus, according to lemma 3-3, the resultant tree can

always span to all its receiver nodes without delay-constraint violation when

$$\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} < \Delta_{\phi_j}.$$

Figure 3-5 shows the tree constructed for a multicast session with source node at C and destination nodes at E, G, F, A and B by MGST-T / DC-MGST-T, MCT and CBT. The CBT in figure 3-5(c) has its core node located at G. G is selected as the core for CBT because it has the least average minimum cost to the other receiver nodes. It can be observed that MGST-T / DC-MGST-T and MCT have tree cost of 12 units while CBT has tree cost of 13 units. Although it is fairly simple to obtain the minimum cost tree through pruning of the shared-tree in MGST-T / DC-MGST-T, one disadvantage is that the minimum cost tree may not be obtained for all cases. For example, if only a handful of nodes are members, then the tree obtained via pruning from the spanning shared-tree may consist of a large number of non-leaf nodes which are not necessary (redundant nodes) to connect the receiver nodes. This leads to redundant cost in the tree and increases the tree cost instead of minimizing it. Figure 3-6 shows one such example where a multicast session has the source node at C and receiver nodes at B and F. As node B and node F are leaf-nodes in the shared-tree obtained from MGST-T / DC-MGST-T with respect to the source C shown in figure 3-6(a), no edges can be pruned and the total tree cost is 12 units. This is in big contrast to the tree obtained from MCT and CBT, which has a tree cost of 5 units each. To resolve this, we perform “trace back” to MGST-T / DC-MGST-T.

Chapter 3

Figure 3-5: For a multicast session with C as the source node, and {E, G, F, A, B} as the receiver nodes, the tree constructed by (a) MGST-T /DC-MGST-T, (b) MCT and (c) CBT.

Figure 3-6: For a multicast session with C as the source node, and {F, B} as the receiver nodes, the tree constructed by (a) MGST-T /DC-MGST-T, (b) MCT and (c) CBT.

Chapter 3

3.3.2.3 Stage 2 Improvement: “Trace Back” on Resultant Tree Construction

After the construction of the resultant tree in stage 2, the control plane will now perform “trace back” for the immediate child node of each node n in the resultant tree starting from the source node from top down fashion to reduce the resultant tree cost. We refer to our proposed algorithms MGST and DC-MGST with the “trace back” feature as MGST-T and DC-MGST-T respectively. “Trace back” determines if the resultant tree path to each immediate child node w of each node n in the resultant tree can be replaced by a cheaper direct minimum cost link ingressed at one of the parent nodes of n in the resultant tree. We refer to the resultant tree path considered to be replaced as RTP and the direct minimum cost link as RL. The direct minimum cost link between the two nodes here refers to the least cost link between two nodes in the physical graph. Node n itself cannot be ingress to the RL to its immediate child node w since the link between n and w is part of the RTP. Hence, “trace back” on a node n requires finding the RL between w and among the parent nodes of n in the resultant tree, and finding the RTP. The RTP is egressed at node w and ingressed at the ‘anchor’ node nearest to w . An ‘anchor node’ is the ingress node of the replaced path which will not create any disjoint trees when replaced. Hence, an anchor node can be the source node, ingress of the replacement link, a receiver node, or a node with out-degree greater than one. If the replacement link has a lower cost than the replaced path, then the replacement link to w will replace the replaced path. The idea is to remove any redundant nodes in the RTP. For delay-constraint scenarios, the least cost RL

Chapter 3

will replace the RTP if and only if they do not cause violation in the delay-constraint for any of the receiver nodes of the multicast session.

Figure 3-7: Performing “Trace Back” in stage 2 of MGST-T / DC-MGST-T for multicast session with source at node C to destination node B and F.

The “trace back” process is not as straight-forward if a node n in the resultant tree has more than one immediate child node w . This is because three scenarios are possible. The first scenario is that no RLs can be found for any of the immediate child nodes. Hence, no replacement is required. The second scenario is that at least one of the immediate child nodes cannot find its RL. We group immediate child nodes unable and able to compute their RL in set $G1$ and $G2$ respectively. This means that no replacement is required for at least one of the immediate child nodes of node n . Hence, the RTP for those immediate child nodes in set $G2$ can only be ingressed at node n since node n is used as a relay node for at least one of the immediate child node in set $G1$ and hence is the

Chapter 3

nearest ‘anchor’ node to the immediate child nodes. Since all the RTPs are ingressed at node n , the RTPs can be expressed as a sub-tree. We shall henceforth refer the replacement portion of the resultant tree at each phase of the “trace back” process as RTT where RTT may consist of only one RTP or a sub-tree. To determine the RL for more than one immediate child nodes of node n , the RL with the least cost to node b in set $G2$ is first selected. The next least cost RL ingressed at one of the parent nodes of node n or node b is then selected. This continues until the RL to all the immediate child nodes in set $G2$ are determined. If the combined cost of all RLs is less than the cost of RTT, the RLs will replace the RTT. The third scenario is that the RLs for all the immediate child nodes can be computed. If node n is an ‘anchor’ node, then RTT must be rooted at node n to ensure that no dis-joint sub-trees are created, and the procedure to find the RLs for the immediate child nodes of n is the same as that in the second scenario. However, if node n is not an “anchor” node, the RTT will include the resultant tree path ingressed at the nearest “anchor” node to n and egressed at node n , and all the resultant tree links ingressed at node n and egressed at the immediate child nodes of n . The procedure to find the RLs for the immediate child nodes of n is the same as in the second scenario.

The random network in figure 3-2 is used for the example shown in figure 3-7. The shared-tree is obtained in stage 1 of MGST-T / DC-MGST-T shown in figure 3-4(d). After stage 2 of MGST-T / DC-MGST-T, the resultant tree for the multicast session is marked in bold lines and shaded nodes shown in figure 3-7(a). Starting from node C, we perform “trace back” to the immediate child

Chapter 3

nodes of each node in the resultant-tree in a top-down fashion. Since there is no RL ingressed at the parent nodes of node D (consist of node C) to node D, no replacement takes place for the immediate child node of C. This is the same for the immediate child node of D and A. The next node to consider is node G. Node F and node H are the immediate child nodes of G. Link C-F is the RL to F while there is no RL between H and its parent nodes. Hence, the least cost RL, C-F, is first selected. Now that link C-F is a possible link to replace the RTP to F, node F can be considered as a parent node to H, and the next least cost RL to H is link F-H. With both F and H with RL not ingressed at G, the RTT includes link G-F, link G-H, and the path from G to source node C since A and D are not anchor nodes. Since the combined cost of link C-F and link F-H is 6 units and the cost of the RTT is higher at 9 units, the RLs replace the RTT shown in figure 3-7(b). Node H requires no replacement for its immediate child node E since no RL can be computed for E. At node E, its immediate child node B has a RL ingressed at C. Since node F is a receiver node, RTP is ingressed at node F and egressed at node B. Since the cost of the RL has a cost smaller than the RTP, the RL replaces the RTP shown in figure 3-7(d). The final tree cost in figure 3-7(d) after “trace back” is only 5 units. This is a significant improvement compared to 12 units before the “trace back” shown in figure 3-6(a).

3.3.2.4 Stage 3: Connection of all receiver nodes not connected in Stage 2

For DC-MGST-T, stages 1 and 2 may not be sufficient to construct the delivery tree for each multicast session. As shown from lemma 3-4, the resultant tree constructed in stage 2 will span to all the receiver nodes if

Chapter 3

$\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} < \Delta_{\phi_j}$. Hence, if $\max_{\forall a, b \in \hat{T}} \{D(P_{\hat{T}}(a, b))\} \geq \Delta_{\phi_j}$, there is no guarantee that all receiver nodes can be connected via the resultant tree to receive data within the specified delay-constraint. We refer to those receivers not connected to the resultant tree after stage 2 as unconnected receivers. Minimum delay paths from the source to the unconnected receivers are appended to the resultant tree to connect all the receiver nodes. The reason for using minimum delay paths is because if the minimum delay path is unable to connect a receiver node due to the strict delay-constraint, then there will not be any solution to connect the node. The following is a proof to show that all the receiver nodes can be connected if and only if the specified delay-constraint is always greater than the maximum delay of all the minimum delay paths between any two nodes in the network. To avoid confusion, we refer to the final tree T_s as the tree comprises of the resultant tree constructed in stage 2 and the minimum delay paths appended to the resultant tree for those unconnected receiver nodes after stage 2. The resultant tree is the final tree if all the receivers can be connected in the shared-tree to receive data within the specified delay-constraint.

Lemma 3-5: Final tree T_s will connect all the receiver nodes if and only if

$\Delta_{\phi_j} > \max_{\forall a, b \in N} \{D(U(a, b))\}$, where U is the set of all minimum delay paths between any two nodes.

Proof: Consider the worst case when a minimum delay path between the source and a receiver node has the maximum delay among all the minimum delay paths. If the specified delay-constraint is greater than the delay of this minimum

Chapter 3

delay path, then the smaller delay of all the minimum delay paths from the source to the rest of the receiver nodes will be smaller than the specified delay-constraint and thus have no problem in satisfying the specified delay-constraint and can be connected either by the resultant tree or the minimum delay paths.

The minimum delay path with the least delay allowance (i.e. difference between the path delay and delay-constraint) to an unconnected receiver node will be connected first. This is to allow more nodes to be connected via the minimum delay paths even in the event of strict delay-constraint. When a minimum delay path between the source and a receiver node is appended to the resultant tree, it is likely that some of the links in the path already exist in the resultant tree. Hence, duplication of edges and loops may exist when a minimum delay path is appended to the resultant tree. For duplication of edges, only those links in the path not in the resultant tree are appended to the resultant tree. To remove looping at node n , we remove the existing edge to node n in the resultant tree and append the minimum delay path's edge to the same node to remove any loops created at node n . This is possible because the delay from the source to node n via the newly appended edge is always smaller than or equal to via the existing edge since any two nodes in the minimum delay path has the minimum delay possible. Thus, if the delay from the source node to node n via the existing edge is within the delay-constraint, then the delay from source to node n via the new appended edge is also within the delay-constraint.

Chapter 3

Figure 3-8: An example of appending minimum delay path to the resultant tree.

Figure 3-9: The final tree after removal of duplicate edges and loop.

Figure 3-8 shows an example of appending a minimum delay path to the resultant tree. The resultant tree comprises of nodes A, B, C, D, F and G while the minimum delay path comprises of nodes A, B, D, E, F, G and H. The source node is node A while the receiver nodes are node C, G and H. It can be observed that edges A-B, B-D and F-G have duplicated edges due to both the resultant tree and the minimum delay path containing these edges. Note that looping will occur at node F when edge E-F and D-F are directed towards node F. Thus, for resource efficiency, the duplicated edges are removed. Since the delay from A to F via the minimum delay path has the minimum delay possible, it is less than the delay from A to F via the resultant tree which is delay constrained, and hence will not cause node G to violate the delay-constraint.

Chapter 3

Thus, edge D-F of the resultant tree is removed to eliminate the loop at F. Figure 3-9 shows the final tree after the removal of duplicated edges and loops.

The minimum delay paths appended to the resultant tree may not be cost-efficient since they have minimum delays and may have excessive delay allowance which is not needed. Thus, a systematic improvement process is required to cost-optimize those minimum delay paths appended to the resultant tree without creating any delay-constraint violation in any of the receiver nodes. To maximize cost improvement, minimum cost paths are used to replace parts of each minimum delay path appended to the resultant tree iteratively. The minimum cost paths (MCPs) and minimum delay paths (MDPs) between any two nodes can always be constructed during the initialization stage using Prim's algorithm [Prim57] before any arrival of multicast sessions because these paths are independent of the traffic patterns. Hence, the construction of MDPs and MCPs do not increase the required overhead and the tree's ready time for each incoming multicast session. The objective is to improve the cost between nodes in the appended minimum delay path without causing any delay-constraint violations. However, the number of candidate minimum cost paths to improve the minimum delay paths will be enormous if all the possible paths are considered. To reduce the search space, the egress node of the appended minimum delay path is "pivoted" in such a way that only the minimum cost paths between this "pivoted" node and its ancestor nodes are considered to improve the minimum delay path.

Chapter 3

The cost reduction factor, CR , for each candidate minimum cost path is first computed by taking the difference between the cost of the minimum cost path and the cost of the appended minimum delay path (i.e. $CR_{x,y} = C_{MCP}(P_{x,y}) - C_{MDP}(P_{x,y})$). The one with the largest CR will be used to replace the corresponding path in the appended minimum delay path. Since the minimum cost path is used to replace certain parts of the appended minimum delay path, delay-constraint violation needs to be checked to ensure that the replacement does not cause any delay-constraint violation not only to the receiver node of the improved appended minimum delay path but also to the rest of the receiver nodes. For the receiver node of the improved appended minimum delay path, the delay-constraint is the constraint specified for the group which the multicast session is originated from and hence can be easily checked to see if the improved path violates this constraint. However, if the improved path intersects any of the nodes in the resultant tree, the delay to those nodes via the improved path cannot exceed a certain value which causes delay-constraint violations to their descendent receiver nodes. Thus, delay-constraint of these intersection nodes has to be computed to ensure that no delay-constraint violation occurs at their descendent receiver nodes. Since the delay-constraint of each intersection node k is defined as the minimum delay allowed from the source node to node k without causing any delay-constraint violation to any of the receiver nodes descendent to node k , the difference between the delay-constraint specified for the receiver nodes and the maximum delay from node k to its entire descendent nodes in the resultant tree is the new delay-constraint of node k . Take figure 3-8 for example, $\Delta_{\phi_j} - \max\{D(P_{\hat{T}}(F,G), D(P_{\hat{T}}(F,H))\} = \Delta_{\phi_j} - D(P_{\hat{T}}(F,H))$ is the delay-constraint

Chapter 3

for intersection node F. No delay-constraint exists for those nodes in the improved path which does not exist in the resultant tree. This is because they are used as transition nodes, and the delay-constraint will be satisfied at the receiving end of the path. Thus, the minimum cost path with the largest cost reduction factor and able to satisfy the delay-constraint of all nodes in its path is selected to replace the corresponding sub-path in the appended minimum delay path. Since the appended minimum delay path is now improved and changed, further improvement can be achieved by repeating the improvement process. The process stops when the largest cost reduction factor is zero. Thus, the entire improvement process improves all the appended minimum delay paths without causing any delay-constraint violation to all the receiver nodes.

3.4 Simulation Results and Discussion

3.4.1 Performance Metrics

3.4.1.1 Session Blocking Probability (SBP)

In this chapter, we consider the more practical blocking scenarios where some receiver nodes may not be able to receive the data within the specified delay-constraint. We define a multicast session as blocked as long as one of its receiver nodes is unable to satisfy the delay-constraint specified for the group which the multicast session is originated from. Session blocking probability is the fraction of the incoming multicast sessions blocked.

3.4.1.2 Cost Competitiveness (CC)

CC measures the cost-performance of a heuristic algorithm. For this chapter, the near optimal KMB heuristic is used as the benchmark. It is important to

Chapter 3

note that although KMB is able to achieve near optimal cost results and does not consider any delay-constraint, it is not impossible for delay constrained algorithms such as STAR and DC-MGST-T to achieve better cost results than KMB in very relax delay-constraint scenarios. This is because of the tree cost improvement stages implemented in both STAR and DC-MGST-T. Hence, the minimum value of CC here is not limited to 1. The expression of CC for an algorithm is given as: $CC = \frac{1}{|L|} \sum_{s \in L} \frac{C(T_s)}{C(Q_s)}$ where T_s and Q_s are the final delivery trees constructed by the algorithm and KMB heuristic respectively for session S . KMB focuses on minimizing cost and disregards any delay-constraint scenarios. Thus, KMB will give the least cost in most occasions and an algorithm with CC close to 1 means that it has a cost-performance almost the same as KMB, which is good.

3.4.1.3 Overheads required per unblocked session (K)

We define overheads to be the cost associated with the control packets required to activate and deactivate links in the process of constructing a delivery tree for each unblocked session. For stage 1 of MGST-T and DC-MGST-T, control packets are required to construct the single spanning shared-tree. However, for stage 2, no control packets are required since the control plane can be programmed to check each node in the shared-tree starting from the source node in a top-down and left-right fashion for any descendent receiver nodes, and plant an *End-Flag* in the node if none exists. In this way, data can be transmitted from the source to all the receiver nodes in the shared-tree by ending the data flow at all nodes with an *End-Flag*. For “trace back”, control packets are required to activate the RLs and deactivate those replaced RTPs.

Chapter 3

For stage 3 in DC-MGST-T, control packets are required for appending the cost-optimized minimum delay paths to the resultant tree obtained in stage 2. For KMB and STAR, control packets are required to append each ‘best’ path to the active tree for each step of the tree’s construction algorithm.

3.4.1.4 Tree’s Ready Time per unblocked session (T_i)

This metric measures the time to get the delivery tree for an unblocked session ready for transmitting data. It is also referred to as the tree’s construction time for each unblocked session. We are interested in the time lapse between the arrival of a multicast session request and its delivery tree beginning to transmit data to its receiver nodes. This is important as a small tree’s ready time will eliminate race conditions and better utilize resources.

3.4.2 Simulation Model

3.4.2.1 Random Graphs Generation

Randomly generated networks are used for the simulation to ensure that the results obtained are independent of any characteristic peculiar to a particular network topology. We use Waxman method proposed in [Wax88] to generate random networks with varying node sizes. Vertices are first placed randomly in a rectangular grid system by randomly generating the x and y coordinates of each vertices. The connectivity of the graph is then ensured by first constructing a spanning tree in the graph. This is carried out by connecting all distinct nodes to the active tree for each step of the spanning tree construction. The remaining edges are then added to the active tree if a randomly generated number for each edge is less than the probability function $P(u,v)$ based on the edge distance

Chapter 3

between u and v . The probability function is given by $P(u, v) = \beta e^{-d(u, v)/2\alpha N}$ where $d(u, v)$ is the Euclidean distance between u and v ; $N=|V|$ is the number of nodes in the graph; β and α are the tunable parameters where increasing β will increase the degree of each node, and increasing α will increase the number of connections between far-off nodes.

3.4.2.2 Simulation Parameters

Networks ranging from 20 to 100 nodes are randomly generated using the method described in *section 3.4.2.1*. The coordinate grid used extends from (0,0) to (200,200). β and α are set to achieve an average node degree of between 3 and 4. The networks generated are sparse networks containing edges of about 5% to 10% of the total possible edges. It was mentioned in [RMM99] that sparse networks with low average node degree are more representative of real networks and pose a tougher problem for finding the optimal delivery tree. Each edge cost is set in the range (0,10] and the edge delay is set proportional to its distance with the maximum delay set at 10. It should be noted that this maximum delay is set arbitrarily so that the delay-constraint can be assigned to simulate blocking scenarios where certain nodes may violate the delay-constraint. The delay-constraint is either uniformly or non-uniformly assigned. For uniform setting, all multicast groups will have the same delay-constraint (i.e. $DC = \Delta_{\phi} = \Delta_{\phi_0} = \Delta_{\phi_1} = \dots = \Delta_{\phi_z}$). For non-uniform delay-constraint, multicast groups will vary in their delay-constraint. We assume that all members within the same group will have the same delay-constraint since each group is usually assigned to an application which requires the same delay

Chapter 3

requirement for all its members for desirable service. It is stated in lemma 3-5 that DC-MGST-T is able to connect all the receiver nodes if the delay-constraint is greater than the maximum delay among all the minimum delay paths. Hence, no blocking will occur for as long as the delay-constraint is set to be greater than the maximum delay among all the minimum delay paths. We shall denote this maximum delay as the minimum delay-constraint ($DC_{\min}$) that can be assigned to achieve non-blocking for DC-MGST-T. In order to simulate the blocking and non-blocking scenarios, we use a set of delay-constraint (DC) obtained from repeated experiments to simulate strict, average and relax DC conditions for all our simulations. After numerous experiments, the set of DC ideal for our simulation is found to range from 15 to 60 units with an interval of 5 units. Hence, for uniform delay-constraint scenarios, we perform simulations for DC=15 to 60 units with an interval of 5 units. The lower bound delay constraint depends on the multicast applications. For example, a video live-streaming application may require the data to be sent within a stricter delay constraint compared to a video-on-demand application. However, it is important to note that the constraint cannot be too small as it will be impossible on some occasions to find a suitable route to deliver the data to a receiver within the delay-constraint and hence blocking occurs. Similar to most of the existing related studies, propagation delay is assumed as the major delay contributor although other delays such as node's processing delay etc can be added in as well. Note that for uniform delay-constraint scenarios, we assume that all multicast groups have the same DC requirement. For non-uniform delay-constraint scenarios, the DC requirement for each multicast group is randomly generated within a range of a mid-point specified. Since the mid-

Chapter 3

point of the set of DC settings used for uniform delay-constraint scenarios is around 35 units, we set the mid-point DC for non-uniform delay-constraint scenarios as 35 units, and a range with 35 units as the mid-point whereby each multicast group's DC requirement is randomly generated within this range. For example, if a range of X units is set, the DC requirement of each multicast group Δ_{ϕ_j} is randomly generated in the range $[35-X/2, 35+X/2]$.

The multicast group size $z = |\phi|$ is randomly set to 100 with the member size of each group uniformly set at 30%, 50% or 70% of the network size (i.e. $|M| = |M_{\phi_1}| = \dots = |M_{\phi_z}|$ and $|M| = 0.3|V|, 0.5|V|, 0.7|V|$). The idea for this setting is to investigate the behavior of the algorithms for low-tier membership size (30%), middle-tier membership size (50%) and high-tier membership size (70%). For each simulation, each of the 100 multicast sessions are randomly selected from among the multicast groups generated.

3.4.3 Discussion of Results

We compare our proposed DC-MGST-T with the STAR heuristic proposed in [LMZ04], which is known to be the most efficient among all the existing algorithms. For uniform delay-constraint scenarios, we tabulate the difference in results between STAR and DC-MGST-T. It should be noted that for very large DC ($DC > 40$), DC-MGST-T behaves like MGST-T since very large DC is as good as no delay-constraint imposed. Hence, we only perform simulations for DC-MGST-T in this thesis and observe the results for strict, average and large DC settings.

3.4.3.1 Comparison between DC-MGST-T and STAR in terms of SBP

Table 3-2 and table 3-3 show the session blocking probability (SBP) performance of STAR and DC-MGST-T respectively for networks of different sizes with varying uniform delay-constraints (from 15 to 60) for all the multicast groups. It is observed that DC-MGST-T has a much smaller SBP than STAR for the same delay-constraint and networks of the same size. For very strict delay-constraint (i.e. $DC = 15$), both STAR and DC-MGST-T have very high SBP. This is mainly because the delay-constraint is too strict for any solution to exist. When the delay-constraint is relaxed (i.e. DC increases), the SBPs for both STAR and DC-MGST-T decrease. This is expected because when the delay-constraint is relaxed, data will be able to deliver to more receiver nodes within the delay-constraint, and hence decrease the overall SBP. DC-MGST-T is observed to have a greater decrease in SBP as DC increases compared to STAR. This is due to the use of minimum delay paths (MDPs) to deliver traffic to those receivers not reachable via the shared-tree. In addition, it is also observed that for some DC values, DC-MGST-T is able to achieve zero SBP while STAR has $SBP > 0$. This shows that DC-MGST-T is more delay tolerant than STAR. This means DC-MGST-T is able to deliver traffic to more receiver nodes than STAR for the same DC specified. For very relax (large) delay-constraint, both DC-MGST-T and STAR will eventually achieve zero SBP since the scenario with a very relax delay-constraint can be modeled as a scenario with no constraint and hence will not inflict any blocking due to delay-constraint violation.

Chapter 3

Table 3-2: SBP for STAR (Membership size: 50% of network size).

SBP_{STAR}	$ V =20$	30	40	50	60	70	80	90	100
DC=15	0.82	0.76	0.83	0.9	0.97	0.93	1	1	0.99
20	0.47	0.45	0.54	0.79	0.8	0.76	0.96	0.95	0.96
25	0.24	0.21	0.3	0.42	0.65	0.57	0.92	0.96	0.92
30	0.12	0.13	0.07	0.31	0.37	0.43	0.72	0.82	0.64
35	0.04	0.02	0.04	0.18	0.21	0.33	0.53	0.58	0.58
40	0	0	0.03	0.09	0.17	0.21	0.46	0.51	0.3
45	0.02	0	0	0.07	0.13	0.19	0.29	0.32	0.19
50	0.01	0	0	0.02	0.05	0.11	0.17	0.2	0.14
55	0	0	0	0.02	0.01	0.03	0.17	0.17	0.06
60	0	0	0	0	0	0	0.09	0.06	0.03

Table 3-3: SBP for DC-MGST-T (Membership size: 50% of network size).

$SBP_{DC-MGST-T}$	$ V =20$	30	40	50	60	70	80	90	100
DC=15	0.7	0.36	0.27	0.56	0.69	0.55	0.94	0.95	0.87
20	0.29	0.01	0	0.08	0.4	0.05	0.46	0.65	0.38
25	0.07	0	0	0	0.11	0.01	0.12	0.23	0.05
30	0	0	0	0	0	0	0	0.03	0.01
35	0	0	0	0	0	0	0	0.02	0
40	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0

Table 3-4: $SBP_{STAR} - SBP_{DC-MGST-T}$ (Membership size: 50% of network size).

$SBP_{STAR} - SBP_{DC-MGST-T}$	$ V =20$	30	40	50	60	70	80	90	100
DC=15	0.12	0.4	0.56	0.34	0.28	0.38	0.06	0.05	0.12
20	0.18	0.44	0.54	0.71	0.4	0.71	0.5	0.3	0.58
25	0.17	0.21	0.3	0.42	0.54	0.56	0.8	0.73	0.87
30	0.12	0.13	0.07	0.31	0.37	0.43	0.72	0.79	0.63
35	0.04	0.02	0.04	0.18	0.21	0.33	0.53	0.56	0.58
40	0	0	0.03	0.09	0.17	0.21	0.46	0.51	0.3
45	0.02	0	0	0.07	0.13	0.19	0.29	0.32	0.19
50	0.01	0	0	0.02	0.05	0.11	0.17	0.2	0.14
55	0	0	0	0.02	0.01	0.03	0.17	0.17	0.06
60	0	0	0	0	0	0	0.09	0.06	0.03

Table 3-4 shows the SBP difference between STAR and DC-MGST-T (i.e.

$SBP_{STAR} - SBP_{DC-MGST-T}$). All the values in the tables are greater or equal to 0

Chapter 3

indicating that DC-MGST-T performs as well or better than STAR for all occasions. It is observed that the majority of the values in table 3-4 is greater than 0, indicating DC-MGST-T has better blocking performance than STAR. As mentioned before, this is because MDPs will connect those receiver nodes not able to satisfy the delay-constraint using the spanning shared-tree while STAR focuses on selecting the least cost delay constrained path each time to connect a node during tree construction. Hence, for each step of the tree's construction process using STAR, the delay residue of the active tree may not be enough to connect other receiver nodes within the specified delay-constraint, leading to a higher SBP. The values marked in bold represent the maximum difference between STAR and DC-MGST-T in the range from DC=15 to 60 units for each network size. The maximum difference in SBP for each network size occurs at $DC < 30$. This again reflects the greater delay-tolerance of DC-MGST-T over STAR.

3.4.3.2 Comparison between DC-MGST-T and STAR in terms of CC

Table 3-5 shows the difference in CC between STAR and DC-MGST-T ($CC_{STAR} - CC_{DC-MGST-T}$). A negative value indicates that STAR has a lower CC than DC-MGST-T and hence is better. It is observed that most of the values in table 3-5 are less than 0. This is expected due to two reasons. The first reason is because DC-MGST-T constructs the delivery tree through pruning of a common shared-tree while STAR constructs each delivery tree from scratch by selecting the least cost paths to append to the tree during the tree's construction. The second reason is that DC-MGST-T employs MDPs to connect those receiver nodes unable to satisfy the delay-constraint via the shared-tree. MDPs

Chapter 3

have minimum delay but not cost. Hence, some of the MDPs employed may have high cost and thus increases the overall CC for DC-MGST-T. However, it can be seen that the cost difference of STAR over DC-MGST-T is at most 0.44 for all DC and different network sizes ranging from 20 to 100 nodes. This is due to the cost-improvement features implemented in DC-MGST-T to improve the cost efficiency of trees to the maximum.

Table 3-5: $CC_{STAR} - CC_{DC-MGST-T}$ (Membership size: 50% of network size).

$CC_{STAR} - CC_{DC-MGST-T}$	$ V =20$	30	40	50	60	70	80	90	100
DC=15	-0.107	-0.330	-0.358	-0.188	-0.305	-0.444	N.A.	N.A.	-0.188
20	0.074	-0.253	-0.383	-0.289	-0.221	-0.259	-0.139	-0.122	-0.173
25	0.025	-0.286	-0.239	-0.060	-0.160	-0.211	-0.238	-0.095	-0.063
30	-0.036	-0.153	-0.104	-0.026	-0.051	-0.186	-0.022	-0.085	-0.074
35	-0.088	-0.111	-0.127	0.021	-0.027	-0.088	-0.095	-0.125	-0.061
40	-0.088	-0.060	-0.007	0.002	-0.034	-0.009	-0.143	-0.145	-0.024
45	-0.088	-0.065	-0.004	-0.008	-0.045	-0.038	-0.107	-0.100	-0.021
50	-0.088	-0.009	-0.001	-0.004	-0.041	-0.015	-0.152	-0.075	-0.018
55	-0.088	-0.010	-0.001	-0.004	-0.038	-0.018	-0.138	-0.066	-0.014
60	-0.088	-0.010	-0.001	-0.004	-0.038	-0.021	-0.131	-0.053	-0.014

3.4.3.3 Comparison between DC-MGST-T and STAR in terms of K

Table 3-6 shows the difference in overheads required per unblocked session ($K_{STAR} - K_{DC-MGST-T}$) for STAR and DC-MGST-T. The values in bold indicate maximum difference in K for each network size. A positive value indicates that the overheads required by DC-MGST-T is lower than that by STAR and hence is better. It can be seen that the values in table 3-6 for all DC and network sizes is greater than 0, indicating that DC-MGST-T requires smaller number of overheads per unblocked session than STAR. This is because while STAR has

Chapter 3

to construct each delivery tree from scratch and hence requires overheads for each construction, DC-MGST-T only incurs overheads for constructing the common shared-tree and the cost-improved MDPs for connecting those receivers not able to satisfy the delay-constraint via the shared-tree. It can be observed that the positive difference in overheads required between STAR and DC-MGST-T for different network sizes of 20 to 100 nodes increases significantly as DC increases. This indicates that DC-MGST-T requires much lesser overheads than STAR and this difference in overheads increases as DC increases. The main reason is because more receiver nodes are able to receive data within the specified delay-constraint (DC) via the shared-tree for DC-MGST-T when the DC is relaxed, and hence requires smaller overheads to connect the number of receiver nodes via cost-improved MDPs in stage 3 of DC-MGST-T. As the values in table 3-6 are in the scale of $\log_{10}$, it is important to note that the extra overheads required for STAR is significant compared to DC-MGST-T.

Table 3-7 shows the average tree's ready time per unblocked session for STAR and DC-MGST-T. A positive value indicates that the average tree's ready time per unblocked session for DC-MGST-T is less than for STAR and hence is better. It is observed that for smaller and medium size networks (≤ 70 nodes), the difference in AT_i between DC-MGST-T and STAR is small. This is because there is a smaller number of cost-improvement iterations for STAR due to the smaller number of sub-trees rooted at trim node to be re-constructed. Hence, the time required for STAR to get the tree ready for each session decreases to a value near or on some occasion lesser than DC-MGST-T.

Chapter 3

However, for larger-sized networks (≥ 80 nodes), it is observed that DC-MGST-T has much smaller AT_i than STAR, and this difference increases with network size. This is because more sub-trees rooted at trim root are required to be re-constructed leading to longer tree's ready time. The positive values in table 3-7 for most of the DC settings and network sizes indicates that DC-MGST-T has compensated the time taken to perform cost-improvement processes like “trace back” and cost optimization of MDPs in stages 2 and 3 respectively with the time saved for not needed to construct each delivery tree from scratch.

Table 3-6: $K_{STAR} - K_{DC-MGST-T}$ ($\log_{10}$) (Membership size: 50% of network size).

$K_{STAR} - K_{DC-MGST-T}$ ($\log_{10}$)	$ V =20$	30	40	50	60	70	80	90	100
DC=15	0.388	0.412	0.479	0.452	0.465	0.582	N.A.	N.A.	0.385
20	0.753	0.504	0.556	0.620	0.621	0.699	0.541	0.586	0.761
25	0.949	0.632	0.687	0.729	0.752	0.781	0.642	0.707	0.899
30	1.114	0.838	0.850	0.889	0.901	0.833	0.702	0.813	1.035
35	1.256	1.024	1.007	1.049	1.039	0.942	0.782	0.915	1.140
40	1.254	1.134	1.138	1.186	1.139	1.084	0.882	1.023	1.246
45	1.254	1.240	1.281	1.250	1.247	1.256	0.984	1.155	1.315
50	1.254	1.262	1.354	1.285	1.326	1.362	1.068	1.280	1.363
55	1.254	1.262	1.386	1.293	1.355	1.392	1.139	1.356	1.387
60	1.254	1.262	1.386	1.297	1.359	1.394	1.198	1.411	1.390

Table 3-7: $ATi_{STAR} - ATi_{DC-MGST-T}$ (Membership size: 50% of network size).

$ATi_{STAR} - ATi_{DC-MGST-T}$	$ V =20$	30	40	50	60	70	80	90	100
DC=15	-0.005	0.005	-0.003	0.046	0.056	0.477	N.A.	N.A.	0.690
20	-0.001	0.000	0.007	0.008	0.144	0.169	0.331	0.509	0.762
25	0.001	-0.008	0.003	0.032	0.111	0.247	0.699	0.598	0.688
30	0.001	-0.007	-0.005	0.020	0.062	0.184	0.368	0.522	0.805
35	0.000	-0.002	0.006	0.018	0.046	0.170	0.337	0.582	0.782
40	0.000	0.001	0.005	0.032	0.051	0.151	0.342	0.512	0.842
45	-0.001	0.001	0.010	0.039	0.078	0.175	0.309	0.546	0.815

Chapter 3

50	-0.001	0.000	0.013	0.040	0.087	0.203	0.311	0.536	0.855
55	0.001	0.003	0.018	0.046	0.098	0.217	0.326	0.576	0.932
60	0.001	0.002	0.016	0.048	0.099	0.216	0.328	0.635	0.948

From table 3-2 to table 3-7, we can conclude that DC-MGST-T achieves much lower SBP, lower average tree's ready time and smaller overheads required per unblocked session at the expense of slightly higher CC compared to STAR. The advantages of significantly lower SBP, average tree's ready time and overheads required per unblocked session for DC-MGST-T far outweighs the slightly higher CC incurred compared to STAR. For example, DC-MGST-T can achieve up to 87% lower in SBP with only 0.063 higher in CC compared to STAR (for DC =25, $|V|$ =100). In addition, DC-MGST-T requires up to 1.39 lower overheads in the scale of $\log_{10}$ with only 0.014 higher in CC compared to STAR (for DC=60, $|V|$ =100). Also, DC-MGST-T requires up to 0.948 lower average tree's ready time with only 0.063 higher in CC compared to STAR (for DC=25, $|V|$ =100). Even for the scenario where STAR has the lowest CC of 0.444 over DC-MGST-T when DC =15 and $|V|$ =70, the SBP achieved by DC-MGST-T is 38% lower than STAR; the overheads required per unblocked session for DC-MGST-T is 0.582 lower than STAR in the scale of $\log_{10}$; and the average tree's ready time per unblocked session for DC-MGST-T is 0.477 lower than STAR.

3.4.3.4 Effect of Uniform DC and Membership Size

So far, we have compared the performance of STAR and DC-MGST-T for various network sizes and uniform DC settings, and set the membership size at 50% of the network size. In this section, we present the results whereby simulations are carried out in a network of 100 nodes with variation in uniform

Chapter 3

DC and membership size. Note that $|M| = 0.3|V|$, $0.5|V|$ and $0.7|V|$ in the figures indicates membership size is at 30%, 50% and 70% of the network size respectively.

Figure 3-10: Effect of Uniform DC and Membership Size on SBP in a 100 nodes network.

Figure 3-10 shows that the SBP of STAR are all higher than that of DC-MGST-T for membership size ($|Y|$) set at 30%, 50% and 70% of the network size ($|V|$). This indicates that regardless of the membership size, DC-MGST-T is able to achieve much lower SBP than STAR, and this difference is even more evidence when the membership size increases. Both STAR and DC-MGST-T achieve lower SBP as the DC increases because more receivers can be reached within the specified delay-constraint and hence lower the SBP when the delay-constraint relaxes. Also, the SBP for STAR and DC-MGST-T increases with increase in membership size. This is because as the membership size increases,

Chapter 3

the probability of at least one receiver not able to receive data within the specified DC increases, and since blocking of a multicast session occurs when at least one of its receivers is not able to receive data within the specified DC, the SBP increases as well. In addition, it can be observed that DC-MGST-T generally reaches 0 SBP when $DC=35$. This is because DC_{min} for the generated 100 nodes network is found to be 35 units and hence according to our earlier proof, as long as the DC is greater or equal to DC_{min} , all the receivers can be reached within the DC specified. STAR can only approach 0 SBP when $DC > 60$. This larger DC indicates that DC-MGST-T is more delay-tolerant than STAR, thanks to the use of MDPs to connect receivers unable to receive data within the DC specified via the shared-tree.

Figure 3-11: Effect of Uniform DC and Membership Size on CC in a 100 nodes network.

Chapter 3

Figure 3-11 shows that DC-MGST-T achieves lower CC for higher membership size. This is because as the membership size increases, the tree constructed from the shared-tree will be more cost-optimal since the shared-tree is most cost-optimal when all the nodes in it are receivers. This is in contrast to STAR where the CC is lower for lower membership size. The reason is because by having a lower membership size, more receivers are able to be connected using the least cost delay-bounded paths rather than the least cost MDPs. However, because DC-MGST-T first construct the delivery tree through pruning of a pre-constructed cost-efficient spanning shared-tree while STAR constructs each tree cost-effectively from scratch, DC-MGST-T has higher CC than STAR for all DC settings. The small difference in CC between DC-MGST-T and STAR indicates that the tree cost improvement features in DC-MGST-T such as the “trace back” phase (at stage 2) and the cost improvement phase of the employed MDPs (at stage 3) are highly effective in reducing the overall tree cost. It is also observed that the CC for STAR and DC-MGST-T decreases with an increase in DC. The reason is because the least cost MCPs will be used more frequently in STAR to connect receivers during a tree’s construction, while “trace back” and cost improvement of employed MDPs in DC-MGST-T will reap a higher cost reduction for more relaxed DC.

Figure 3-12 shows that the overheads required per unblocked session (K) for DC-MGST-T are significantly lesser than that required for STAR. For very small DC, DC-MGST-T may require more overheads since most receivers can only be connected within the DC specified using the MDPs which require overheads to activate the edges, and not via the pre-constructed shared-tree.

Chapter 3

However, it can also be noted that at very small DC, the SBP is very high and may even reach 100%. Hence, the performance for larger DC ($DC \geq 30$) is more appropriate here.

Figure 3-12: Effect of Uniform DC and Membership Size on K in a 100 nodes network.

It is observed in figure 3-12 that the K for STAR increases as membership size increases. The reason is because more paths are required to connect to more receivers and hence required more overheads generally. Also, because each tree is constructed from scratch, control packets are required to activate least cost paths to connect receivers, therefore regardless of the DC, the number of overheads required for STAR is more or less constant. In contrast, while the overheads required for DC-MGST-T increases with membership size, this is only true for smaller DC. As the DC increases, it can be observed that the overheads required for DC-MGST-T approaches a constant low level. The reason is because when the DC is small, many receivers cannot be connected

Chapter 3

within the DC specified using the pre-constructed spanning shared-tree which does not required any control packets to connect the receivers. Most receivers will need control packets to activate the improved MDPs for connection to the source, and hence increase the overall overheads required. As the DC increases, more receivers are able to be connected within the DC specified using the shared-tree and hence require no extra overheads leading to lower overall overheads required.

Figure 3-13: Effect of Uniform DC and Membership Size on ATi in a 100 nodes network.

Figure 3-13 shows that the average tree's ready time per unblocked session (ATi) for both STAR and DC-MGST-T increases with an increase in membership size. This is because more paths need to be searched and activated to connect to more receivers during the tree construction process. There is a big

Chapter 3

increase in AT_i when membership size increases for STAR, indicating that significant time is required to select the least cost paths to connect each receiver as well as to improve the tree cost iteratively by reconstructing each sub-tree rooted at each trim node. On the contrary, the increase in AT_i for DC-MGST-T when the membership size increases is significantly smaller compared to that for STAR. This is because no extra time is required to connect those receivers which can be reached within the DC specified via the pre-constructed spanning shared-tree. In addition, performing “trace back” and cost improvement of employed MDPs in DC-MGST-T takes very minimal time to process leading to even lower AT_i when compared to STAR. The small increase in AT_i when the membership size increases for DC-MGST-T at small DC settings ($DC \leq 35$) where MDPs are most likely used, reinforces the claim that it takes minimal time to improve MDPs. Similarly, the small increase in AT_i when membership size increases for DC-MGST-T at higher DC settings ($DC \geq 35$) where receivers are most likely reached via shared-tree, reinforces the claim that the “trace back” process takes minimal time. It is observed that the AT_i for STAR is only comparable to DC-MGST-T when the membership size is at 30% of the network size.

3.4.3.5 Effect of Non-Uniform DC

Table 3-8 shows that DC-MGST-T is able to achieve significantly smaller SBP than STAR for all ranges of non-uniform DC in a 100-node network. This observation is consistent with that of the uniform DC scenarios and it is not unexpected due to the fact that DC-MGST-T employs MDPs to connect those receivers not reachable within the specified DC via the shared-tree. STAR only

Chapter 3

considers the least cost delay-constrained paths to connect each receiver. Thus, future receivers will be more difficult to be connected to the delivery tree since all the previously connected receivers may cause very limited delay-residue to connect future receivers. In addition, it can be observed from table 3-8 that as the membership size increases, the SBP for STAR increases significantly. This further reinforces the claim that future receivers are harder to be connected due to the limited delay residue for STAR. This is because a session is deemed blocked as long as a receiver is unable to be connected within the DC specified and hence the increase of membership size will increase the SBP when future receivers are harder to be connected due to the limited delay residue. On the contrary, the increase in SBP for DC-MGST-T is much smaller when membership size increases. This shows that receivers can be easily connected within the DC specified for as long as a solution is possible and hence is independent of the number of receivers for each session. As the non-uniform delay-constraint range increases, it can be observed that the SBP for DC-MGST-T increases. This is because as the range increases, some of the multicast groups will be assigned a very strict DC and hence leads to higher SBP. It is interesting to observe that STAR on the other hand has more or less high SBP for all the non-uniform delay-constraint range settings. Also, it can be observed that DC-MGST-T has zero SBP for all membership sizes for the non-uniform DC range of 4 to 16 units, and its SBP increases slowly as the DC range increases. This is because when the range is small, the smallest DC among the multicast groups is not too far from the mid-point DC of 35 units where the SBP is zero. When the range is large, the smallest DC is further away from the mid-point DC, and hence the SBP is higher.

Chapter 3

Table 3-8: Effect of Non-Uniform DC on SBP in 100 nodes network.

SBP	M =0.3 V		M =0.5 V		M =0.7 V	
	STAR	DC-MGST-T	STAR	DC-MGST-T	STAR	DC-MGST-T
Range = 4	0.27	0	0.62	0	0.78	0
8	0.26	0	0.59	0	0.82	0
12	0.22	0.01	0.62	0	0.78	0
16	0.27	0.01	0.57	0	0.79	0.01
20	0.31	0	0.7	0.04	0.77	0.02
24	0.26	0.03	0.68	0.06	0.76	0.02
28	0.25	0.02	0.68	0.13	0.72	0.03
32	0.27	0.04	0.67	0.18	0.6	0.05
36	0.38	0.16	0.71	0.13	0.75	0.06
40	0.29	0.09	0.55	0.21	0.67	0.16

Table 3-9: Effect of Non-Uniform DC on CC in 100 nodes network.

CC	M =0.3 V		M =0.5 V		M =0.7 V	
	STAR	DC-MGST-T	STAR	DC-MGST-T	STAR	DC-MGST-T
Range = 4	1.003	1.105	1.105	1.204	1.003	1.130
8	0.989	1.099	1.124	1.233	1.025	1.111
12	1.004	1.121	1.120	1.205	1.041	1.078
16	0.999	1.093	1.086	1.235	1.005	1.166
20	0.992	1.092	1.074	1.228	1.018	1.103
24	1.027	1.079	1.037	1.246	1.009	1.056
28	1.025	1.117	1.146	1.233	0.994	1.086
32	0.995	1.052	1.026	1.137	0.999	1.120
36	0.992	1.098	1.060	1.213	1.006	1.057
40	0.996	1.032	1.035	1.187	0.998	1.087

Table 3-9 shows that the CC for DC-MGST-T is generally higher than that for STAR. This is expected even for non-uniform DC since DC-MGST-T constructs each tree from the shared-tree and improves MDPs while STAR constructs each tree from scratch cost-effectively. It is important to take note that the higher CC for DC-MGST-T compared to STAR is generally confined to a small value. The CC for both STAR and DC-MGST-T do not have a consistent increase or decrease pattern with an increase in the non-uniform delay-constraint range because the DC for each multicast group is assigned

Chapter 3

randomly within the range specified. Hence, as the range increases, some multicast groups are assigned an even stricter DC or more relax DC corresponding to the two ends of the range. It can be seen from table 3-9 that the STAR performs slightly better than DC-MGST-T in terms of CC, even for non-uniform DC scenarios.

Table 3-10: Effect of Non-Uniform DC on K in 100 nodes network.

K ($\log_{10}$)	M =0.3 V		M =0.5 V		M =0.7 V	
	STAR	DC-MGST-T	STAR	DC-MGST-T	STAR	DC-MGST-T
Range = 4	2.764	1.946	3.175	2.340	3.419	2.099
8	2.763	1.933	3.174	2.346	3.416	2.107
12	2.763	1.943	3.171	2.349	3.417	2.084
16	2.762	1.954	3.177	2.353	3.413	2.153
20	2.764	1.973	3.174	2.340	3.415	2.123
24	2.771	1.955	3.178	2.364	3.416	2.098
28	2.763	1.978	3.178	2.360	3.419	2.154
32	2.766	1.963	3.180	2.339	3.415	2.146
36	2.759	2.019	3.178	2.357	3.409	2.147
40	2.764	1.954	3.180	2.300	3.417	2.129

Table 3-10 shows that the overheads required per unblocked session (K) for STAR are much higher than for DC-MGST-T even for non-uniform DC scenarios. It is also observed that the overheads required for STAR remain constant when the range is widened. This is consistent with the uniform DC scenarios and within expectation because overheads are required to activate the path to connect each receiver regardless of the DC setting for each unblocked session. However, it is interesting to observe that the overheads required for DC-MGST-T are almost constant for non-uniform delay scenarios when DC-MGST-T depends on the DC setting for the overheads required, as indicated in figure 3-12 for uniform DC scenarios. The reason is that the multicast group is assigned a random DC within the range specified; hence, multicast groups may

Chapter 3

have stricter DC or more relaxed DC assigned and the difference in the DC assigned between multicast groups increases with the increase in the range specified. Thus, the average overheads required per unblocked session will not differ much for the different range settings. The important finding from table 3-10 is that the average overheads required per unblocked session for STAR is significantly higher than that required for DC-MGST-T even for non-uniform DC scenarios.

Table 3-11: Effect of Non-Uniform DC on ATi in 100 nodes network.

ATi	M =0.3 V		M =0.5 V		M =0.7 V	
	STAR	DC-MGST-T	STAR	DC-MGST-T	STAR	DC-MGST-T
Range = 4	2.907	0.445	6.328	0.479	8.449	0.687
8	2.903	0.437	7.143	0.486	7.489	0.693
12	2.884	0.430	6.396	0.479	7.415	0.690
16	3.032	0.432	6.376	0.465	8.717	0.660
20	2.976	0.429	6.775	0.477	7.385	0.668
24	3.068	0.427	6.266	0.501	7.281	0.667
28	3.465	0.423	6.662	0.486	7.949	0.634
32	3.376	0.397	6.731	0.518	7.653	0.607
36	2.934	0.381	6.086	0.513	8.724	0.599
40	2.975	0.402	5.495	0.552	8.880	0.642

Table 3-11 shows that the average tree's ready time for STAR is significantly higher than for DC-MGST-T. It is observed that the extra average tree's ready time for STAR compared to DC-MGST-T is much higher in non-uniform delay-constraint scenarios than in uniform DC scenarios. This is because for non-uniform delay-constraint scenarios, the DC for each multicast group is different; hence, the delay constrained minimum cost paths used in STAR need to be constructed for each single group leading to the increase in the average tree's ready time. This problem does not exist in DC-MGST-T since DC-MGST-T only uses a shared-tree, MCPs and MDPs, which are all pre-

Chapter 3

constructed and independent from the DC settings. While there is a significant increase in the average tree's ready time with an increase in membership size due to the very same reason as the uniform DC scenarios shown in figure 3-13 and the need to compute more delay constrained minimum cost paths for each multicast group; there is no significant increase in AT_i for DC-MGST-T when the membership size increases. The latter characteristics can be seen from the fast construction of delivery trees in DC-MGST-T even in the event of increased number of receivers for each multicast session.

3.5 Summary

In this chapter, we propose two new heuristic algorithms, namely, MGST-T and DC-MGST-T, for constructing non-delay-constrained and delay-constrained minimum cost trees respectively for multi-group multicast in an efficient manner using the shared-tree concept. It has been concluded in many existing studies that the use of pre-constructed shared tree to construct the delivery tree for each multicast session from one of the multicast groups will not achieve efficient cost since each tree is not cost effectively constructed from scratch. However, the same conclusion is not applicable to our proposed MGST-T and DC-MGST-T as they are equipped with cost improvement features which greatly increased the overall cost efficiency. For delay-constrained scenarios, it is understood that receivers which are able to receive data within the specified delay-constraint are able to achieve quality multicast services. Hence, beside cost efficiency, DC-MGST-T is also implemented in such a way that it is delay-tolerant and can allow as many receivers to receive data within the specified delay-constraint as possible even for very strict delay-constraint scenarios. To

Chapter 3

ensure that MGST-T can be easily upgraded to DC-MGST-T for delay-constrained scenarios, DC-MGST-T is developed from MGST-T simply by having an additional step to use cost-improved MDPs to connect all those receivers not able to receive data within the specified delay-constrained using the shared-tree. We compare the performance of DC-MGST-T with a recently implemented efficient heuristic algorithm known as STAR, which is described in [LMZ04]. The simulation results lead us to draw the following conclusions for our proposed DC-MGST-T algorithm.

- DC-MGST-T allows many more receivers than STAR to receive data within the specified delay-constraint, especially when the delay-constraint gets stricter.
- At the expense of slightly higher cost, DC-MGST-T could get ready a delivery tree for data transmission faster than STAR.
- At the expense of slightly higher cost, DC-MGST-T incurs a much lower overheads to construct the delivery tree for data transmission for each multicast session.
- DC-MGST-T is delay-tolerant since it uses the cost improved MDPs to connect a receiver from the source if it cannot be reached within the specified delay-constraint.
- The complexity of DC-MGST-T is made simple because it makes use of pre-constructed cost-efficient shared tree, minimum cost paths and minimum delay paths between any two nodes to construct delivery tree for each multicast session. These can be constructed easily using the existing fast and efficient algorithms.

Chapter 3

- Finally, DC-MGST-T is flexible and can be used in any scenarios. It can be downgraded to MGST-T by excluding the final third step if a relax (or no) delay-constraint is specified. If a strict delay-constraint is specified, MGST-T can be upgraded to DC-MGST-T by including the final third step.

The simulation experiments are carried out based on the assumption that the members in each multicast-group are fixed and hence each incoming multicast session request is randomly selected from one of the groups and the tree for the session is constructed accordingly. When this assumption is removed, the membership in each group is subjected to change when a new member/receiver for a multicast session joins or leaves the group. Although DC-MGST-T can still connect new receivers by means of appending the cost improved MDPs to the delivery tree or removes any paths to leaving receivers via pruning, a dynamic membership multicast version of DC-MGST-T is still required for constructing a cost-efficient and delay-tolerant tree that caters for members leaving and joining a multicast group. Development of the dynamic membership multicast version of DC-MGST-T is an interesting topic for further investigation.

Chapter 4

Proactive Rearrangement in Delay-Constrained Dynamic Membership Multicast

4.1 Introduction

Demand for multicast applications has increased in recent years due to the proliferation of Internet. As mentioned in the previous chapters, the problem of computing minimum cost tree for a multicast session can be modeled as the Steiner tree problem [Kar72] which is NP-complete. [KMB81], [Ram96] and [SK97] have since developed heuristic algorithms to solve such problems. Due to the evolution of multimedia applications over the Internet, traffic delivery delay will have a big impact on the performance of multicast applications and this leads to research directed at solving the delay-constrained minimum cost tree problem. The problem was addressed in [KPP93], [LA05], and [LMZ04]. Beside the delay constraint, dynamic membership multicast is also an essential feature to provide flexibility and resource efficiency. Dynamic membership multicast refers to multicast communications where members are allowed to join or leave a multicast session. For dynamic membership multicast, if the tree is not rearranged, the tree quality will degrade after a series of add/remove (delete) member update requests and this will lead to undesirable extra tree cost. We use the term “remove member” and “delete member” interchangeably for convenience. On the other hand, rearranging the tree frequently to obtain a

Chapter 4

lower tree cost will cause many of the remaining members to experience substantial service disruption. The disruption metric used was reviewed in [SMM99] and it was suggested in [RM06] that a member is considered disrupted as long as its route from the source is changed after each update.

Hence, dynamic membership multicast has two conflicting goals; i.e., to minimize disruptions and tree cost. [RMM99] and [RM06] proposed algorithms based on monitoring for such problems under delay-constrained environment and [BV97] proposed algorithms for non-delay-constrained environment. For algorithms based on monitoring, tree rearrangement (full or partial) is triggered only if the quality of the tree (whole or portion) falls below a specified threshold. In this way, the extra cost incurs due to the degrading of tree quality is reduced without causing too much perturbation to the remaining members. The whole tree is rearranged for full rearrangement and only certain parts of the tree are rearranged for partial rearrangement. More recently, [LLL04] proposed to fine-tune this technique by improving the triggering function that sets off the rearrangement process.

Besides disruptions and cost, the speed at which each session's tree is ready for data transmission after each update is an important factor to consider. The time taken to rearrange the session's tree after each update of adding/dropping a member (tree's setup time) has to be kept small to achieve efficient resource utilization. Most existing literature focused on minimizing disruptions and cost but few tried to address the issue on the need to reduce the rearrangement processing time (tree's setup time). Besides minimizing disruptions and cost,

Chapter 4

reducing the tree's rearrangement processing time (tree's setup time) is another goal of our proposed algorithm.

In this chapter, a delay-constrained environment was considered due to its relevance to today's multimedia applications. We refer to the dynamic multicast problem under such environment as *delay constrained dynamic membership multicast* (DCDMM) problem. In DCDMM problem, we seek to meet the three aforementioned goals by ensuring our proposed algorithm satisfies three conditions: (i) tree cost minimization; (ii) disruption minimization (via disruptions control scheme); and (iii) fast and efficient rearrangement (via proactive rearrangement strategy). So far, the existing algorithms seek to satisfy (i) and (ii) but not (iii). Also, to our best knowledge, no literature seeks to achieve (ii) using a disruption control scheme but only through rearrangement control scheme like those proposed in [RMM99] and [BV97]. The rest of this chapter is organized as follows. In section 4.2, an overview of dynamic membership multicast in the Internet and an effective existing reactive rearrangement control algorithm to DCDMM problem are presented. In section 4.3, the formal problem specification and the proposed algorithm are presented. Section 4.4 presents and discusses simulation results. Finally, the chapter is summarized in section 4.5 with the advantages of our proposed algorithm highlighted.

4.2 Overview of Dynamic Membership Multicast in Internet

4.2.1 Dynamic Membership Multicast's Scenarios

The adding and deleting of members in a single session of dynamic membership multicast consists of the four scenarios described below. We will discuss the best possible way to handle each scenario with cost and disruption reduction as the objectives. We refer to the algorithm proposed in [RMM99] as *controlled Rearrangement for Constrained Dynamic Multicast* (CRCDM) and the algorithm proposed in [RM06] as *Improved Controlled Rearrangement for Constrained Dynamic Multicast* (I-CRCDM) for convenience.

4.2.1.1 Existing-Tree (last updated tree of a session) Interior Node Added as Member

The best way to add a member node which is part of the existing multicast tree is to retain the tree and make no changes other than reflecting the node's status from non-member to member using an appropriate protocol. This way, no additional cost and disruptions will be incurred. However, for a delay-constrained environment, such a method may pose a problem. Since the node is previously included in the tree as a non-member due to low tree cost consideration before it is added as a member, the delay from the source node to this node may not necessarily satisfy the delay constraint. Hence, if the node is just reflected as a member without any change in the tree, the node's delay constraint may not be satisfied. The only exception is when the delay constraint is the same for all members (i.e. the delay to non-leaf node is always lesser than to those leaf members. Hence, if the leaf member can satisfy the delay constraint, the non-leaf node can satisfy the delay constraint as well).

Chapter 4

4.2.1.2 Node not Part of the Existing-Tree Added as Member

The best way to add a node not part of the existing tree as member is to construct a Delay Constrained Minimum Cost Tree (DCMCT) based on Prim's algorithm whereby the least cost delay constrained minimum cost path (DCMCP) is appended to the tree for each addition of member node which is not part of the tree beforehand. Such a method will result in minimum cost only if there is only one choice of least cost delay constrained minimum cost path to be appended for each added member. If there is more than one choice with the same least cost, the path to be appended will be selected arbitrarily. Such a selection may affect the cost of future trees since future addition of members can be of scenario 1 whereby no cost is incurred.

4.2.1.3 Non-Leaf Member Node Deletion

The best way to remove (delete) a non-leaf member node is to retain the tree and awaits rearrangement so that the rest of the members will not be disrupted.

4.2.1.4 Leaf Member Node Deletion

The best way to remove (delete) a leaf member node is to prune the path that services the deleted member only. In this way, the remaining members will be unaffected.

4.2.2 Effective Existing Reactive Rearrangement Control Algorithms

CRCMD was proposed in [RMM99] to limit the number of rearrangements using a Quality Factor (QF). QF measures the region's quality after each member deletion. If the QF of some region falls below a certain threshold,

Chapter 4

rearrangement will take place and the region will be removed, leaving behind some fragments. These fragments will then be connected by means of DCMCPs. The QF of a region is the ratio of the region's descendent member nodes and the number of M-nodes in the region. M-node is the equivalent of damaged node. Thus, rearrangement is activated without considering the disruptions. In other words, there is no control in the number of disrupted members per rearrangement in CRCDM. As service providers may offer QoS services in the form of allowing only a fraction of total memberships to be disrupted for each rearrangement, it is important to be able to keep track and control the number of disrupted members per rearrangement. The best way for CRCDM to ensure a given maximum disruptions not exceeded is to monitor the disruptions caused by each rearrangement. Rearrangement will not occur if the total numbers of disruptions are greater than the limit. Such practice is time consuming since rearrangement has to be performed before knowing if the disruptions produced will exceed the limit. In addition, since CRCDM rearranges trees independent from the disruptions produced, there is no mechanism for CRCDM to achieve the best possible cost tree for a given maximum number of disruptions allowed.

The total delay between the time when a delete member update is requested and the time when it is actually deleted is the duration for the fragments to be connected together if rearrangement is performed in CRCDM. This is because the tearing down of old links should be done only after the new links are set up to prevent excessive packet loss ("make before break" technique). For each connection path, the long setup time for the computation of each candidate

Chapter 4

DCMCP will result in a large delay in removing a member and cause resource inefficiency. This is because the current reserved resource supposedly to be released can be used for other incoming requests. Delay will also be incurred for an add member request as it takes time to set up the “best” DCMCP to connect the new member. It should be noted that CRCDM performs rearrangement only at certain intervals. This means that for a session, the tree cost will only be the minimum possible for some update requests, resulting in higher average tree cost.

CRCDM considers only cost minimization but not disruption. To overcome this weakness, [RM06] included another parameter in the triggering condition for rearrangement such that rearrangement is influenced by both cost and the expected disruption produced. We shall name the improved algorithm as I-CRCDM. Unfortunately, no disruption control is applied in I-CRCDM, resulting in unpredictable disruption produced. Also, [RM06] suggested that rearrangement processing time can be reduced through more link sharing. DCMCPs from the source node to all the other nodes are first pre-computed. Each time a member needs to be appended to a tree, the link cost of each candidate path within the existing tree will be set to 0. The candidate path with the least total link cost will then be selected to append the new member to the existing tree. Although this helps to significantly reduce the processing time compared to conventional method where each DCMCP candidate between two nodes is computed each time, the optimality achieved is limited to the set of DCMCPs computed from the source node, and hence may be far from the absolute optimal. This error may get larger as the network gets bigger because

Chapter 4

there are more candidate paths available to connect two nodes. The mere selection of just one out of the many candidate paths ingressed at the source node and egressed at the receiver node to be connected will result in less globally optimal solution.

4.3 Proposed PRDCDMM Algorithm

We name our proposed algorithm as *Proactive Rearrangement for Delay-constrained Dynamic Membership Multicast* (PRDCDMM).

4.3.1 Problem Formulation

4.3.1.1 Network Model

We model the network within the Internet as a graph $G = (V, E)$ where V is a set of nodes and E a set of edges. All nodes are assumed to be access points for multicast applications. Each edge $e \in E$ connecting node x and node y is denoted as (x, y) . Note that we consider a backbone network where each node represents a router, a source node is a source terminal connected to a router in the backbone, and a member node is a receiving terminal connected to a router in the backbone. Hence, each terminal (or member) node can be a non-leaf or leaf node in the backbone network.

4.3.1.2 Cost and Delay Model

Each edge e has a non-negative cost $C(e)$ and delay metric $D(e)$ associated with it. We assume symmetric cost and delay for each edge (i.e. $C(x, y) = C(y, x)$ and $D(x, y) = D(y, x)$) for simplicity although our algorithm is

Chapter 4

applicable for edges of asymmetric cost and delay as well. The cost of any path

$P = (x_0, x_1, \dots, x_n) = P(x_0, x_n)$ in the network is expressed as:

$$C(P) = \sum_{i=0}^{n-1} C(x_i, x_{i+1}) \quad (4.1)$$

The cost of any directed tree $T = (V_T, E_T)$ in the network is expressed as:

$$C(T) = \sum_{e \in E_T} C(e) \quad (4.2)$$

Each directed path $P_T(x_i, x_j)$ in tree T will hence have a cost denoted as

$C_T(x_i, x_j)$. Similarly, the delay of the corresponding directed path is denoted

as $D_T(x_i, x_j)$.

4.3.1.3 Update Requests Model

There will be a series of member addition and deletion requests for dynamic membership multicast. We refer to these requests as updates. A sequence of n updates are represented by a vector $U = \{u_1, u_2, \dots, u_n\}$. An update for adding

member node v in session $m_b = \{s_b, X_b^k\}$ is expressed as (add, v, m_b, Λ_v) , where

s_b and X_b^k are the source node and the set of members for session m_b at k^{th}

update respectively, and Λ_v is the delay constraint of the added node. If

member node v is to be removed from session m_b , the update is expressed as

$(delete, v, m_b)$. The delay-constrained minimum cost tree constructed for each

session m_b is denoted as T_b . After each update, some session trees may be

rearranged, therefore each multicast session will have a series of trees

corresponding to each update request for that session. The set of trees for

Chapter 4

session m_b is denoted as $\bar{T}_b = \{T_b^1, T_b^2, \dots, T_b^a\}$ where a is the number of update requests for session m_b .

4.3.1.4 Disruption Model

There are two main procedures for full and partial rearrangements. The first procedure is the tree construction procedure where DCMCT (full-rearrangement) or DCMCPs (partial-rearrangement) are computed. The second procedure handles the reservation and activation of new links and tearing down of old redundant links. It was mentioned in [CM01] that the severity of service disruption can be assessed in terms of the number of packets loss during disruption due to rearrangement. If the disruption duration is significant, it will cause many packets to be lost and the QoS of affected members will be degraded to an unacceptable level. The tree construction procedure takes the most time among all the processes invoked during rearrangement since the best path or tree has to be computed and its worst case complexity is $O(|V|^2)$. Thus, to minimize the disruption duration, a more pragmatic way is to adopt “make before break” approach during each rearrangement. This means that the reservation and activation of new links are carried out first before the old redundant links are torn down to deliver the traffic after each rearrangement. In this study, we consider the situation in which a member is considered disrupted as long as there is a change in its route from the original tree after each update (i.e. $P_{T_b^k}(s_b, f) \neq P_{T_b^{k-1}}(s_b, f)$) as this is the worst case.

Chapter 4

4.3.1.5 Problem Specification

The input to the DCDMM problem is given the function (M, U, DL, Θ) where U is the update request vector; $M = \{m_1, m_2, \dots, m_I\}$ is the set of multicast session requests introduced in the network, each with a source s_b and a group of initial members X_b^0 given by $m_b = \{s_b, X_b^0 = (x_1, x_2, \dots, x_w)\}$ and DL is the maximum disruption allowed per rearrangement expressed either as a percentage of the member group-size ($DL-x\%$) or as the number of members disrupted (DL). The elements in $\Theta = \{\Delta_1, \dots, \Delta_I\}$ contain the delay constraints of the corresponding member nodes for each session; and it is initialized with $\Delta_b^0 = \{\Lambda_1, \Lambda_2, \dots, \Lambda_w\}$ where each element contains the delay constraint of the corresponding initial member. The objective of the DCDMM problem is to determine a sequence of trees for the corresponding updates of each session m_b , $\bar{T}_b = \{T_b^1, T_b^2, \dots, T_b^a\}$ such that the following statements are true for each T_b^k where $1 \leq k \leq a$ using the least time possible:

(a) T_b^k spans the set $s_b \cup X_b^k$.

(b) $H_k < DL$, where $H_k = \sum_{f \in D_b^k} Y_f$ are the disruptions produced for k^{th} update,

and $Y_f = \begin{cases} 1, & \text{if } P_{T_b^k}(s_b, f) \neq P_{T_b^{k-1}}(s_b, f) \\ 0, & \text{if otherwise} \end{cases}$.

(c) $D_{T_b^k}(s_b, v) < \Lambda_v, \forall v \in X_b^k$.

(d) $C(T_b^k)$ is the minimum among all possible trees which satisfy (a), (b) and (c).

Chapter 4

4.3.2 Algorithm Description

The process block diagram of PRDCDMM is shown in figure 4-1. At the initialization stage before any multicast traffic is introduced into the network, the *Traffic-Ready Link Graph* (TRLG) is first computed. Details of TRLG computation is presented in section 4.3.3.1. Multicast traffic is then introduced and the update requests arrive randomly. If the update is an ‘add’ update and the new member node to be added is in the existing tree, the new member node will be included into the member group of the multicast session without any changes in the tree. This helps to prevent incurring any extra cost as well as disruptions. However as pointed out in section 4.2.1.1, the added member may not be able to satisfy its delay constraint. Hence, rearrangement by *Hybrid Cost Tree Partial Rearrangement* (HCTR-P) is performed to ensure that the delay constraint of each member is met. Details of HCTR-P are presented in section 4.3.3.8. Disruption control scheme is also employed to ensure that the disruptions produced from the rearrangement do not exceed the maximum specified.

If the new member node to be added is not part of the existing tree, the *minimum hybrid-cost path* (MHCP) with the least *selection function* (SF) value from the tree to the added node will be appended to the tree. Note that only the ingress of this path is allowed to be in the existing tree to prevent looping. Details of node addition are presented in section 4.3.3.2. As explained in section 4.2.1.2, if there is more than one choice with the same selection criterion to append the path, the choice may affect future updating. One example is when members can be added as interior nodes of a tree at no extra

Chapter 4

costing instead of as non-interior nodes with an extra cost equal to the cost of MHCP to reach that added member. Hence, HCTR-P is necessary to be performed to ensure that the configuration of the tree after each update is approximately cost optimal.

After the removal of a leaf-node member of a tree in response to the receipt of a ‘remove’ update, the edges used by the deleted member will be pruned and the ingress node of the pruned path is marked as a damaged node. Upon receiving a ‘remove’ update to delete a non leaf-node member from the tree, that member will be deleted and marked as a damaged node with no change in the tree. Details of node deletion are described in detail in section 4.3.3.3. Since the tree might not be a minimum cost tree after for the remaining members after a ‘remove’ update, HCTR-P is performed for both kinds of ‘remove’ updates to ensure that a minimum cost tree is achieved after each update. As we have to perform HCTR-P for every update, the update processing time has to be very fast to ensure that the objective of fast updating can be achieved by our proposed *PRDCDMM*.

Figure 4-1: PRDCDMM Blocks Diagram.

Chapter 4

4.3.3 Algorithm Details**4.3.3.1 Traffic-Ready Link Graph (TRLG) Computation**

In order to eliminate delay in tree setup due to DCMCT construction during each update, reservation channels are pre-assigned on those traffic-ready links. Since MHCPs are used in the construction of a tree, the links within MHCPs need to be assigned with channels before traffic is introduced. We refer to these links as *traffic-ready links*. MHCPs are first computed using Dijkstra's algorithm (refers to section 4.3.3.5 for details). MHCPs are then merged into a graph in which each edge in the graph is a link belonging to at least one of the paths. As each traffic-ready link is shared among the multicast sessions to compute the tree at any one update, the link will not be available if it has only one channel and is in use. We assume each channel to be able to carry one unit of the multicast traffic.

By assigning more channels to each traffic-ready link, there is a higher chance for those necessary traffic-ready links to be activated for successful rearrangement to obtain DCMCT after each update. The worst case happens when all the multicast sessions use the same traffic-ready link simultaneously to compute DCMCT throughout the sequence of update requests. Thus, the range of the number of reservation channels assigned for each traffic-ready link (v_1, v_2) is expressed as $0 \leq R_{BL}(v_1, v_2) \leq |M|$. Note that it is not desirable to assign too many channels for each link as it may lead to resource under-utilization. One way to deal with this problem is to employ non-uniform assignment of reservation channels such that some links will be assigned more

Chapter 4

channels if the usages are high. The total number of reservation channels assigned will be lesser with non-uniform assignment mode.

4.3.3.2 Node Addition Algorithm

If a member node to be added is not in the existing session tree, the MHCP with the least SF will be selected to connect the node to the tree. If the member node to be added is currently in the existing session tree, the tree structure remains unchanged.

4.3.3.3 Node Removal Algorithm

To delete a leaf member node from the tree, the interior nodes of the path which service the removed member node will be pruned. The ingress node of the pruned path has to be either a member or has an out-degree greater than 1 and the egress node is the removed member node. This ensures that no other members get disrupted from the pruning since the ingress node is the nearest node to the removed member node used to service other members as well. After pruning, the ingress node will be marked as a damaged node if it is not a member. If a non-leaf member node is to be deleted, the tree structure will be retained with the deleted node marked as damaged node.

4.3.3.4 Hybrid Cost Tree Rearrangement - Full Rearrangement (HCTR-F)

It has been shown in [KMB81] that good unconstraint *minimum cost tree* (MCT) approximation can be obtained by means of the Steiner tree approximation algorithm. We will refer this algorithm as KMB for convenience. In KMB, MCT is computed by appending a *minimum cost path*

Chapter 4

(MCP) to connect each member in an iteratively manner until all members are connected. It is also a fast heuristic since MCP between any two nodes can be computed within a very short time using Dijkstra's algorithm [Dijk59]. Once the MCPs are computed, they can be reused for each rearrangement's iteration and other multicast sessions' rearrangement. Intuitively, KMB can be extended to compute DCMCT by means of appending DCMCPs.

Unfortunately, while MCPs need to be computed only once in non-delay-constrained environment and can be reused for future rearrangement iterations, it is not the case in delay-constrained environment. This is because to connect the tree to a member which satisfies its delay constraint, the delay from the source to the point of attachment between the path and the tree has to be known and evaluated in order to find the maximum delay allowed along the connection path so that the least cost path which has delay smaller than the maximum allowed can be selected to append to the tree. Hence, there is no initial maximum delay allowed along each connection path to allow one time computation of DCMCP between any two nodes for all rearrangement iterations. DCMCP from any two nodes has to be recomputed each time as long as the delay from the source to the ingress node of the DCMCP changes.

Thus, to extend KMB for delay-constrained environment, DCMCPs from each edge of the tree to each unconnected members have to be computed using any existing delay constrained minimum cost routing strategy ([SRV97a], [HLP98] or [RMM98]) during each iteration of the rearrangement, and the least cost path will be appended to the tree. However, the number of computed DCMCPs will

Chapter 4

be enormously large and the rearrangement will take a significant amount of time, which hinders the achievement of the third goal mentioned earlier. Hence we propose an algorithm to emulate such extension and refer it as *Hybrid Cost Tree Rearrangement* (HCTR). Since paths are appended iteratively to the tree to connect the members in HCTR, it can be used for both full and partial rearrangement. We adopt full rearrangement of HCTR (HCTR-F) as the benchmark for the algorithms studied, and partial rearrangement of HCTR (HCTR-P) for the per update rearrangement in PRDCDMM.

The Hybrid cost (H-cost) of a path between any two nodes can be expressed as

$$Hy(P_{u,v}) = \frac{C(P_{u,v})}{\Lambda - D(P_{u,v})} \quad (4.3)$$

Note that this cost function prefers paths with low hybrid cost where path cost is small and delay slack is high for connection of members to the tree. The *minimum hybrid cost path* (MHCP) between any two nodes can be computed using Dijkstra's algorithm with some modification. To compute MHCP from $s \in V$ to all other nodes, Dijkstra's algorithm is still employed but four instead of two parameters are stored in each node u in V' . The four parameters are the MHCP from s to u given by $P_{s,u}$, its cost $C(P_{s,u})$, its delay $D(P_{s,u})$ and its hybrid cost $Hy(P_{s,u})$. To determine the minimum hybrid cost path from s to the next node $v \in V - V'$, for each $u \in V'$ and each $v \in V - V'$, the hybrid cost of each corresponding path is computed using the expression,

$$Hy(P_{s,v}) = \frac{C(P_{s,u}) + C(u,v)}{\Lambda - (D(P_{s,u}) + D(u,v))} \quad (4.4)$$

Since $P_{s,v} = P_{s,u} \cup e_{u,v}$,

Chapter 4

$$\Rightarrow \begin{cases} C(P_{s,v}) = C(P_{s,u}) + C(u, v) \\ D(P_{s,v}) = D(P_{s,u}) + D(u, v) \end{cases}$$

Therefore, equation (4.4) can be further reduced to $Hy(P_{s,v}) = \frac{C(P_{s,v})}{\Lambda - D(P_{s,v})}$,

which is consistent with the hybrid cost path definition given in (4.3). The path with minimum $Hy(P_{s,v})$ will then be selected as the minimum hybrid cost path from s to v and $V' = v \cup V'$. This is repeated until $V - V' = \emptyset$. By repeating the modified Dijkstra's algorithm considering each of the nodes in V as source each time, the minimum hybrid cost path between any two nodes is computed.

After the MHCPs between any two nodes for all the nodes in the network are constructed, the next step is to select one of them to append a member node to the tree constructed iteratively until all the member nodes are connected (for full rearrangement version of HCTR). The final tree will be the DCMCT. Recall that $D_T(s, u)$ is the delay of the directed path in the tree constructed so far from source s to node u . The selection function of each path for uniform delay constraint in [RMM99] is given by,

$$SF(P_{u,v}) = \frac{C(P_{u,v})}{\Lambda - D_T(s, u) - D(P_{u,v})} \quad (4.5)$$

The objective is to select the path $P_{u,v}$ with minimum $SF(P_{u,v})$ where node u is the only point of intersection between the tree and the path and v is a member node. The idea behind this is to minimize cost and maximize delay slack after the path to each member is appended to the existing tree so that the final tree can be low cost and able to accommodate future member's addition easily. Hence we can express (4.5) in terms of hybrid cost and apply the same

Chapter 4

reasoning to select the path to connect each member to the tree. Let $Hy(P_{u,v})$ be the hybrid cost of path P with u and v as its ingress and egress node respectively.

For uniform delay constraint, from (4.3), we can express the hybrid cost of a path as,

$$Hy(P_{u,v}) = \begin{cases} \frac{C(P_{u,v})}{\Lambda - D(P_{u,v})}, & \text{if } \Lambda > D(P_{u,v}) \\ \infty, & \text{if otherwise} \end{cases} \quad (4.6)$$

By rearranging (4.6), we have for $\Lambda > D(P_{u,v})$,

$$SF_{\text{Uniform}}(P_{u,v}) = \frac{C(P_{u,v})}{\frac{C(P_{u,v})}{Hy(P_{u,v})} - D_T(s,u)} \quad (4.7)$$

Hence, for uniform delay constraint, after the SF of each candidate path is computed using (4.7), we select the minimum hybrid cost path with minimum $SF_{\text{Uniform}}(P_{u,v})$ to append to the tree for member node v and repeat this until all the members are connected.

For non-uniform delay constraint, from (4.4), we can express the hybrid cost of a path as,

$$Hy(P_{u,v}) = \begin{cases} \frac{C(P_{u,v})}{\Lambda_{\max} - D(P_{u,v})}, & \text{if } \Lambda_{\max} > D(P_{u,v}) \\ \infty, & \text{if otherwise} \end{cases} \quad (4.8)$$

From [RMM99], the selection function for non-uniform delay constraint is expressed as,

$$SF_{\text{Non-Uniform}}(P_{u,v}) = C(P_{u,v}) * (D_T(s,u) + D(P_{u,v})) \quad (4.9)$$

Chapter 4

By rearranging (4.8) and (4.9), we have for $\Lambda_{\max} > D(P_{u,v})$,

$$SF_{\text{Non-Uniform}}(P_{u,v}) = C(P_{u,v}) * \left[\Lambda_{\max} - \frac{C(P_{u,v})}{Hy(P_{u,v})} + D_T(s, u) \right] \quad (4.10)$$

Hence, for non-uniform delay constraint, the $SF_{\text{Non-Uniform}}(P_{u,v})$ for each candidate MHCP is first computed using (4.10). The MHCP with the minimum $SF_{\text{Non-Uniform}}(P_{u,v})$ is then selected to be appended to the tree for member node v and this will be repeated until all the members are connected. Note that only those MHCPs which intersect the tree at its ingress will be considered for selection to prevent any looping.

4.3.3.5 Definition of Rearrangeable-sub-tree (R-sub-tree)

We define R-sub-tree with the following features:-

- (a) It is a subset of the existing tree.
- (b) It is rooted either as a non-leaf member or a source node.
- (c) It spans to its descendent member nodes and contains at least one damaged node.

The reason to select a non-leaf member node as the root of R-sub-tree is because we want to maximize the rearrangement region with minimum disruption. For example, if a non-leaf non-member node is used as the root of the sub-tree, the rearrangement is confined to a region from that root to its descendent nodes. If the nearest non-leaf member parent node to this non-leaf non-member node is used as the root, the rearrangement region will be expanded from the non-leaf member root to its descendent nodes without causing additional disruption. This is because the region from the non-leaf member root to the initially used non-leaf non-member node does not contain

Chapter 4

any members as it is the nearest non-leaf member parent node to the non-leaf non-member node.

Full rearrangement will take place if the source node is selected as the root of the sub-tree. The source node is also considered here because in full-rearrangement, not all members are disrupted since some may retain their original optimal cost path configuration and may still satisfy the disruption limit. It is possible for a R-sub-tree to be a subset of another when the root of a R-sub-tree is the descendent node of the root of another R-sub-tree. By the definition of R-sub-tree, we can easily prove that the root of each R-sub-tree is either the source node or a member node, and the leaf nodes of each R-sub-tree are all member nodes. Figure 4-2 shows an example of a tree with three R-sub-trees.

Figure 4-2: An example of three R-sub-trees.

Chapter 4

4.3.3.6 R-sub-tree Rearranged Region and Parameters Computation

Each R-sub-tree will need to compute two parameters to decide if it should be selected for rearrangement. The two parameters are cost saving (CS) and resulting disruption (RD). Cost saving (CS) is the difference between the R-sub-tree cost and the cost of its corresponding rearranged region. Larger CS means more cost saving if the R-sub-tree is rearranged. The resulting number of member nodes within the R-sub-tree which have their routes changed due to the rearrangement is defined as the resulting disruption (RD) factor of that R-sub-tree.

To compute these two parameters, we have to compute the rearranged region of the R-sub-tree first. To compute such a region, the R-sub-tree is first removed. The unconnected members are then connected to the remaining tree by iteratively appending the MHCP with the least SF from one of the unconnected member to the remaining tree (original tree less the removed R-sub-tree). This continues until all the unconnected members within the R-sub-tree are connected. The computed final tree less the original remaining tree after the R-sub-tree is removed will be the rearranged region. Note that only those traffic-ready links which have sufficient reservation channels will be considered in the computation of connection paths to ensure non-blocking performance.

4.3.3.7 Definition of Disruptions Limit per Update

We define Disruptions Limit per update (DL) as the maximum number of members allowed to be disrupted per update. It can be expressed in terms of the number of members or percentage of the member group size of each session

expressed as $DL-x\%$ where $x = \frac{100 * DL}{|D_j|}$. It may be argued that the disruption

Chapter 4

limit set for the entire traffic duration is more interesting than the per update limit. We believe that per update disruption is more meaningful and useful because per update affects each routing of each session and allows QoS to be fine-tuned according to each rearrangement. For example, there can be significant disruption in one rearrangement but none in others resulting in low overall lifetime disruption. But we know that this is not acceptable because all clients' services are disrupted for at least one time. Hence, it is more practical to consider the damage done per rearrangement.

Since the RD factor shows the number of disruptions for each update request if a selected R-sub-tree is rearranged and the CS factor shows the corresponding cost reduction, if the disruption limit per update is given, the appropriate sub-tree can be selected for rearrangement to give maximum cost saving without violating the disruption limit. This provides the flexibility to trade off between cost and disruption. For applications which allow no disruption, DL can be set to 0. For multicast sessions with different disruption requirements, DL can be set differently for each session.

4.3.3.8 Hybrid Cost Tree Rearrangement-Partial Rearrangement (HCTR-P)

The R-sub-trees within the tree are first identified and ranked according to the number of damaged nodes contained within them. As a R-sub-tree can be a sub-tree of another, it is easy to see that the largest R-sub-tree will contain the maximum number of damaged nodes. Hence, starting from the highest to lowest ranking R-sub-tree, the rearranged region, and CS and RD parameters of

Chapter 4

the R-sub-tree are computed. If the RD is less than the DL specified, the search ends and that R-sub-tree will be selected for rearrangement with its corresponding rearranged region and CS already computed, otherwise the next ranked R-sub-tree is considered in a similar fashion until a suitable R-sub-tree is found. In this way, we need not consider all R-sub-trees and hence able to search for the best R-sub-tree fast for rearrangement to achieve maximum cost saving with acceptable disruption.

Those required traffic-ready links will be activated straight away to rearrange the tree after the rearranged region is computed. Since only links with sufficient bandwidth will be considered for the computation of the rearranged region, as long as a rearranged region exists and there is no physical damage to the links, the rearranged process will not experience any delay. In addition, since only MHCPs are considered, the R-sub-tree rearrangement may not always be able to construct a valid tree connecting all members. In the case of a failure to construct a valid tree, that particular R-sub-tree will not be considered. If all the R-sub-trees are unable to construct a valid tree during rearrangement, the previous tree (before rearrangement) less the deleted node will be retained.

4.3.3.9 Reservation Channels Assignment Mode

We introduce two modes of reservation channel assignment for PRDCDMM. The first is the *uniform channels assignment* mode (UA). For UA, all traffic-ready links are assigned with the same number of channels. The second mode is the *non-uniform channels assignment* mode (Non-UA). For Non-UA, a fixed total number of channels are assigned. However, these channels are assigned

Chapter 4

non-uniformly among the traffic-ready links. The details on how to assign the channels to each traffic-ready link in the non-uniform mode is presented next. We shall refer to the UA version of PRDCDMM as UA PRDCDMM and the Non-UA version as Non-UA PRDCDMM. Note that the difference in these two versions is merely the choice of reservation channel assignment mode.

4.3.3.10 Non-Uniform Channels Assignment Mode

How frequent to use a traffic-ready link to deliver traffic depends very much on which of the nodes need to have their routes changed to achieve minimum cost configuration. Since the knowledge of the updates is not known, the best way to approximate traffic-ready link usage is to find out how frequent the traffic-ready link is used to construct DCMCT. We refer to this frequency as *link criticality weight* (LCW). If a traffic-ready link is often used to reach the destinations in a tree (higher criticality weight), naturally the probability of using it is higher compared with one which it is seldom used (lower criticality weight). Thus, the criticality weight of a traffic-ready link is expressed as,

$$LCW(e) = \sum_{z \in Z} X(z, e) \quad (4.11)$$

where Z is the set of all minimum hybrid-cost paths between each pair of

nodes, and $X(z, e) = \begin{cases} 1, & \text{if } e \in z \\ 0, & \text{if otherwise} \end{cases}$.

A traffic-ready link which has a higher criticality weight indicates that it has a higher chance to be activated to compute DCMCT when compared with one which has a lower weight. Hence, the traffic-ready links are ranked according to their weights with the largest weight having the top ranking (Rank 1). Those which have the same weight will be ranked as a group with the same ranking.

Chapter 4

As some of the links may have very low weight, they could be omitted for reservation with little impact on cost saving. This will reduce the number of redundant traffic-ready links and the resources could be better shared among the more critical traffic-ready links. Hence we introduce the *weight limit ratio* (WLR) factor to distinguish which traffic-ready links can be omitted for reservation. WLR is defined as the minimum weight required for the traffic-ready link to be assigned at least one channel.

The channels assigned to each link (i, j) is given by the expression,

$$\left\lfloor \frac{LCW(i, j)}{\sum_{\hat{e} \in \hat{E}} LCW(\hat{e})} \times CH_{\text{Total}} \right\rfloor \quad (4.12)$$

where $\hat{E}$ is the set of all edges in TRLG, and CH_{Total} is the total reservation channels to be assigned for Non-UA PRDCDMM. The expression in (4.12) ensures that the total number of channels assigned will never exceed the limit set. However it does not guarantee that all channels will be assigned. To ensure that all the channels that need to be assigned are assigned, the remaining channels will be assigned to each link in a round-robin manner in order of link ranking.

4.3.3.11 Modification to Reduce Total Disruptions in PRDCDMM

PRDCDMM controls the number of disruptions per update (and hence per multicast session) and does not restrict the total number of disruptions produced for all the multicast sessions combined. It is not desirable for the total number of disruptions to be very large. To overcome this problem, we modify PRDCDMM in terms of the rearranged tree selection criteria. Recall that the

Chapter 4

criteria for selecting a R-sub-tree for rearrangement is $RD < DL$ and since the largest R-sub-tree is considered first, the selected R-sub-tree also has maximum CS . It does not take into consideration the number of disruptions produced for as long as it does not exceed the DL specified. Hence, in some situations, the cost reduction might be small (e.g. 1 unit) while the number of disruptions produced is significantly larger (e.g. 10 or even 20), which is not desirable. Thus, we introduce a new parameter to take into consideration the number of disruptions produced. Let $\mathfrak{R}$ be the set of all R-sub-trees and T_{Re}^x be the corresponding rearranged tree after the R-sub-tree x , is rearranged. Hence, for a particular multicast session m_b , the updated tree T_b^k at its k^{th} update is computed as follows:

$$T_b^k = \rho T_{Re}^x \quad (4.13)$$

$$\text{where } CS_x = \max_{i \in \mathfrak{R}} CS_i \text{ and } \rho = \begin{cases} 1, & \text{if } \frac{CS_x}{RD_x} > 1 \\ \frac{T_b^{k-1}}{T_{Re}^x}, & \text{if otherwise} \end{cases}$$

The ratio $\frac{CS_x}{RD_x}$ is defined as the maximum cost saving per disruption produced.

Hence, the above equations ensure that the cost saving must be greater than the disruption produced in order to justify that the advantage of cost saving outweighs the disruption produced and hence proceeds with the rearrangement. Note that this parameter can always be adjusted, for example, CS and RD can be normalized according to the user's definition of per unit cost and disruption. Note that the above equation merely takes into consideration the number of disruptions produced for each rearrangement tree to reduce the total number of disruptions produced.

4.4 Simulation Results and Discussions

In this section, we present simulation results and discuss the performance of PRDCDMM. There are two sets of performance metrics to be used: (a) Performance metrics for disruption and tree costs; (b) Performance metrics for resource utilization. The first set is to compare PRDCDMM with CRCDM, I-CRCDM and optimal algorithms in terms of disruption and tree cost. The second set is to compare the resource utilization between UA and Non-UA PRDCDMM.

4.4.1 Performance Metrics for Disruptions and Tree Costs

4.4.1.1 Cost Competitiveness (CC)

This metric is similar to the one used in [RMM99]. It is a measure of a given algorithm's performance in terms of average tree cost with respect to an optimal algorithm. Finding the optimal DCMCT is proven to be NP-complete, thus it is impractical to use the optimal algorithm for comparison. As mentioned earlier we shall use the full rearrangement version of *Hybrid Cost Tree Rearrangement (HCTR-F)* proposed in section 4.3.3.4 as the benchmark. Since HCTR-F only considers minimum hybrid-cost paths in its rearrangement, it may not always be possible to construct a valid tree. In our simulation, we repeat the experiments until valid trees can be constructed for all sessions. Throughout the entire dynamic multicast period, a sequence of trees obtained by a given algorithm and by *HCTR-F* for the corresponding updates of each session m_b is given by $\bar{T}_b = \{T_b^1, T_b^2, \dots, T_b^a\}$ and $\bar{Y}_b = \{Y_b^1, Y_b^2, \dots, Y_b^a\}$ respectively where a out of n update requests are directed to session m_b . Therefore the CC of the given

Chapter 4

algorithm is expressed as, $CC = \frac{1}{n} \sum_{m_b \in M} \left(\frac{C(\bar{T}_b)}{C(\bar{Y}_b)} \right)$ where M is the set of multicast sessions introduced, $C(\bar{T}_b) = \sum_{i=1}^a C(T_b^i)$ and $C(\bar{Y}_b) = \sum_{i=1}^a C(Y_b^i)$. A low CC is desired as it means that the given algorithm is able to perform the update requests without incurring too high a cost compared to the optimal algorithm.

4.4.1.2 Total Disruptions (TD)

This measures the total number of members which have their services disrupted after a series of rearrangements for the entire sequence of update requests. We assume that the service of a member is disrupted as long as its original route from the source is changed after a rearrangement. The number of disruptions incurred for k^{th} update is expressed as, $H_k = \sum_{f \in D_b^k} Y_f$ and hence the total number of disruptions is expressed as, $TD = \sum_{k \in U} H_k$. A low TD means that the total number of disruptions caused by a given algorithm is small and is highly desirable.

4.4.1.3 Maximum Disruption (MD)

This measures the maximum number of members experiencing service disruption among all the update requests. It is expressed as, $MD = \max_{k \in U} H_k$. A low MD means that the maximum number of disruptions caused by a given algorithm for any update is small and is highly desirable since the disruption QoS can be achieved more easily.

Chapter 4

4.4.1.4 Disruption Competitiveness (DC)

CC is introduced to measure the cost competitiveness of a given algorithm with respect to the optimal algorithm. Since the optimal algorithm will produce the most number of disruptions due to full rearrangement, we can measure disruption competitiveness (DC) of a given algorithm with respect to the optimal algorithm by computing the difference in number of disruptions produced. By taking the optimal (cost) algorithm as the reference, the DC of a heuristic is expressed as,

$$DC = \frac{TD_{Opt} - TD_{Algor}}{TD_{Opt}} \quad (4.14)$$

where TD_{Opt} and TD_{Algor} are the total number of disruptions produced by the optimal algorithm and a target algorithm respectively for all multicast sessions for all the updates. A low DC means that the total number of disruptions produced by a given algorithm is not too far off compared to the high number of disruptions produced by the optimal algorithm and hence is not desirable. A high DC means that the total number of disruptions is very small compared to the optimal algorithm and is desirable. The minimum and maximum value of DC is 0 (i.e. $TD_{Algor} = TD_{Opt}$) and 1 (i.e. $TD_{Algor} = 0$) respectively.

4.4.1.5 CC-DC Performance Index (Υ)

Since a high DC and low CC is preferred, the CC-DC Performance Index is express as:

$$\Upsilon = \frac{CC}{DC} \quad (4.15)$$

Chapter 4

Since $CC \geq 1$, $0 < DC \leq 1$ and $DC \neq 0$ for $Y \neq \infty$, $Y \geq 1$. The best performance occurs when a given algorithm computes trees with the same cost as the optimal algorithm with no disruption. Hence, $CC=1$ and $DC=1$ will give the best performance index. Thus, an algorithm which has a low index, Y indicates that it is able to compute trees and produce disruptions not much different from the best performance and hence highly desirable. This new index is able to give a better gauge of dynamic multicast performance of a given algorithm since both disruption and cost are now taken into consideration holistically instead of individually like those used in the existing literature.

4.4.2 Performance Metrics for Resource Utilizations

4.4.2.1 Redundant Traffic-Ready Links (RL)

This measures the total number of redundant traffic-ready links. Redundant traffic-ready link is defined as a link which has reservation channels assigned to it but not used for any update requests. A low RL indicates that the amount of resource wasted is small and is highly desirable.

4.4.2.2 Average Maximum Effective Traffic-Ready Link Utilization (AMELU)

For each traffic-ready link, the maximum effective traffic-ready link utilization (MELU) is computed by taking the ratio between the maximum number of channels used in the link for all multicast sessions and the total number of channels assigned for the link. AMELU is then computed by taking the average of MELU among all traffic-ready links excluding those redundant traffic-ready

Chapter 4

links. A high AMELU indicates that the algorithm is able to operate with efficient usage of the reserved channels in traffic-ready links.

4.4.3 Simulation Model

4.4.3.1 Random Network Generation

We make use of Waxman's method [Wax88] to generate random networks of 20, 40, 60, 80 and 100 nodes with an average node output-degree of 3 to 4 to analyze the performance of PRDCDMM. 100 multicast sessions are randomly generated and setup initially using *HCTR-F* before the arrival of update requests. Situations whereby only one member exists and a deletion update is requested, or when no member exists and an add update is requested will not be considered in this thesis as they can be easily solved by removing or appending the appropriate paths without incurring much redundant tree cost or disruption since the number of members are very small. Readers are referred to section 3.4.2.1 and [Wax88] for more details on network generation and its cost and delay allocation.

4.4.3.2 Update Request Generation

We use the probabilistic model employed in [RMM99] to generate the addition and removal update requests. For selecting a new member to be added in an update request, non-member is randomly chosen from the graph. For removal request, older members have a higher chance to be deleted. Each multicast session is generated with 100 update requests to arrive in random fashion together with those from other multicast sessions.

Chapter 4

4.4.3.3 Delay Constraint Parameters

For uniform delay constraints where all members have the same constraint, we set the constraint to be 50. For non-uniform delay constraint, we set a range with 50 as the mid-point. For example, a range of 10 means that the delay constraint among members will vary from 45 to 55.

4.4.3.4 Simulation Parameters

In PRDCDMM, we have two variable parameters, *weight limit ratio* (WLR) and *disruption limit* (DL) per update. WLR- $x\%$ indicates $WLR = 0.01 \times x \times$ (total number of minimum hybrid-cost paths). Note that between each pair of nodes in the network, a minimum hybrid-cost path is computed. DL- $y\%$ indicates DL is $0.01 \times y \times |D_b^k|$ for $k = 1, 2, \dots, n$, where $|D_b^k|$ is the member-group size of session m_b at k^{th} update. We use a threshold of 3 and 6 for CRCDM. The threshold settings for CRCDM will not affect the comparison evaluations between PRDCDMM and CRCDM because our proposed performance index takes both cost and disruptions into consideration. While comparing PRDCDMM with I-CRCDM, the DL and threshold for PRDCDMM and I-CRCDM respectively are set in such a way that the CC-DC performance index difference between the two is less than 0.3%. The idea is to compare PRDCDMM and I-CRCDM in terms of tree setup time and maximum number of disruptions produced per update under the condition that both achieve similar CC-DC performance index. A list of parameters used is given in table 4-1.

Chapter 4

Table 4-1: Simulation Parameters

Parameter	Values/Method
Source node	Randomly selected.
Added member	Any non-member equally likely to be selected.
Deleted member	Older member more likely to be selected.
Average node degree	Between 3 and 4.
$ M $; $ U $; CRCDM Threshold; $ V $	100; 10000; 3,6; 20,40,60,80,100
Average member group size	$0.3 V $
WLR-x%; DL-y%	$x = 0.2, 0.5$; $y = 10, 20, 30, 40, 50$
Channels assigned (UA).	3 per traffic-ready link
Channels assigned (Non-UA).	40% of total in UA.
Uniform member's delay constraint	$\Lambda = 50$
Non-uniform member's delay constraint range	10,20

4.4.4 Simulation Results and Discussions

The results in figure 4-3 to figure 4-6 are obtained from the average of 100 multicast sessions randomly generated for each network size. Full rearrangement is employed to show the full potential of the two algorithms without restriction on disruptions. Figure 4-3 shows that HCTR-F performs slightly better than the approximate optimal $DCMT_{CD}$ reported in [KPP93]. This is within the expectation since HCTR-F and $DCMT_{CD}$ employed the same selection concepts; however HCTR-F connects members directly via selected paths while $DCMT_{CD}$ connects members via hop-by-hop links. Figure 4-4 shows that the time complexity of HCTR-F is much smaller than $DCMT_{CD}$. This is because HCTR-F considers minimum hybrid-cost paths from the existing tree to members for attachment and these paths are required to compute only once and can be used for future tree computation. On the other hand, $DCMT_{CD}$ considers links to the neighboring nodes of the existing tree nodes for attachment and this process is repeated until all members are connected. Since HCTR-F attaches each member directly via a selected path which needs to compute only once while $DCMT_{CD}$ attaches each member via a direct link or

Chapter 4

over a series of links with non-member nodes in between, hence the computation time is longer for DCMT_{CD} . In addition, no looping occurs for HCTR-F since only those paths which have their ingress nodes as the only intersection point with the existing tree are considered while DCMT_{CD} needs additional time to remove looping and re-compute a non-looping tree.

Figure 4-3: Cost comparison (uniform delay constraint =50).

From Figure 4-5 it can be observed that the average tree cost decreases when the delay constraint relaxes. This is logical since lower cost paths are now more likely to satisfy the delay constraint and be attached to the tree. HCTR-F has a lower average tree cost than DCMT_{CD} mainly because HCTR-F connects each member directly via a selected path in one single iteration while DCMT_{CD} connects some members indirectly via a series of selected links over a series of iterations. Thus, during tree computation per iteration, HCTR-F considers cost in terms of each selected path from the existing tree to a member while

Chapter 4

DCMT_{CD} considers cost of each link to neighboring nodes of the existing tree. Since the neighboring nodes may not be member nodes, the cost consideration for DCMT_{CD} may not be as optimal as HCTR-F, resulting in higher average tree cost. The only advantage of DCMT_{CD} over HCTR-F is that it is a distributed- approach with lesser overheads.

Figure 4-4: Tree's Setup Time comparison (uniform delay constraint =50).

Figure 4-6 shows the average tree cost performance of HCTR-F and DCMT_{CD} under non-uniform delay constraint environment where each member has a different delay constraint. The results are consistent with that of uniform delay-constraint environment except that HCTR-F has slightly better performance than DCMT_{CD}. Unlike the uniform delay-constraint case where the delay slack component in the selection function can be accurately computed due to knowledge of delay-constraint of each member, non-uniform delay-constraint selection function is unable to do likewise since each member's delay-

Chapter 4

constraint is different. Owing to inability to compute the delay slack, the best minimum cost path or link cannot be selected. Selection is based on low cost and low delay of path from the source node to the connected node instead. Since some members may have a loose constraint which allows a low cost and large delay path to be attached, there is no way the selection function is able to identify it since delay slack is not considered. Thus this leads to higher average tree cost. In HCTR-F, the hybrid cost computation for non-uniform delay-constraint considers the maximum possible delay slack. Thus, each potential minimum hybrid-cost path has already taken the estimated delay slack into consideration and has better performance than DCMT_{CD} .

Figure 4-5: Cost comparison under different uniform delay constraint in a network of 100 nodes.

Figure 4-3 to figure 4-6 show that HCTR-F is able to construct DCMCT better than DCMT_{CD} within a much shorter time frame. Although one may argue that

Chapter 4

HCTR-F may not be able to find a solution when the delay constraint is very tight as it only considers a fraction of all the possible paths between any two nodes, we can also say the same for $DCMT_{CD}$. This is because HCTR-F only considers paths with low cost and large delay slack to be added to the tree. Hence, if no solution is found, it is mainly because no possible solution exists rather than because all possible paths are considered.

Figure 4-6: Cost comparison under different network size when delay constraint is non-uniform at 50 with range of 30 (i.e. [35, 65]).

Figure 4-7 and figure 4-8 show the CC and disruption results respectively for UA PRDCDMM and CRCDDM. It can be observed that for different network sizes, UA PRDCDMM in general has lower CC and lower total number of disruption than CRCDDM. This shows that the disruption control strategy in PRDCDMM is able to minimize the competing effect of disruption and cost by

Chapter 4

having optimal disruption to achieve minimum cost for a given DL, and hence performs better than CRCDM. For UA PRDCDMM, as DL increases, the disruption curve shifts up while the CC shifts down since more disruptions per update are allowed, causing more DCMCT to be computed per update. However, it can be seen that such shifting is minimal, and for large networks, the total disruptions and CC for different DL settings of UA PRDCDMM are almost similar. This is mainly due to the modification we introduced in section 4.3.3.11 to ensure that disruptions are accompanied by large cost reduction.

Figure 4-7: CC comparison between UA PRDCDMM and CRCDM under uniform delay constraint of 50.

Chapter 4

Figure 4-8: Disruptions comparison between UA PRDCDMM and CRCDM under uniform delay constraint of 50.

Figure 4-9: UA PRDCDMM and CRCDM comparison in maximum disruptions under uniform delay constraint of 50.

Chapter 4

Figure 4-9 shows that CRCDM has higher maximum number of disruptions compared to UA PRDCDMM. This is expected because there is no disruption control in CRCDM. As long as rearrangement is allowed, the least cost tree will be computed regardless of the number of disruptions incurred. However, for UA PRDCDMM, a disruption control scheme is adopted and the maximum number of disruptions incurred cannot exceed the DL specified. Hence, UA PRDCDMM has lower maximum number of disruptions for any update. It can be observed that the maximum number of disruptions for UA PRDCDMM increases with an increase in DL. Thus, the flexibility of setting an appropriate DL for the performance required by the user can be provided in PRDCDMM.

Figure 4-10 shows the overall performance that combines both the number of disruptions as well as cost. As observed in figure 4-7 and figure 4-8, CRCDM-Threshold-6 has slightly smaller CC than CRCDM-Threshold-3. However, it has much higher total number of disruptions than CRCDM-Threshold-3. Thus, in figure 4-10, it is clear that the performance index of CRCDM-Threshold-3 is much better (recall that a lower index means better performance) than CRCDM-Threshold-6. Since both CC and number of disruptions by UA PRDCDMM are lower than CRCDM, the performance index of UA PRDCDMM is better than CRCDM as shown. Thus, under uniform delay constraint environment, UA-PRDCDMM performs better than CRCDM. However, as mentioned earlier, the nature of channel reservations in UA-PRDCDMM might cause inefficiency in resource utilization. Thus, Non-UA PRDCDMM is introduced to solve this problem.

Chapter 4

Figure 4-10: UA PRDCDMM and CRCDM comparison in CC-DC Performance Index under uniform delay constraint of 50.

Figure 4-11: Average Tree's Setup Time Results.

Chapter 4

Figure 4.11 shows the average rearrangement processing time (average tree's setup time) per update. As the paths to low cost trees during rearrangement in PRDCDMM are computed and reserved with channels during initialization, time to compute and set up rearranged trees are eliminated during each rearrangement. This is in contrast to CRCDDM where DCMCPs are computed during rearrangement to connect all the fragments together. As observed from figure 4.11, CRCDDM processing time increases exponentially with network size since it takes longer time to find DCMCP in larger networks while PRDCDMM rearrange tree for each update almost instantly. This is a very important feature of PRDCDMM since it reduces the 'delay effect' of adding and deleting a member greatly and it reduces the likelihood of race conditions significantly where the next update request arrives before the rearrangement is completed. As mentioned earlier, what is really important is the duration between the time an update is requested and the update is actually processed (tree's setup time). In addition, race conditions should be minimized or eliminated. Hence, the rearrangement time or tree setup time is much more critical than the initial setup timing. For this reason, we focus on the rearrangement time in between updates.

Although I-CRCDDM was proposed in [RM06] to improve the triggering function for rearrangement by considering both the cost and disruptions produced, we are able to show in figure 4-12 and figure 4-13 that for almost the same CC-DC performance index, UA PRDCDMM is able to have much smaller average tree setup time and lesser maximum disruptions produced. The threshold of I-CRCDDM and DL of UA PRDCDMM are set such that the CC-

Chapter 4

DC performance index difference between them is less than 0.3% to achieve the results shown in figure 4-12 and figure 4-13. The much smaller average tree setup time and lesser maximum disruptions produced in UA PRDCDM over I-CRCDM are due to the proactive nature of UA PRDCDM and the presence of disruption control. It can be observed from figure 4-13 that as the network size increases, I-CRCDM's average tree setup time can be up to 100 times more than UA PRDCDM. The reason behind CRCDM's substantial increases in average processing time (tree's setup time) is that the tree's re-construction is done only when a re-arrangement is required. This means that the new tree has to be re-constructed using the appropriate existing routing algorithms before continue to deliver data to the updated members. Hence, if the network size increases, the re-construction process will take longer time since more nodes need to be considered and searched to construct the best tree. For PRDCDM, all the potential new paths to route the data to the updated members are constructed proactively prior the arrival of the multicast session requests (during initialization stage). Thus, those appropriate new tree-paths which have idle wavelength channels can be immediately used to construct the new tree and deliver data without any delay after each member's joining or leaving the session. Hence, there is no significant increase in average processing time even if the network size increases. The significance of such a big improvement in tree's setup time is that there will be almost no delay in transmitting data to the existing members should a member join or leave the multicast session. This means that the disruption time to deliver data to existing members after a member join or leave the multicast session is extremely small.

Chapter 4

Figure 4-12: Maximum Disruptions Comparison between I-CRCDM and UA PRDCDMM while achieving the difference in CC-DC Performance Index between them less than 0.3%.

Figure 4-14 to figure 4-16 show the performance of Non-UA PRDCDMM over UA PRDCDMM. As observed in figure 4-14, when a lower value of WLR is set, the performance index is higher. This indicates that more traffic-ready links are assigned with reservation channels and thus allows more successful rearrangements to take place to compute DCMCT. Although more cost saving can be achieved with increased successful rearrangements, the number of disruptions also increases. The increase in the performance index is due to the saving in tree cost not large enough to compensate for the corresponding disruptions produced. However, it can be observed that the difference in CC-DC performance only increases when DL increases, and the CC-DC performance index of Non-UA PRDCDMM (WLR-0.2%) is not too far off

Chapter 4

from that of UA PRDCDMM. Non-UA PRDCDMM (WLR-0.2%) has far fewer redundant links and far higher AMELU than UA PRDCDMM, as shown in figure 4-15 and figure 4-16 respectively, while achieving similar performance index as UA PRDCDMM. This indicates that we are able to achieve a reasonably good CC-DC performance index with high resource utilization and efficiency (i.e. low RL and high AMELU) by setting an appropriate WLR in Non-UA PRDCDMM. As observed from the results in figure 4-14 to figure 4-16, WLR of 0.2% is an appropriate setting for 100 nodes network because Non-UA PRDCDMM (WLR-0.2%) has almost similar good performance as UA PRDCDMM while having much lower RL and much higher AMELU than UA PRDCDMM.

Figure 4-13: Average tree's rearrangement processing time (tree's setup time) between I-CRCDM and UA PRDCDMM while achieving the difference in CC-DC Performance Index between them less than 0.3%.

Chapter 4

Figure 4-14: Non-UA and UA PRDCDMM comparison in terms of CC-DC Performance Index for 100 nodes network under uniform delay constraint of 50.

Figure 4-15: Non-UA and UA PRDCDMM comparison in terms of RL for 100 nodes network under uniform delay constraint of 50.

Chapter 4

Figure 4-16: Non-UA and UA PRDCDMM comparison in terms of AMELU for 100 nodes network under uniform delay constraint of 50.

Figure 4-17: Effect of network size on CC-DC performance index comparison.

Chapter 4

Figure 4-18: Effect of network size on redundant links produced between Non-UA and UA PRDCDMM.

Figure 4-19: Effect of network size on AMELU between Non-UA and UA PRDCDMM.

Chapter 4

Figure 4-17, figure 4-18 and figure 4-19 show the performance of Non-UA and UA PRDCDMM in terms of performance indices RL and AMELU respectively for different network sizes. DL and WLR are set to 30% and 0.2% respectively for a uniform delay constraint of 50. The reason for using these settings is that both Non-UA and UA PRDCDMM yield almost similar performance under these settings for 100-node networks, as shown in figure 4-14. Thus using these settings, we are able to show that Non-UA PRDCDMM is able to provide similarly good performance as UA PRDCDMM for different network sizes with much smaller redundant links (shown in figure 4-18) and higher traffic-ready link utilization (shown in figure 4-19). The results in figure 4-17 to figure 4-19 lead us to conclude that UA PRDCDMM performs better than CRCDM but may have low traffic-ready link utilization and high redundant links which are not desirable. Hence, using a suitable WLR, Non-UA PRDCDMM is able to achieve similarly good performance as UA PRDCDMM but at a much higher utilization rate and lower redundant links.

The two parameters RL and AMELU are used to gauge if my proposed proactive rearrangement algorithm is efficient. As mentioned, one major problem of proactive rearrangement algorithm is that there may be resources reserved but not used, and the utilization rate in some links may be too low. In other words, there may be a lot of cases where wavelength channels are reserved proactively, but not used at all. Hence, an algorithm which has high RL and low AMELU indicate that the resource is not used efficiently. Although my proposed algorithm UA-PRDCDMM is able to achieve better results than CRCDM, it has high RL and low AMELU which are not desirable. Thus, with

Chapter 4

low RL and high AMELU results, my proposed improved algorithm Non-UA PRDCDMM is shown to be able to not only achieve the similar better result compared to CRCDM, but also with much efficient resource usage.

Table 4-2 shows that as WLR-x% increases, the total number of traffic-ready links that are assigned with channels decreases. This is because as WLR increases, there are fewer traffic-ready links with LCW greater than the specified WLR to warrant channel assignment.

Table 4-2: Number of Traffic-Ready Links Assigned with Channels.

Network Size (Number of nodes)	% of total traffic-ready Links assigned with channels	
	WLR-0.2%	WLR-0.5%
20	86.3	86.0
40	80.2	73.6
60	77.2	49.1
80	71.0	44.2
100	62.0	39.2

Table 4-3: Performance in Non-uniform Delay Constraint Environment with delay range of 10 and network size of 100 nodes.

Delay Range =10	CC	Total Disruptions	Maximum Disruptions	CC-DC Performance Index	RL	AMELU
Non-UA PRDCDMM (WLR-0.2%, DL-30%)	1.139353	702	8	1.17757	115	0.807759
UA PRDCDMM (DL-30%)	1.133068	836	9	1.17862	328	0.541265
CRCDM (Threshold-3)	1.174968	734	11	1.21624	-	-
CRCDM (Threshold-6)	1.168469	1512	29	1.25628	-	-

Chapter 4

Table 4-4: Performance under Non-uniform Delay Constraint Environment with delay range of 20 and network size of 100 nodes.

Delay Range =20	CC	Total Disruptions	Maximum Disruptions	CC-DC Performance Index	RL	AMELU
Non-UA PRDCDMM (WLR-0.2%, DL-30%)	1.144607	688	9	1.182241	121	0.810598
UA PRDCDMM (DL-30%)	1.136446	787	9	1.179392	333	0.533333
CRCDDM (Threshold-3)	1.175612	730	11	1.216707	-	-
CRCDDM (Threshold-6)	1.167016	1501	29	1.254112	-	-

Table 4-3 and 4-4 show the performance of Non-UA and UA PRDCDMM for 100-node networks under non-uniform delay constraint when the delay constraint range is set to 10 to 20. The results from the tables indicate that PRDCDMM performs better than CRCDDM. They also show that Non-UA PRDCDMM has much higher AMELU and lower RL while having almost the same CC-DC performance as UA PRDCDMM. It is clear from the table that maximum disruptions per rearrangement for PRDCDMM (for both Non-UA and UA) are also lower. The results from table 4-3 and table 4-4 lead us to conclude that the better performance of PRDCDMM over CRCDDM is consistent for both non-uniform and uniform delay constraint environments.

4.5 Summary

In this chapter, we propose a new algorithm and named it PRDCDMM to minimize not only tree cost and disruptions in delay constraint dynamic multicast problem but also the per update rearrangement processing timing (or

Chapter 4

tree setup time). In addition, we propose to control disruption by controlling the number of disruptions per rearrangement. The disruption control mechanism helps to obtain the best cost and disruptions performance while incurring a maximum allowable number of disruptions. It also helps to limit disruptions incurred for each update to a pre-specified maximum. To ensure fast rearrangement in PRDCDMM, a hybrid cost tree rearrangement (HCTR) strategy is proposed. We show why an effective algorithm such as KMB from [KMB81] which computes minimum cost tree using MCP could not be extended to delay constraint environments by simply employing DCMCP for dynamic multicast. We propose HCTR for constructing a tree by appending suitable connection paths known as minimum hybrid-cost paths instead of DCMCPs from the tree to each unconnected members using a modified Prim's algorithm. We show that HCTR can be applied for full rearrangement (HCTR-F) and partial rearrangement (HCTR-P). The paths used for rearrangements are computed and reserved with channels before traffic arrives and can be used for future updates by any multicast sessions to construct low cost trees, thus eliminating the lengthy processing time needed to compute and reserve these paths wherever rearrangement occurs. However, we show that if assigning reservation channels uniformly (UA-PRDCDMM) may result in resource inefficiency and poor utilization. Thus we introduce non-uniform assignment scheme (Non-UA PRDCDMM) which only assign critical links with reservation channels. We also introduce a *CC-DC Performance Index* for measuring the dynamic multicast performance of an algorithm by considering both tree cost and disruptions in totality.

Chapter 4

The results obtained from our simulation studies show that:

- (a) HCTR-F yields almost the same or better performance (in terms of cost) than the pseudo-optimal $DCMT_{CD}$ proposed in [KPP93] in computing static delay constrained minimum cost tree. This justifies the use of HCTR-F in per update rearrangement in PRDCDMM to obtain low cost trees for delay constrained dynamic multicast.
- (b) HCTR-F requires much lesser time than $DCMT_{CD}$ to compute the low cost tree.
- (c) UA PRDCDMM performs better than CRCDM due to its ability to minimize the competing effect of cost and disruptions minimization.
- (d) UA PRDCDMM incurs much fewer disruptions compared to CRCDM due to its disruption control mechanism.
- (e) PRDCDMM performs faster rearrangement per update than CRCDM (in the order of 100 times) and hence able to reduce the processing time to add or drop a member.
- (f) Non-UA PRDCDMM with appropriate WLR performs as well as UA PRDCDMM with much higher resource utilization and fewer redundant links (higher efficiency).
- (g) PRDCDMM provides two tunable parameters -- DL for controlling disruptions per update in UA PRDCDMM and Non-UA PRDCDMM, and WLR for controlling the resource utilization and redundant links in Non-UA PRDCDMM. This gives PRDCDMM the flexibility to provide different disruption QoS without incurring too much resource.
- (h) PRDCDMM can accommodate applications that do not allow any disruptions to the existing connections by setting DL to zero as well as

Chapter 4

applications which are not concerned with disruptions by setting DL to 100%.

- (i) PRDCDMM is able to achieve similar CC-DC Performance Index as the improved CRCDM proposed in [RM06] with a much smaller average rearrangement processing time (average tree setup time) and lower maximum number of disruptions per update.

In this chapter, we propose a new strategy of rearrangement for dynamic multicasting and show that it is more flexible and able to construct lower cost trees with lower number of disruptions per update and faster rearrangement processing time compared to CRCDM proposed in [RMM99] and the improved version described in [RM06]. We also show that higher resource utilization with fewer redundant links can be achieved by modifying the reserve channels assignment strategy appropriately. Also, as the processing time of each rearrangement in PRDCDMM is much shorter, there will be lesser chance for the next update request to arrive before the rearrangement is completed, and this helps to minimize race conditions.

Chapter 5

Optimal Placement of MC Nodes and WCC Nodes in M-OVPN

5.1 Introduction

With the increased usage of the Internet for multicast applications, the drive towards a larger capacity optical Internet is inevitable. Due to the availability of the Internet in most parts of the world, subsidiaries within the same parent organization have to communicate with each other and with the parent organization via a secured and virtually private network. In addition, employees of each organization who travel to other parts of the world for business must also find a way to communicate securely to the organization. Multicast Optical Virtual Private Network (M-OVPN) is an attractive solution for providing a cost effective and secured means of communication for high-bandwidth multimedia applications within an organization.

In M-OVPN, an all-optical or transparent optical network is used as the backbone to carry large volume of unicast and multicast traffic between peer-to-peer site and peer-to-multiple- peer sites. Each wavelength channel in each link in the transparent optical network is able to support a capacity of up to 40 GB/s and beyond with today's technology. Besides able to provide enormous capacity on each wavelength channel, additional benefits such as expandable

Chapter 5

bandwidth and end-to-end delay reduction can be achieved through the use of wavelength-division-multiplexing (WDM) technique on the M-OVPN. A transparent optical network is highly desirable for achieving high efficiency within the VPN with large volume of traffic exchanged. It is shown in [SM99] that by implementing multicast trees in an optical network, much saving can be achieved in terms of network-wide average packet hop distance and total number of transceivers used. To implement such a network where the entire routing occurs at the optical layer, multicast-capable (MC) nodes and wavelength conversion-capable (WCC) nodes are required [SB99]. As mentioned in chapter 2, each MC node is assumed to have the required optimal number of power splitters [Muk97] to split incoming signal into multiple outgoing signals and transmits them out to the fibers of the outgoing ports of the node. Likewise for a WCC node, it is assumed to have the required optimal number of wavelength converters to convert an incoming signal's wavelength to another wavelength within the range of bandwidths used for that communication.

It was mentioned in chapter 1 as well as in [MZQ98] and [AD00] that a MC node is expensive to build due to the large amount of power amplification required and high level of difficulty in the fabrication of large number of power splitters required per node. The WCC node with multiple converters is also expensive to build. Hence, it is desirable to limit the number of MC nodes and WCC nodes within the backbone in M-OVPN to achieve cost efficiency. At the same time, we have to maintain an acceptable blocking probability so as to ensure that the QoS requirement can be met. In order to achieve these two

Chapter 5

objectives, we have to find a way to optimally place MC nodes and WCC nodes. Similar to the definition adopted in most existing literatures, placement is deemed optimal when the overall blocking probability is the same as the scenario when all the nodes in the backbone are MC and WCC nodes.

5.1.1 Our Results

In this chapter, we study the optimal placement of MC and WCC nodes in the optical backbone of M-OVPN. We first develop an analytical model for sparse-splitting and sparse-wavelength conversion configuration to compute the approximate blocking probability and use the results to determine the optimal placements. In addition, we propose heuristics based on the “Removal” concept and show that they can determine the optimal placement of MC nodes and WCC nodes more efficiently than existing work based on the “Add” concept without degrading the quality of the blocking performance. We then combine the individual heuristics for MC and WCC node placements and refer to them as *hybrid-mode* placement to determine the placement of both MC and WCC nodes jointly. Results show that the *hybrid-mode* placement is able to achieve optimal blocking performance with only a fraction of the nodes in the backbone as MC and/or WCC nodes. In addition, we introduce the *dual-mode* placement whereby each node is considered to be an MC node as well as a WCC node at the same time, and show that the *dual-mode* placement achieves almost similar blocking performance as the *hybrid-mode* placement but with a much lower computation complexity.

Chapter 5

5.1.2 Previous Works

Although there are many studies on WCC node optimal placement, as can be seen from the review in chapter 2, only [AS00] and [CLC03] consider the more complex blocking scenario which is closer to the real world. We will briefly discuss only these two studies as a prelude to our proposed heuristic WCC node placement algorithm which takes blocking into consideration. In [AS00], a heuristic which we shall refer to as *total outgoing traffic unicast* (TOTU) for convenience was proposed. In this heuristic, WCC nodes are placed based on the total traffic load directed out from each node. The total outgoing traffic of each node comprises of those originated from the node and those in transit to the other receiver nodes of a multicast session. The argument articulated in [AS00] was that the higher the total outgoing traffic from an arbitrary node u , the more likely that a particular outgoing wavelength is not available and hence requires node u to be a WCC node to convert an incoming wavelength wl_{in} into any available outgoing wavelength wl_{out} . If the node u is not a WCC node, the receivers expecting the data carried by wl_{in} via node u will fail to receive any data and hence increase the overall receiver blocking probability. In [CLC03], the authors linked the WCC placement with the RWA strategy used to minimize the average receiver blocking probability as well as the total number of WCC nodes required. They proposed to use the MBPF placement heuristic together with *Fixed-Alternate Routing with First-Fit Wavelength Assignment* (FAR-FF RWA), and the WMSL placement heuristic with the *Least-Load Routing with first-Fit Wavelength Assignment* (LLR-FF RWA). It should be noted that both [AS00] and [CLC03] solve the placement problem in the context of unicast traffic. As mentioned in chapter 2, a WCC

Chapter 5

node performs wavelength conversion which is a unicast operation while the responsibility of splitting the incoming signal lies entirely with the MC node, therefore the heuristics proposed for the WCC optimal placement problem for unicast traffic can be easily applied for multicast traffic.

It was proposed in [Ali02] that the optimal placement of MC nodes be carried out with the *Multicast-Add* and *Random-Add* heuristics based on the “Add” concept. For *Multicast-Add*, all nodes are assumed to be Multicast-Incapable (MI) initially and in each step of the placement, the best node is selected to be made an MC. The best node is found in the following way: A selected MI node is assumed to be MC and the routing trees for a set of incoming multicast sessions are constructed based on the new temporary configuration of MC nodes using the minimum path heuristic. The approximate average receiver blocking probability is then computed by means of an analytical model for the set of multicast sessions using the constructed routing trees for traffic delivery. The whole process is repeated for all MI nodes using the same set of incoming multicast sessions and routing heuristic and the one with the minimum average receiver blocking probability is deemed the best node. The next step of placement is then carried out on the updated new configuration of MC nodes until all the required numbers of MC nodes are placed. For *Random-Add*, a group of MI nodes are randomly selected to be MC. These two heuristics are compared in [Ali02] and *Multicast-Add* is shown to achieve better placement results than *Random-Add*.

Chapter 5

It has been shown in [YL03] and [CY05] that it is desirable to solve the MC node and WCC node optimal placement problems jointly. Optimal algorithms such as MILP and ILP were employed in [YL03] and [CY05] respectively for the optimal placement of the MC node and WCC node jointly. Although [YL03] and [CY05] showed the benefits of solving both problems jointly, the MILP and ILP algorithms used are too computation intensive. Although there is also enough evidence in existing works to show that MC node and WCC node placements can be efficiently solved individually while assuming full wavelength conversion and full splitting respectively, there is a lack of research in considering the impact of both sparse-splitting and sparse-wavelength conversion to the blocking performance of optical networks. In this chapter, we propose a much faster and flexible optimal placement heuristic for a wide range of physical constraints (in terms of number of MC nodes and WCC nodes allowed) without compromising the blocking performance.

5.1.3 Chapter's Outline

The rest of this chapter is organized as follows. In section 5.2, an overview of the M-OVPN architecture is provided and the optical multicasting and wavelength conversion activities within the M-OVPN discussed. In section 5.3, an analytical model to approximate the average blocking probability at each receiver under sparse-splitting and sparse-wavelength conversion conditions is presented. In section 5.4, our proposed heuristics based on “Removal” concept to place MC nodes and WCC nodes optimally are presented and discussed. In section 5.5, simulation results are presented and discussed. Finally, conclusions are drawn in section 5.6.

5.2 M-OVPN Architecture

A Virtual Private Network (VPN) is a logical network established on top of a shared physical (backbone) network to realize a virtual dedicated network. A Multicast-Optical Virtual Private Network (M-OVPN) is a VPN which supports multicasting inherently over an optical fiber based physical network. As shown in figure 5-1, each VPN site is connected to the underlying physical network by means of a CE-PE pair and data is sent from one site to the other via logical paths and trees established within the optical network.

The network can be modeled as a graph $G(V, E)$ where V is the set of nodes and E is the set of directed links. Each node within the backbone network consists of an optical cross-connect and each edge (link) node consists of an extra local station for optical add/drop. The set of cross-connects and local stations are represented by CC and LS respectively. Stations are labeled from 1 to $|LS|$ and cross-connects from $(|LS|+1)$ to $(|LS|+|CC|)$ such that $V = LS \cup CC$. Two directed fiber links in opposite directions are assumed to connect any two cross-connects or a cross-connect and a station, and all fibers are assumed to have the same wavelength capacity of W wavelength channels. Each station is assumed to be able to transmit on all wavelengths at the same time using W lasers and also able to receive on all wavelengths using W filters, each tuned to a specific wavelength.

Chapter 5

Figure 5-1: An Example of M-OVPN.

An example of unicast and multicast communication within the M-OVPN is given in figure 5-1. For unicast communication, data is sent from site 1 to site 3 via node A within the optical backbone. Node A is assumed to be WCC, hence wavelength capacity is the only constraint for the lightpath to be established between VPN site 1 and site 2. If node A is not WCC, an additional wavelength continuity constraint for the entire lightpath will be imposed. For multicast communication, site 2 is to send data to its receivers in site 4 and site 5 via a delivery tree within the backbone provided that node B is an MC. For simplicity, we use only one node (node B) within the backbone to form the tree although it is noted that in general the tree is likely to consist of more than one node. If node B is not an MC, then two separate lightpaths have to be constructed to send the data. The worst case scenario is when node B is also not a WCC. When this happens, the two lightpaths have to satisfy the wavelength continuity constraints in order to send the data successfully. Thus, it can be observed that it is best for all nodes to be MC and WCC; however, such a configuration is expensive and not practical. Hence, the objective is to locate optimal placement for a limited number of MC and WCC nodes in M-OVPN such that the blocking performance is not degraded.

5.3 Approximate Blocking Model for Sparse-Splitting and Sparse-Wavelength Conversion Network

Blocking probability has to be computed to verify if a particular placement is optimal. Hence, we extended the analytical model in [Ali02] for sparse-splitting environment to that of sparse-splitting and sparse-wavelength conversion environment. Table 5-1 lists the notations used in this chapter.

Table 5-1: List of Notations used in this Chapter.

Q	Set of multicast sessions in the network where $Q = \{R_1, R_2, \dots, R_n\}$ and $R_i = \{s_i, D_i\}$, s_i being the source and D_i being the set of destinations.
H_i	Set of suboptimal trees to carry the traffic of session R_i .
$H_{i,j}$	The j^{th} tree of session R_i .
$A[1 \dots Q]$	A vector representing the routing tree used for each session in the set Q , i.e. $Y[i] \in H_i$.
$mc[1 \dots N]$	A vector representing the multicast allocation, where $mc[\phi] = 1$ if node ϕ is MC, $=0$ if otherwise.
$C[1 \dots N]$	A vector representing the wavelength converters allocation, where $C[\phi] = 1$ if node ϕ is WCC, $=0$ if otherwise.
k	Size of MC nodes.
c	Size of WCC nodes.
$\psi = \{k, c\}$	Set of MC and WCC nodes configuration.
$T^{R_i}(s, R')$	The path used to reach destination $R' \in D_{R_i}$ of session R_i using the routing trees in A .
$Tc_{i,j}[R']$	Connection status. $Tc_{i,j}[R'] = 0$ if the destination R' of session R_i using j^{th} tree cannot be reached due to absence of MC nodes, and $=1$ if otherwise.

In this chapter, we investigate the impact of MC and WCC node placements on the overall receiver (destination) blocking probability for a set of incoming multicast sessions. Although the average session blocking performance can be considered instead of receiver blocking performance, it is more straight-forward to see the impact of the MC and WCC nodes within the tree paths to individual receiver nodes compared to when a session is deemed to be blocked if any of its receiver nodes is blocked. Thus, we shall henceforth refer to blocking

Chapter 5

probability as the average receiver blocking performance for a set of incoming multicast sessions.

5.3.1 Analytical Model

We consider a network with dynamic traffic loading. Let W be the maximum number of wavelengths per fiber (i.e. the wavelength number), $1/\mu$ be the average holding time of a multicast session and λ_i be the corresponding session arrival request rate on link i where link i is a branch of the session's tree. To simplify our notation, time is normalized such that the average service time is unity (i.e. $1/\mu = 1$) so that the load for each branch with the session tree is $L_i = \lambda_i$. We assume the arrival request follows a Poisson process and the holding time is exponentially distributed with unit mean.

For a *Non-wavelength-Continuous* (NWC) network, it has been shown in [Gir90] that it can be studied by directly extending the approach used for analyzing traditional circuit networks. Hence, the blocking probability of link f with W wavelengths is given by the Erlang loss formula:

$$B_f = \frac{L_f^W / W!}{\sum_{j=0}^W L_f^j / j!} \quad (5.1)$$

Since blocking in NWC network occurs if and only if all the wavelengths in any of the links within the path used to relay data to a receiver (destination) node are in use, the blocking probability for a receiver in a session can be expressed as:

$$B_{R'}^R = 1 - \prod_{f \in T^R(s, R')} (1 - B_f) \quad (5.2)$$

Chapter 5

For a network with only a fraction of the nodes to be MC (sparse-splitting configuration), some of the receiver nodes within a multicast session may not be able to receive multicast traffic due to the absence of MC nodes. For such cases, a direct unicast connection is established to these receiver nodes to ensure that the blocking probability is kept reasonably low [Ali02]. Thus, the total offered load of each link is contributed by both the multicast and unicast connections. It has been shown in [Kel86] that the offered load of a link contributed by a direct connection can be approximated with the following expression:

$$L_{f,DC} = \sum_{R \in Q} a_{f,T^R(s,R')} e_R \frac{1 - B_{R'}^R}{1 - B_f} \quad (5.3)$$

where $a_{f,T(s,R')}$ is the link-path incidence matrix defined as:

$$a_{f,T^R(s,R')} = \begin{cases} 1, & f \in T^R(s,R') \\ 0, & \text{otherwise} \end{cases}$$

The term $\sum_{R \in Q} a_{f,T^R(s,R')} e_R (1 - B_{R'}^R)$ represents the load carried by link f , and is smaller than the offered load by a factor of $(1 - B_f)$ due to blocking contributed by insufficient wavelengths in any of the link within the path. The offered load contributed by multicast traffic can be expressed as:

$$L_{f,MC} = \sum_{R \in Q} a_{f,T^R(s,R')} e_R \frac{1 - B_f^{\min,R}}{1 - B_f} \quad (5.4)$$

where $B_f^{\min,R}$ is the minimum blocking probability among all the tree paths using link f contributed by session R . $B_f^{\min,R}$ can be expressed as

$$B_f^{\min,R} = \min_{f \in T^R(s,R')} \{B_{R'}^R\} \quad (5.5)$$

Chapter 5

From (5.3) and (5.4), the total offered load on link f can be computed with the expression:

$$L_f = L_{f,DC} + L_{f,MC} \quad (5.6)$$

We have so far considered an NWC network where all nodes in the network are assumed to be WCC and hence no wavelength continuity constraint is imposed. For a wavelength continuous network where none of the network nodes is WCC, the wavelength continuity constraint has to be imposed. To model a network with wavelength continuity constraint, we can adopt the *birth-and-death* process approach suggested in [Ali02].

Let X_f be the random variable denoting the number of idle wavelengths on link f at steady state. We assume that X_f are mutually independent for different f . Although this assumption may not be strictly true, it simplifies our analysis and will lead to reasonably accurate results, as pointed out in [SBCLC01]. We also assume that link occupancies are independent with for different links. Let $q_f^w(j) = \Pr[X_f = j]$, $\forall j = 0, 1, \dots, W$ be the idle capacity distribution on link f . We make the approximation that when there are j idle wavelengths on link f , the time elapses until the next call is set up is independent of the number of idle wavelengths on link f and is exponentially distributed with a mean of α_f^w . This is also the call setup rate on link f when j wavelengths are free on the link. According to [Ali02] and [SBCLC01], the number of unused wavelengths on link f can be viewed as a *birth-and-death* process and the distribution is given by:

Chapter 5

$$q_f^\psi(j) = \left[\frac{W(W-1)\dots(W-j+1)}{(\alpha_f^\psi)^j} \right] q_f^\psi(0) \quad (5.7)$$

where

$$q_f^\psi(0) = \left[1 + \sum_{j=1}^W \frac{W(W-1)\dots(W-j+1)}{(\alpha_f^\psi)^j} \right]^{-1} \quad (5.8)$$

The call setup rate of α_f^ψ on link f is actually the total offered load on the link. Since we have mentioned earlier that direct connections are used to connect those receiver nodes which cannot be reached via multicast tree due to absence of MC nodes at some parts of the network, the total offered load on link f is contributed by both the multicast and direct connection traffic and is expressed as:

$$\alpha_f^\psi = \alpha_{f,DC}^\psi + \alpha_{f,MC}^\psi \quad (5.9)$$

where $\alpha_{f,MC}^\psi$ and $\alpha_{f,DC}^\psi$ are the call setup rate on link f for multicast and direct connection traffic respectively.

By replacing B_f with $q_f^\psi(0)$ in (5.3) and (5.4), we obtain the following expressions of $\alpha_{f,DC}^\psi$ and $\alpha_{f,MC}^\psi$.

$$\alpha_{f,DC}^\psi = \sum_{R \in Q} a_{f,T^R(s,R)} e_R \frac{1 - B_{R'}^R}{1 - q_f^\psi(0)} \quad (5.10)$$

$$\alpha_{f,MC}^\psi = \sum_{R \in Q'} a_{f,T^R(s,R')} e_R \frac{1 - B_f^{\min,R}}{1 - q_f^\psi(0)} \quad (5.11)$$

where $B_f^{\min,R}$ is the same as in (5.5).

Denote t_f as the probability that a particular wavelength is free on link f . t_f can be expressed as:

Chapter 5

$$t_f = q_f^w(1) \frac{1}{W} + q_f^w(2) \frac{2}{W} + \dots + q_f^w(W) = \sum_{d=1}^W q_f^w(d) \frac{d}{W} \quad (5.12)$$

where $q_f^w(d) \frac{d}{W}$ denotes the probability that a particular wavelength is free on link f when there are d free wavelengths on this link.

Let $T^R(s, R') = \{l_1, \dots, l_z\}$ and $SP^R(s, R') = \{m_1, \dots, m_v\}$ be the routed path and shortest path, respectively, to destination R' in session R , and $\{\omega_1, \dots, \omega_j\}$ be the set of idle wavelengths on link f . Under the independence assumption for traffic on different links and for $j \neq 0$; the probability that a receiver node can be reached by means of the multicast tree, given that j wavelengths on link f is available, is given by:

$$\Pr[X_{T^R(s, R')} > 0 \mid X_f = j] = 1 - \prod_{h=1}^j \left(1 - \prod_{\substack{a=1 \\ a \neq f}}^z t_{l_a} \right) = 1 - \left(1 - \prod_{\substack{a=1 \\ a \neq f}}^z t_{l_a} \right)^j \quad (5.13)$$

Therefore, the probability that the path in the multicast tree to destination R' of session R will be blocked is given by:

$$B_{R'}^R = 1 - \Pr[X_{T^R(s, R')} > 0] = \left(1 - \prod_{a=1}^z t_{l_a} \right)^W \quad (5.14)$$

The probability that the destination is reached via a direct connection is given by:

$$\Pr[X_{SP^R(s, R')} > 0 \mid X_f = j] = 1 - \prod_{h=1}^j \left(1 - \prod_{\substack{a=1 \\ a \neq f}}^v t_{m_a} \right) = 1 - \left(1 - \prod_{\substack{a=1 \\ a \neq f}}^v t_{m_a} \right)^j \quad (5.15)$$

The corresponding blocking probability for destination R' of session R is

Chapter 5

$$B_{R'}^R = 1 - \Pr[X_{SP^R(s,R')} > 0] = \left(1 - \prod_{a=1}^v t_{m_a}\right)^W \quad (5.16)$$

For both non-wavelength continuous and wavelength continuous networks, the average receiver (destination) blocking probability for a given set of multicast sessions Q can be obtained using the following expression:

$$BP_{avg} = \frac{\sum_{R \in Q} \sum_{R' \in D_R} e_R B_{R'}^R}{\sum_{R \in Q} e_R |D_R|} \quad (5.17)$$

So far we have derived the expressions for computing the approximate receiver blocking probability for a given set of multicast sessions for both non-wavelength continuous and wavelength continuous networks. These two networks are not operating very efficiently. For non-wavelength continuous networks, all the network nodes are WCC and such a configuration increases the cost and complexity of each node. For wavelength continuous networks where none of the network nodes is WCC, the blocking probability will obviously be significantly higher when the number of wavelengths that can be used per fiber is small. Thus, a solution in between these two configurations which we termed partial-wavelength continuous (PWC) network will be most desirable. Note that a PWC network is also referred to as a sparse-wavelength conversion network in some literature. In a PWC network, only a fraction of the network nodes are WCC; hence, some parts of the network will have the wavelength continuity constraint imposed and thus the term partial-wavelength continuous.

Due to the insertion loss contributed by splitters, it is desirable to use sparse splitting together with sparse-wavelength conversion where only a fraction of

Chapter 5

the network nodes are MC or WCC or both. We assume for each WCC node, each incoming wavelength can be converted to any other outgoing wavelength within the color band used for that fiber. Since not all nodes within the path to destination R' of session R have wavelength conversion capabilities, we can model each path as a series of segments, whereby each segment consists of a series of links with only its ingress and egress nodes, apart from the path ingress and egress to the WCC as shown in figure 5-2 below. It is possible that a segment consists of only a single link when both ingress and egress nodes are WCC. We define a segmented path as a path where any two of its nodes are connected by a segment instead of a single link.

Figure 5-2: An Example of Multi-Segments in a Tree/Path.

Let $Seg_b^R(s, R') = \{l_1, \dots, l_z\}$ be the b^{th} segment of the segmented path to destination R' of session R . This path may be a direct connection or part of the routing tree paths depending on the availability of MC nodes within this path. Since the wavelength continuity constraint holds within the links in each segment, each segment can be regarded as a lightpath with ingress node of the segment as the source and egress node as the destination. For simplicity, we label them as a set $Y = \{a, b\} = \{\text{Ingress}, \text{Egress}\}$. Thus, the new definition of $Seg_b^R(s, R')$ is modified to $Seg_b^R(s, R', Y)$.

Chapter 5

As mentioned earlier, the number of unused wavelengths on link f in a particular segment b can be viewed as a birth-and-death process and equations (5.7) to (5.16) can be used to find the blocking probability of each segment that can be expressed as

$$B_{Seg_b^R(s, R', Y)} = 1 - \Pr[X_{Seg_b^R(s, R', X)} > 0] = \left(1 - \prod_{a=1}^{z'} t_{l_a}\right)^W \quad (5.18)$$

Since the ingress and egress nodes of each segment are WCC, wavelength can be converted in between segments and a segmented path can be modeled as a non-wavelength continuity path. Following the assumption of traffic in different links are independent, the blocking probability to a destination R' of session R using a segmented path can be obtained using (5.2) except that instead of computing blocking probability of each link within the path, we now consider computing the blocking probability of each segment within the segmented path. Thus, we derive the expression for the blocking probability to a destination R' of session R using a segmented path as:

$$B_{R'}^R = 1 - \prod_{Seg_b^R(s, R', Y) \in T^R(s, R')} (1 - B_{Seg_b^R(s, R', Y)}) \quad (5.19)$$

The average receiver blocking probability for a given set of incoming multicast sessions in the sparse-splitting and partial wavelength continuity network can then be obtained using (5.17).

5.3.2 Numerical Method

For each configuration of MC and WCC nodes in the network given by the expression $\mathcal{W} = \{k, c\}$, we can compute the approximate average receiver blocking probability for a set of incoming multicast sessions by using the *relaxation method* on the analytical model we presented earlier. We shall term

Chapter 5

this process as $Relax-BP_{avg}$. For this process, we perform iterations to solve for the approximate average receiver blocking probability until the result converges to a reasonably small factor. In this case, we set this factor to $\varepsilon = 10^{-6}$ since it was shown in many papers to be an appropriate value. Figure 5-3 shows the pseudo code of $Relax-BP_{avg}$ for computing the average receiver blocking probability for a set of incoming multicast sessions for a given configuration of MC and WCC nodes using the *relaxation method* on the analytical model presented in section 5.3.1.

In step 1, the iteration index and the connection setup rate on all links are initialized to 1, and the blocking probability to each destination and the overall average receiver blocking probability to 0. In step 2, equations (5.7) and (5.8) are used to compute $q_f^{w,iter}(\cdot)$ for all the links. In step 3, the setup rates on all the links are computed using equations (5.9) to (5.11). In step 4, the blocking probability to all the receivers in all the sessions are computed using equations (5.12), (5.18) and (5.19). In step 5, the overall average receiver blocking probability is computed using equation (5.17). In step 6, if the difference in the computed result between iteration is greater than $\varepsilon = 10^{-6}$, step 2 to step 6 are to be repeated and the iteration index increased by 1 notch. Otherwise, we stop the algorithm and the result obtained is the average receiver blocking probability for a set of incoming multicast sessions to a network with a given configuration of MC and WCC nodes.

Chapter 5

BEGIN

Step1: Initialize: Set $BP_{avg}^0 = 0$, $iter = 1$.

Set $B_{R'}^{R,0} = 0 \forall R \in Q, \forall R' \in D_R, \alpha_f^{w,0} = 1, \forall f \in E$.

Step2: Find $q_f^{w,iter}(\cdot), \forall f \in E$ using (5.7) and (5.8)

Step3: Find $\alpha_f^{w,iter}, \forall f \in E$ using (5.9) to (5.11)

Step4: Find $B_{R'}^{R,iter}, \forall R \in Q, \forall R' \in D_R$ using (5.12), (5.18) and (5.19).

Step5: Find BP_{avg}^{iter} using (5.17).

Step6: **IF** $|BP_{avg}^{iter} - BP_{avg}^{iter-1}| < \varepsilon$ **THEN**

Stop algorithm and BP_{avg}^{iter} is the answer.

ELSE

$iter = iter + 1$, Repeat Step 2 to Step 6.

END IF

END

Figure 5-3: Pseudo Code of $Relax-BP_{avg}$.

It should be noted that to find the optimal solution to the problem, one can enumerate all the different solutions in a brute-force way and report the best. But there are substantial number of different solutions to be checked. Hence, we minimize the search space by assuming that the routing trees for each session have already been obtained using Minimum Path Heuristic or other optimal tree routing algorithms. This is a reasonable assumption as our main objective is to find the best placement for a given number of MC and WCC nodes without degrading the blocking performance, and not to find the minimum number of such nodes used in the network. Low network cost in terms of bandwidth (number of edges per tree) takes precedence over the minimum number of MC and WCC nodes required. In addition, it has been shown in numerous existing works that a small number of MC and WCC nodes are sufficient to achieve optimal blocking performance. Hence, it is not necessary to find the minimum numbers of MC and WCC nodes required but

Chapter 5

very critical to find the optimal placement for a given numbers of MC and WCC nodes allowed for use in the network.

5.4 Proposed Placement Heuristics

This section describes our proposed placement heuristics based on the “Removal” concept for a given numbers of MC and WCC nodes in M-OVPN. The average receiver blocking probability computed in section 5.3 will be used on all the heuristics and the objective is to place the MC and WCC nodes such that the blocking probability is as small as possible.

5.4.1 MC-Removal Heuristic

For this heuristic, it is assumed that at the onset, all network nodes in M-OVPN are MC nodes. Then for each step of the heuristic, the multicast capabilities of MC nodes are removed one at a time to compute their MC-Criticality weight. The MC-Criticality weight here refers to the degree of connectivity between the average receiver blocking probability and the multicast capability requirement in that node. The MC-Criticality weight of a MC node u is computed by taking the difference in the blocking performances between all MC node configurations and the one when multicast capability of MC node u is removed. After the weights have been computed, they are ranked with the least weight on top. Hence, for a given y number of MC nodes and a network size of n nodes, the top $|n-y|$ nodes will have their multicast capabilities removed. Since the weight is only computed once, the worst-case complexity for this heuristic is $O(n^2)$. This is in contrast to the *Add-heuristic* in [Ali02] which has a worst-case

Chapter 5

complexity of $O(n^3)$ as the entire iteration process has to be repeated for each placement.

The “*Removal*” concept and the “*Add*” concept appear to be opposite in operation. However, the implication is entirely different and hence different in efficiency. Although one may argue that placement of MC and WCC nodes is a static issue before traffic routing is carried out, and carried out only once, it is possible for MC and WCC nodes to be placed dynamically in future where splitters and converters can be programmed to be switched on or off in a particular node to achieve better performance and higher efficiency in terms of usage and cost. Hence, the time processing and the complexity issue of the placement problem are as important as the blocking performance and should not be ignored.

5.4.2 WCC-Removal Heuristic

This heuristic is the same as *MC-Removal* except that now the wavelength conversion capabilities of WCC nodes are removed. WCC-Criticality weights of each node is computed, and that we assume all nodes in M-OVPN are MC at the onset. Again, the worst-case complexity of such heuristic is $O(n^2)$. The WCC-Criticality weight here means the degree of connectivity between the average receiver blocking probability and the requirement of wavelength conversion capability in that node. The way of computing this weight follows the same reasoning as the one for computing MC-Criticality weight except that now WCC nodes are involved instead of MC nodes.

Chapter 5

5.4.3 MWC-Removal Heuristic (Hybrid Mode)

This heuristic performs the removal of multicast capabilities in MC nodes first before using the updated configuration to remove the wavelength conversion capabilities in WCC nodes in order to obtain the final configuration placement. Since the first phase uses *MC-Removal* heuristic whose worst-case complexity is $O(n^2)$, and the second phase uses *WCC-Removal* heuristic which also has a worst-case complexity of $O(n^2)$, the worst-case complexity for this hybrid mode heuristic is $2O(n^2)$.

5.4.4 WMC-Removal Heuristic (Hybrid Mode)

This heuristic performs the removal of wavelength conversion capabilities in WCC nodes first before using the updated configuration to remove the multicast capabilities in MC nodes in order to obtain the final configuration placement. Since *WCC-Removal* and *MC-Removal* heuristics are employed in phases 1 and 2 respectively, the *WMC-Removal* heuristic has the worst-case complexity of $2O(n^2)$ as well.

5.4.5 Dual-Removal Heuristic (Dual Mode)

This heuristic removes both MC and WCC capabilities from a specific node at the same time. In other words, each node is either having dual capabilities of MC and WCC or none at all. For this heuristic, it is assumed that at the onset, all network nodes in M-OVPN are both MC and WCC. We shall refer to these nodes to be both MC and WCC as dual nodes. Then, for each step of this heuristic, multicast and wavelength conversion capabilities are removed from dual nodes one at a time to compute their Dual-Criticality weight. The Dual-

Chapter 5

Criticality weight is defined as the degree of connectivity between the average receiver blocking probability and the requirement of multicast and wavelength conversion capabilities in that node. The Dual-Criticality weight of a dual node u is computed by taking the difference in the blocking performances between all dual node configurations and the one when dual node u is removed with its multicast and wavelength conversion capabilities. After the weights have been computed, they are ranked with the least weight on top. Hence, for a given y number of dual nodes (i.e. number of MC nodes = number of WCC nodes) and a network size of n nodes, the top $|n-y|$ nodes will have their multicast and wavelength conversion capabilities removed. Since the placements of both kinds of nodes are considered jointly, the worst-case complexity for this heuristic is only $O(n^2)$. Figure 5-4 presents the pseudo code for all the proposed heuristics based on the “Removal” concept.

BEGIN

Step1: Initialize: All nodes are MC and WCC

Step2: If MC-Removal Heuristic, compute MC-Criticality Weights - (1).

If WCC-Removal Heuristic, compute WCC-Criticality Weights - (2).

If MWC-Removal Heuristic, perform (1) then (2) of step 2.

If WMC-Removal Heuristic, perform (2) then (1) of step 2.

If Dual-Removal Heuristic, compute Dual-Criticality Weights.

Step3: Based on $c = \{\psi, \eta\}$, remove capabilities of those MC and WCC nodes with least weight.

Step4: The remaining MC and WCC nodes give our placement solution.

END

Figure 5-4: Pseudo Code of all Proposed “Removal”.

5.5 Experiments and Results

Figure 5-5: NSFNET. Each of the nodes can be a provider's edge in M-OVPN.

For our simulation studies, the 14-node NSFNET is used as the optical backbone network in M-OVPN. 10 multicast sessions are randomly generated and we assume that the traffic load of each session is the same. Arrival requests are assumed to follow a Poisson process and the holding time is exponentially distributed with unit mean. We assume a capacity of three wavelengths per fiber for all our experiments unless specified otherwise. It is obvious that the placement problem is network topology dependent. However, our placement heuristics and the associated blocking probability model are rather general and are applicable in any networks. Each data point on each result curve is the average of 1000 repeated experiments.

The MC and WCC-nodes placement problems are generally offline problems. This means that the placement is carried out offline each time for a set of traffic. Thus, the placement algorithms proposed can still be re-run offline whenever any nodes joining/leaving the group. However, the placement

Chapter 5

algorithm will need to be highly efficient because a very efficient algorithm will place the MC and WCC-nodes within a short time and hence do not incur too much additional transmission delay to other existing nodes whenever nodes join or leave the group. This will be an interesting future work since no literatures have investigated this problem before.

Table 5-2 shows a sample of 10 multicast sessions randomly generated, and their respective routing trees obtained by means of one of the optimal tree algorithms mentioned earlier. By performing the *MC-Removal* heuristic, the MC-Criticality weight for each node is computed and shown in Table 5-3. The nodes are ranked in such a way that the one with the heaviest weight is ranked last. Depending on the number of MC nodes that are allowed to be allocated, the corresponding number of MC nodes will have their multicast capabilities removed. Table 5-4 shows the MC nodes placement configuration for all possible number of allowed MC nodes. The arrival rate of each multicast session is given by e_R . We assume all sessions to have the same e_R .

Table 5-2: Sample of 10 Multicast Sessions with Corresponding Routing Trees.

s_i	D_i	$H_{i,1}$
6	{14,13,12}	6→14,14→13,14→12
4	{13,14,9}	4→11→13,11→12→14,12→9
14	{6,3,10}	14→6,6→3,6→10
8	{11,14,12}	8→9→12,9→13→11,13→14
12	{14,6,13}	12→14,14→6,14→13
13	{9,10,12}	13→9,9→10,9→12
3	{14,13,10}	3→6→14,6→10,14→13
9	{12,11,14}	9→12,12→11,12→14
5	{6,1,9}	5→6,6→3→1,6→10→9
11	{13,9,14}	11→13,13→9,13→14

Chapter 5

Table 5-3: MC-Criticality Weight of each Node in Backbone.

N_Z (node)	$MC-W_{N_Z} (10^{-3})$ (weights)	R_N (Rank)
1	0	1
2	0	2
3	0	3
4	0	4
5	0	5
6	0.7661	14
7	0	6
8	0	7
9	0.4144	11
10	0	8
11	0.0423	9
12	0.4425	12
13	0.2233	10
14	0.5631	13

Table 5-4: MC Nodes Placement Configuration for All Possible Numbers of Allowed MC Nodes in Backbone.

e_R	k	MC nodes placement configuration {set of nodes which are made MC}
0.05	1	{6}
0.05	2	{6,14}
0.05	3	{6,14,12}
0.05	4	{6,14,12,9}
0.05	5	{6,14,12,9,13}
0.05	6	{6,14,12,9,13,11}
0.05	7	{6,14,12,9,13,11,10}
0.05	8	{6,14,12,9,13,11,10,8}
0.05	9	{6,14,12,9,13,11,10,8,7}
0.05	10	{6,14,12,9,13,11,10,8,7,5}
0.05	11	{6,14,12,9,13,11,10,8,7,5,4}
0.05	12	{6,14,12,9,13,11,10,8,7,5,4,3}
0.05	13	{6,14,12,9,13,11,10,8,7,5,4,3,2}
0.05	14	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}

Chapter 5

Figure 5-6: MC-Removal Heuristic for Optimal MC nodes Placement (Using $e_R = 0.05$).

Figure 5-6 shows the blocking performance against the number of MC nodes used for *MC-Removal* heuristic and optimal configuration (all nodes MC and WCC). It should be noted that all nodes in the backbone network in this case are assumed to be WCC. It is observed in figure 5-6 that for the set of optimal trees computed for a set of incoming multicast sessions, the minimum number of MC nodes required to achieve the minimum blocking possible is 6 out of 14. The exponential increase in the blocking probability as the number of MC nodes used decreases clearly indicates that the placement is optimal for each size of MC nodes used. This means that for each given number of MC nodes allowed in M-OVPN, based on the pre-routed optimal trees, *MC-Removal* is able to place them such that the overall blocking performance is optimal. Since the *MC-Removal* heuristic uses the blocking performance as a function to gauge

Chapter 5

the MC-Criticality weight of each node, optimal placement of MC nodes is always possible regardless of the traffic load applied although the minimum number of MC nodes required for the lowest blocking probability possible may vary. However, what we are most concerned here is the optimal placement for a given number of MC nodes and not so much on finding the minimum number of MC nodes required for the optimal blocking which is a much more complex issue that is directly dependent on the routing tree used which is in turn dependent on the network topology. We have mentioned earlier that network cost is of grave concern and hence the routing trees constructed are inclined towards minimizing the network cost. Hence, with the trees already constructed, the optimal placement of MC nodes will lead to minimizing the number of MC nodes required for optimal blocking performance.

Figure 5-7 shows the processing time of the “Removal” and “Add” heuristics to place 6 MC-nodes optimally for different traffic loads. The results clearly show that the “Removal” heuristic takes much lesser time and has an advantage over the “Add” heuristic in terms of computational complexity. This difference is even more acute for heavier traffic load since more blocking probabilities need to be evaluated per iteration for the “Add” heuristic. On the other hand, the “Removal” heuristic only need to compute the criticality weight once, hence, there is only a slight increase in time processing when the traffic load increases. This holds for all the “Removal” heuristics mentioned in section 5.4.

Chapter 5

Figure 5-7: Complexity Comparison between “Removal” and “Add” heuristic.

Table 5-5 shows the WCC-Criticality weight computed for each node in the backbone. The nodes are also ranked from the smallest to the largest weight such that if y WCC nodes are allowed, the top $|N-y|$ WCC nodes in the list will have their wavelength conversion capabilities removed. Table 5-6 shows the placement configuration of the WCC nodes determined by *WCC-Removal* heuristic

Table 5-5: WCC-Criticality Weight of each Node in Backbone.

N_z (node)	$WCC-W_{N_z} (10^{-3})$ (weights)	R_N (Rank)
1	0	1
2	0	2
3	0.0344	7
4	0	3
5	0	4
6	0.9909	14
7	0	5
8	0	6
9	0.5642	12

Chapter 5

10	0.0677	9
11	0.0578	8
12	0.4413	11
13	0.2267	10
14	0.8277	13

Table 5-6: WCC Nodes Placement Configuration for All Possible Number of Allowed WCC Nodes in Backbone.

e_R	c	WCC nodes placement configuration {set of nodes which are made WCC}
0.05	1	{6}
0.05	2	{6,14}
0.05	3	{6,14,9}
0.05	4	{6,14,9,12}
0.05	5	{6,14,9,12,13}
0.05	6	{6,14,9,12,13,10}
0.05	7	{6,14,9,12,13,10,11}
0.05	8	{6,14,9,12,13,10,11,3}
0.05	9	{6,14,9,12,13,10,11,3,8}
0.05	10	{6,14,9,12,13,10,11,3,8,7}
0.05	11	{6,14,9,12,13,10,11,3,8,7,5}
0.05	12	{6,14,9,12,13,10,11,3,8,7,5,4}
0.05	13	{6,14,9,12,13,10,11,3,8,7,5,4,2}
0.05	14	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}

Figure 5-8 shows the blocking performance against the number of WCC nodes used for the *WCC-Removal* heuristic and optimal configuration (all nodes MC and WCC). It should be noted that all nodes in the backbone network in this case are assumed to be MC. It is observed that only 8 out of the total 14 nodes are required to be WCC in order to obtain minimum blocking. The exponential increase in blocking with the increase in the number of WCC nodes used reflects that the placement is optimal. This is so because should the placement not be optimal, the blocking result will be a randomized value when the number of WCC nodes used increases as reflected in the *Multicast-Random* algorithm proposed in [Ali02].

Chapter 5

Figure 5-8: WCC-Removal Heuristic for Optimal WCC nodes Placement (Using $e_R = 0.05$).

The optimal placement configuration in table 5.4 and table 5.6 is for traffic load of 0.05 Erlangs. The placement may differ when the traffic load changes. However, experimental results presented in table 5-7 reveal that there is very little or no change in the placement even if the traffic load increases. This is mainly due to the fact we place MC nodes or WCC nodes according to the impact each node has on the blocking performance when it is MC or WCC or both. Also, we construct routing trees before placement. Hence, the network topology rather than traffic load will have more influence in locating the optimal placement.

Chapter 5

Table 5-7: MC and WCC Nodes Placement Configuration with varying Traffic Loads. (Node with the largest weight will be placed with MC or WCC node first)

	MC Nodes Placement	WCC Nodes Placement
e_R	Largest $\rightarrow$ Smallest weight	Largest $\rightarrow$ Smallest weight
0.05	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.1	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.15	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.2	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.25	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.3	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.35	{6,14,12,9,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.4	{6,14,9,12,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.45	{6,14,9,12,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}
0.5	{6,14,9,12,13,11,10,8,7,5,4,3,2,1}	{6,14,9,12,13,10,11,3,8,7,5,4,2,1}

Figure 5-9 shows the blocking performance of MC-Removal heuristic for different MC/WCC node configurations and for different traffic loadings. It is shown in figure 5-6 that the number of MC nodes required while assuming all nodes are WCC to achieve all MC nodes performance is 6. We refer to such number as $k_{\min}$ (i.e. $k_{\min} = 6$). From Figure 5-9, it is interesting to note that when there are no MC nodes (i.e. $k=0$) and all are WCC nodes, the blocking curve is slightly lower than that when there are 6 MC nodes (i.e. $k=6$) and no WCC nodes for loads ranging from 0.05 to 0.5 Erlangs. Hence, we can deduce that for the optimal trees derived from the NSFNET, strictly sparse splitting with optimal placement of $k_{\min}$ number of MC nodes using MC-Removal heuristic and strictly wavelength conversion configuration with no MC nodes give almost similar performance. When there are more MC nodes, blocking will decrease since more receiver nodes can be reached via multicast trees instead of direct connection paths. Therefore, the upper bound curve is when $k=0$ where there is no MC-nodes, and lower bound when $k=6$, which is the minimum number of MC nodes to achieve the minimum blocking.

Chapter 5

Figure 5-9: MC-Removal Heuristic with varying Load.

Figure 5-10 shows the average blocking probability obtained for different configurations of *WCC-Removal* heuristic and for different traffic loads. It is shown in figure 5-8 that the number of WCC nodes required while assuming all nodes are MC to achieve all WCC nodes performance is 8. We refer to the number as $c_{\min}$ (i.e. $c_{\min} = 8$). The result is consistent with the one in figure 5-9 whereby the curve when $k=0$ and $c=8$ is slightly lower than when $k=6$ and $c=0$. The results obtained are also consistent with the traffic demand. As the traffic load increases, the blocking for each configuration increases exponentially as resources are used up faster and more receiver nodes cannot be reached just like in figure 5-9. This shows that our analytical model gives a reasonable approximation of the blocking performance computed. Likewise, when there are more WCC nodes, the number of paths to the receiver nodes which have to satisfy the wavelength continuity constraint will be reduced and hence lower

Chapter 5

blocking will be obtained. The lower and upper bound curve in blocking performance for *WCC-Removal* will be when there are minimum number of WCC nodes required (i.e. $c=8$) and when there are no WCC nodes (i.e. $c=0$) respectively.

Figure 5-10: WCC-Removal Heuristic with varying Load.

So far, we present the blocking performances of sparse splitting (fraction of network nodes are MC) and full wavelength conversion (all network nodes are WCC) configuration, and sparse-wavelength conversion (fraction of network nodes are MC) and full splitting (all network nodes are WCC) configuration. It is interesting to know how the performance will differ if we have both sparse splitting and sparse-wavelength conversion configuration. Also, if we have such a configuration, will the order of placement (MC nodes then WCC nodes or vice versa) affect the performance? In order to answer these questions, we perform simulations on sparse-splitting and sparse-wavelength conversion

Chapter 5

configuration on the two modes mentioned in section 5.4. One is *WMC-Removal* heuristic where *WC-Removal* is carried out first before *MC-Removal*. The other is *MWC-Removal* heuristic where *MC-Removal* is carried out first before *WC-Removal*.

Figure 5-11: Blocking Performance of WMC-Removal Heuristic vs. k for different c (using $e_R = 0.05$).

Figure 5-11 shows how the performance of *WMC-Removal* heuristic differs with varying k . Figure 5-12 shows how the performance of *MWC-Removal* heuristic differs with varying c . These two heuristics are deemed to be belonging to the *hybrid mode* since they make use of a combined removal strategy. From Figure 5-11, the results show that the minimum number of MC nodes required to obtain the minimum blocking possible remains as 6 for $c \geq 1$ and 5 for $c=0$ regardless of the number of WCC nodes used. This is expected as the increase or decrease in WCC nodes will only affect the number of receiver

Chapter 5

nodes that have to observe the wavelength continuity constraint, and is independent from whether the receiver is reached via multicast trees, direct connection paths or blocked. It is interesting to observe that in between the curves in figure 5-11, the difference gets smaller with the increase in the number of WCC nodes introduced. This is so because as more WCC nodes are introduced, the blocking is due more to the lack of resources in the links, whereas for cases when lesser WCC nodes are introduced, blocking is due more to the wavelength continuity constraint.

Figure 5-12: Blocking Performance of MWC-Removal Heuristic vs. c for different k (using $e_R = 0.05$).

Figure 5-12 shows that the minimum number of WCC nodes required for minimum blocking possible remains at 8 for $k \geq 1$. For the case where there are no MC nodes (i.e. $k=0$), the results show that only 5 WCC nodes are needed. This is because all the receiver nodes are reached via direct connection paths

Chapter 5

when there are no MC nodes. Hence when all the wavelengths per fiber of each link are occupied eventually, it does not matter if wavelength conversion is possible. Again, it is observed that the difference in between curves is smaller when more MC nodes are introduced into the network.

We can deduce from figure 5-11 and figure 5-12 that the minimum number of MC and WCC nodes to be placed for minimum blocking possible will not change much regardless of if we perform *MC-Removal* first or *WCC-Removal* first. Should we be given a set of $\psi = \{k, c\}$, we can use either *MWC-Removal* or *WMC-Removal* to obtain the same optimal placement. One practical use of this conclusion is when we only know the optimal location of WCC nodes in existing infrastructure and are required to find the optimal placement of MC nodes and vice versa.

In table 5-8, the criticality weight of each node is computed when both MC and WCC capabilities are removed from the node and we refer to it as dual-criticality weight. Here we define dual-nodes as a node with both MC and WCC capabilities. According to the ranking in table 5-8, the blocking performance for *Dual-Removal* heuristic is computed for different numbers of dual-nodes used. This heuristic is referred to as *dual mode* since it involves the concurrent removal strategy. The *Dual-Removal* heuristic is then compared with *MWC* and *WMC-Removal* and the results are shown in figure 5-13.

Table 5-8: Dual-Criticality Weight of Each Node in Backbone.

N_Z (node)	$Du-W_{N_Z} (10^{-3})$ (weights)	R_N (Rank)
1	0	1

Chapter 5

2	0	2
3	0.0034	7
4	0	3
5	0	4
6	0.3663	14
7	0	5
8	0	6
9	0.2260	13
10	0.0068	8
11	0.0201	9
12	0.1203	11
13	0.0680	10
14	0.1929	12

The minimum number of dual-nodes required to achieve the lowest blocking probability possible for all the heuristics is 8. It can be observed that the blocking performance of the three heuristics are very close to each other, indicating that they yield almost similar performance except that when the number of dual-nodes used is very small (less than 2). Hence, in terms of dual-nodes, *Dual-Removal* is preferred since it is able to place dual-nodes optimally within a shorter period of time, as both *MWC* and *WMC-Removal* have a complexity of $2O(n^2)$ while *Dual-Removal* has a complexity of $O(n^2)$. However one disadvantage for *Dual-Removal* is that there is a smaller degree of flexibility in the number of WCC and MC nodes used since their numbers must be equal to each other. For example, using dual mode, we need 8 dual-nodes (8 MC and WCC nodes) and takes $O(n^2)$ computation time for minimum blocking. If using *MWC* or *WMC-Removal* heuristics, we need 6 MC nodes and 8 WCC nodes, and takes $2O(n^2)$ computation time. Thus, half the computation time at the expenses of 2 extra MC nodes for dual case. Hence, the objective for faster computation or lesser devices used is the deciding factor when determining if dual or hybrid mode of our proposed heuristics is to be used.

Chapter 5

Figure 5-13: Blocking Performance vs. number of Dual-nodes used for different heuristics (using $e_R = 0.05$)

5.6 Summary

In this chapter, we first introduce the architecture of M-OVPN, and explain the need to limit the number of MC and WCC nodes in the optical backbone of M-OVPN for minimum blocking. We then introduce an analytical model for the first time to compute approximate average receiver blocking probability for a set of incoming multicast sessions to the backbone under sparse splitting and sparse-wavelength conversion conditions. Based on the computed blocking probability, we derive the “Removal” heuristics to place MC and WCC nodes optimally and show that it is more efficient than the existing “Add” heuristic. We then combine the heuristics (*MWC and WMC-Removal*) to place MC and WCC nodes optimally and show that given a set of MC nodes already optimally placed, we are able to use our proposed heuristic to place a given set of WCC

Chapter 5

nodes optimally and vice versa. Hence, we have shown that with an efficient placement heuristic which considers both sparse-splitting and sparse-conversion factors, we can consider MC or WCC node placement sub-problems independently and then combined them together to achieve optimal placement for MC and WCC nodes. To make such a combined placement more efficient, we propose the *Dual-Removal* heuristic which has both MC and WCC capabilities removed concurrently from the nodes and refer to the remaining optimally placed nodes as dual-nodes. It is found that such a heuristic is able to achieve the same minimum blocking but at the expense of extra MC and WCC nodes. Although currently, it may be adequate to employ just O-E-O conversion within the backbone instead of going all-optical, the increase in Internet traffic each day, the trend of moving towards an all-optical backbone is inevitable in the near future, and hence it is worthwhile to perform detailed studies on it.

Chapter 6

Dynamic Multicast Routing and Wavelength Assignment with Minimal Conversions in Delay-Constrained WDM Networks

6.1 Introduction

Multicasting will be a useful feature in future wavelength-routed WDM networks for supporting bandwidth intensive applications such as high definition television (HDTV), video conferencing and video on demand (VOD) [Rou03]. However, many issues still have to be addressed in order for multicasting to operate efficiently in WDM networks. One important issue concerns multicast routing and wavelength assignment (MC-RWA). This problem has been investigated in the context of optimal blocking and optimal wavelength assignment [CTL06], both of which are essential for efficient multicast operations. Constraints such as limited wavelength conversions and light-splitting are also introduced in these studies to allow multicast operation to be supported in a cost effective manner. In addition, static MC-RWA – where incoming traffic is known before routing – is normally assumed instead of the more realistic dynamic MC-RWA. Due to the popularity of real-time and time-sensitive applications, it is only practical and prudent to consider the delays between data sent from the source and received by each destination. This

Chapter 6

delay should be within a defined threshold for the desirable quality and this threshold is known as the destination's delay constraint or delay constraint in short. Such a constraint normally exists in the routing problems for the electronics plane. Although the propagation of light is extremely fast compared to electronics signal, delays are still of utmost importance in WDM networks for ultra time-sensitive applications, especially so when applications are designed in such a way whereby data has to be received fast for premium quality services. Beside the delay constraint, the number of wavelength conversions is also important for efficient multicasting over WDM networks. This is because a large number of wavelength conversions will necessitate a large number of wavelength converters [CLC03] in the transparent optical network which are both costly and hard to fabricate, especially so in multicasting where light-splitters [Muk97] are also required for the construction of efficient trees. For a translucent optical network, the high number of wavelength conversion will lead to a high number of optic-electronic-optic (O-E-O) [RS98] conversions which are costly and time consuming. Hence, it is important to control the number of wavelength conversions during MC-RWA for high multicast efficiency and must be carried out in the dynamic fashion since a link's capacity changes frequently.

In this chapter, we propose an algorithm which makes use of a generic layered auxiliary graph model [ZPCSZ08] to obtain dynamic trees with minimal wavelength conversion as well as blocking during dynamic multicast routing and wavelength assignment under a delay-constraint environment. Unlike the study reported in [ZPCSZ08], we focus in reducing wavelength conversions

Chapter 6

and consider delay constraint for delay sensitive multicast applications. Our algorithm is able to perform both routing and wavelength assignment concurrently while the existing algorithms consider them separately. It is this salient feature that enables us to obtain minimal wavelength conversions for the dynamic MC-RWA problem. The chapter is organized in the following manner. Section 6.2 defines the problem formulation and the network model used. Section 6.3 describes our proposed algorithm. Section 6.4 presents the simulation results and the corresponding discussions. Finally, we provide a summary of the chapter in section 6.5.

6.2 Problem Formulation

6.2.1 Network Model

The network can be modeled as a graph expressed as $G(V,E)$ with a set of vertices, V and edges, E joining between them. Each edge has a non-negative delay metric, D associated with it and it is proportional to its distance. Maximum delay is set to 10 units. The maximum delay is set in such a way that blockings will occur if the delay-constraint specified is low. This is to simulate the practical condition when strict delay-constraint will cause some traffic to be blocked due to the delay-constraint condition not satisfied. Each edge also has two optical fibers reserved, one for each direction. Each optical fiber has a finite capacity of W wavelength channels. Each node is assumed to have splitting and wavelength conversion capability unless specified. For sparse splitting and sparse wavelength conversion scenarios where only a fraction of the network nodes are multicast-capable (MC) or wavelength conversion capable (WCC), the MC and WCC nodes are randomly placed. Multicast session requests arrive

Chapter 6

in a random fashion and each session will have a constructed tree to deliver the traffic. If any receivers cannot receive traffic due to lack of available wavelength channels or inability to meet the delay constraint, the session request is deemed as blocked. We refer to the probability for such blockings as session blocking probability (SBP).

Since we assume that all nodes have splitting capability and the objective is to minimize wavelength conversions, a wavelength conversion is required for each unique pair of incoming and outgoing wavelength assignment at a particular node. Such a conversion can be either O-E-O or purely optical [YRL99]. Wavelength conversion and conversion are used interchangeably in this thesis for convenience.

6.2.2 Problem Description

For each incoming multicast request, a wavelength assigned dynamic tree is constructed to deliver the request to its destinations. The objective of this chapter is to construct such a tree and assign wavelengths to its edges such that the average SBP and average number of conversions are minimal. In addition, the average delay between source and destinations for each tree should be minimal so that the request will not be blocked even when the delay constraint is strict (low).

6.3 Proposed Algorithm

6.3.1 Generic Layered Auxiliary Graph Model

Figure 6-1: An example of GLAGM. Note that the above GLAGM shows that wavelength channel 2 is not available on link CB, DA, BC and BA.

The generic layered auxiliary graph model (GLAGM) is adopted from [ZPCSZ08]. Each layer represents a wavelength plane whereby all the edges between two nodes have that particular wavelength channel available for transmission. Since the availability of each wavelength channel changes after each incoming multicast session, GLAGM is updated to construct the delivery

Chapter 6

tree for the next multicast session. In addition, within the same layer, no more than two outgoing links are allowed from a node which is not multicast capable. In between layers, a wavelength conversion edge connects between a node x in layer A and a nodes y in layer B if 1) wavelength channel B is available for transmission in the edge between nodes x and y , and 2) node x is WCC. Since a wavelength conversion edge connects two different layers, a wavelength conversion is required and we assign one unit of wavelength conversion cost (i.e. $\phi = 1$) to each of this wavelength conversion edge, and zero unit (i.e. $\phi = 0$) for edges on the same layer. Hence, GLAGM becomes a 3D graph instead of the conventional 2D one. Figure 6-1 shows an example of the GLAGM.

6.3.2 Proposed Algorithm: Multi-Layers Tree (ML)

We name our proposed algorithm as multi-layer tree algorithm (ML) since it makes use of GLAGM which consists of multiple layers, each representing a wavelength plane. ML consists of three stages. Stage 1 constructs a minimum weight tree from source to all receivers using a modified Prim's algorithm on GLAGM. Each edge in GLAGM will have its weight computed. This weight takes wavelength conversions and delay residue into consideration and will be presented later in this section. We define delay residue of an edge as the difference in delay-constraint and the delay summation of the edge and the tree path which the edge is appended to. Stage 2 prunes unnecessary edges from the tree constructed in stage 1. Stage 3 performs "trace back" to ensure that the constructed tree is of minimum weight.

Chapter 6

6.3.2.1 Stage 1: Minimum Weight Tree Construction

As the objective is to obtain trees with minimal wavelength conversions and multicast blocking, we have to consider both delay residue and conversion in the link weight. This concept of considering any two metrics is first introduced in [KPP93]. The link weight is expressed as:-

$$LW(l_{ab}) = (1 + \phi_{ab}) / (\Delta - D(l_{ab})), \quad (6.1)$$

where ϕ_{ab} is the wavelength conversion cost on link ab and Δ is the delay constraint for all the receivers in the multicast session. The summation of 1 to the wavelength conversion cost in (6.1) is to ensure that the link weight is never zero. This is to make sure that in the event of zero conversion, the deciding factor lies strictly on the link's delay residue. The link weight itself is not useful during the construction of the tree since the delay residue considered is only with respect to the single link and not with the tree constructed so far. Hence, during the construction of the tree using Prim's algorithm, a selection function (SF) is required to select the next nearest (in terms of hop counts) node to be appended to the tree and this continues until all the receivers are in the tree. The reason for using Prim's algorithm is to construct the tree quickly since it is a distributed process instead of a centralized one. We consider an algorithm a distributed process when only the next "best" hop or link is appended to the tree during a tree's construction, whereas a centralized process will compute all the possible paths and append the "best" path to the tree during construction. Hence, it is clear that a centralized process takes much longer time and more processing than the distributed process. In order to consider both the delay

Chapter 6

residue and wavelength conversion cost with respect to the tree constructed so far, the selection function is expressed as:-

$$SF(l_{ab}) = \left(1 + \sum_{i \in T^k} \phi_i + \phi_{ab}\right) / \left(\Delta - D_{T^k}(s, a) - D(l_{ab})\right), \quad (6.2)$$

where $D_{T^k}(s, a)$ is the delay from source s to node a via the tree constructed so far. From (6.2), it is easy to see that the link with the minimum SF value will be selected to append the next nearest node during tree construction in order to obtain a tree with minimal wavelength conversions as well as blocking due to delay constraint. Since the term $\sum_{i \in T^k} \phi_i$ in (6.2) is a constant and the purpose is to identify the link with the minimum SF value, the selection function can be expressed as:-

$$SF(l_{ab}) = (1 + \phi_{ab}) / \left(\Delta - D_{T^k}(s, a) - D(l_{ab})\right). \quad (6.3)$$

From (6.1) and (6.3), we can easily express SF in terms of LW using the following expression:-

$$SF(l_{ab}) = (1 + \phi_{ab}) / \left(\left((1 + \phi_{ab}) / LW(l_{ab})\right) - D_{T^k}(s, a)\right). \quad (6.4)$$

6.3.2.2 Stage 2: Pruning

From stage 1, since tree construction is carried out using Prim's algorithm in a distributed manner, some of the edges are redundant since the leaf node may not be the receiver. Hence, such edges are pruned off to retain the more efficient delivery tree. The ingress of the pruned path will be either a receiver node or a node with an out-degree greater than 1 to ensure no disjoint sub-trees will be created due to pruning.

Chapter 6

6.3.2.3 Stage 3: Trace Back

For a distributed algorithm where the next nearest node is appended during tree construction, the tree may not be the minimum weight tree. This is because the next nearest node appended to the tree may not necessarily be included in the tree as it is not a receiver node. Hence, there may be some redundant non-leaf nodes in the constructed tree. A redundant node here refers to a node which can be removed and replaced for a smaller weight tree. To solve this, the “trace back” method is used to eliminate such redundant nodes. This method was first introduced in [YPC08] and is adopted here by making use of the GLAGM instead of the conventional graph. The concept of “trace back” is very simple. For each node in the tree, we find a direct link between the node and one of its ancestor nodes. If such a link exists, the SF of the tree path between an “anchor” node and the node is computed and compared with the SF of this direct link. If the SF of the direct link is smaller, it indicates that there exists redundant nodes in the original tree path and hence the link will replace the path to remove any redundant nodes. The ingress of the replaced path must always be an “anchor” node. An anchor node refers to a node which creates disjoint sub-trees if it is removed from the tree.

Figure 6-2 shows an example of performing “trace back” on the constructed tree. A member node for a multicast session is also referred to as the receiver of the data sent from the source for the session. From figure 6-2(a), since $SF(l_{BC}) < SF(l_{BD})$ and $SF(l_{CD}) < SF(l_{BD})$, node C is included according to the tree construction algorithm where the next nearest node of the existing tree is considered. This node C may or may not be a redundant node. Thus, “trace

Chapter 6

back” is performed as shown in figure 6-2(b). Considering node D , since there is a direct link from ancestor node B to D with SF smaller than that of the tree path from B to D via non-receiver node C , the tree path is replaced by the direct link and the final tree is shown in figure 6-2(b). Hence, it is observed that the tree has a smaller SF value of 2. This is repeated for all nodes in the tree until no redundant nodes existed.

Figure 6-2: (a) Original constructed tree. (b) Tree after performing “trace back”.

As mentioned earlier, the main reason for adopting a distributed approach instead of a centralized one is because of the much shorter tree construction duration. A centralized approach requires the minimum weight path between any two nodes for all nodes to be computed whenever the distribution of available channels in the links changes. This is very time consuming and requires a lot of overheads although a more optimal result can be obtained.

Chapter 6

However, it was shown in [YPC08] that the “trace back” method eliminates the “ill effect” of the distributed approach of having redundant nodes and, hence, is able to obtain near optimal result. Note that although the “trace back” procedure may appear to be a centralized procedure where the best replacement link will be found to improve the overall cost, there is actually no computation of extra paths needed since only the SF of the direct link and the corresponding existing tree path between two nodes are required to be computed and compared. Hence, the “trace back” procedure cannot be considered a centralized procedure and our proposed algorithm is in fact a distributed algorithm.

6.3.3 Contribution

Our proposed algorithm is able to map the wavelength conversion planes into routing plane using GLAGM so that wavelength assignment and tree routing can be carried out together. This allows optimal configuration of wavelength converters and light-splitters for transparent optical networks (or O-E-O converters and multicast-capable devices for translucent optical network) in a single node. Existing algorithms are unable to do so as they consider tree routing and wavelength assignment separately. Figure 6-3 shows an example of how GLAGM is able to carry out the wavelength assignment and tree routing concurrently in a transparent optical network.

Chapter 6

Figure 6-3: (a) Routing tree using existing algorithms. (b) One possible configuration of splitters and converters in node C. (c) Another possible configuration of splitters and converters in node C. (d) Routing tree using our proposed algorithm.

Chapter 6

From figure 6-3, the possible configurations of splitters and wavelength converters within node C are shown in figure 6-3(b) and figure 6-3(c). The configuration shown in Figure 6-3(b) requires 3 conversions whereas the configuration shown in figure 6-3(c) requires only 2 at the expense of one more splitter. There is no way for the existing algorithms to identify the configuration since only the routing plane is visible and wavelength assignment is carried out after the tree is constructed, as shown in figure 6-3(a). On the contrary, our proposed algorithm is able to identify and hence select the best configuration shown in figure 6-3(d) since both the routing and wavelength conversion planes are visible using GLAGM.

6.4 Simulation Results and Discussion

6.4.1 Random Network Generation

We generate a random network of 30 nodes using Waxman method [Wax88]. The link delay is set to be proportional to its distance and lies in the range (0, 10]. Multicast request arrivals follow a Poisson distribution with a mean arrival rate of λ . For each admitted multicast session request, the holding time follows an exponential distribution with a mean holding time of μ . Hence, traffic load is expressed as $L = \lambda / \mu$. The source and receivers of each request are randomly selected and the receiver size for all the multicast requests is set uniformly at 50% of the network size, unless specified.

6.4.2 Performance Metrics

6.4.2.1 Session Blocking Probability (SBP)

SBP is defined as the blocking probability of multicast session requests. A

Chapter 6

request is considered blocked as long as one of its receivers is blocked due to insufficient wavelength channels available or unable to meet the delay constraint criteria. The tree path delay between the source and each receiver must be less than the delay constraint specified.

6.4.2.2 Session Wavelength Conversions (SWC)

SWC is defined as the number of wavelength conversions required for each session request. Without loss of generality and for the sake of simplicity, we define that a wavelength conversion is required at a node as long as there exists a unique pair of different wavelengths assigned between the incoming and outgoing link of the node.

6.4.2.3 Average Session's Receiver Delay (ASRD)

ASRD is the average delay among all the receivers of a multicast session request. Each receiver delay is the tree path delay from the source to the receiver.

6.4.3 Comparison Algorithms

In this section, we introduce four algorithms to compare with our proposed algorithm. The first three algorithms obtain the best result for each performance metric mentioned in section 6.4.2. They are used to illustrate how well our proposed algorithm fare for each metric. [SRV97b] indicates that Prim's algorithm can be first used to obtain minimum cost spanning tree before a refinement technique is carried out to obtain a near optimal minimum cost tree for each multicast session. Recall that a minimum cost spanning tree spans to

Chapter 6

all nodes in the network whereas minimum cost tree for each session only spans to the intended recipients of the session from the source. Therefore, it is sufficient to just compare our proposed algorithm with these three near optimal algorithms to estimate the performance differences between our proposed algorithm and the optimal results. For each of the first three comparison algorithms introduced here, Prim's algorithm is first used to compute a minimum weight spanning tree before pruning. As pruning only removes redundant leftover nodes from the tree and the resultant tree is not necessarily cost-optimized for each session, a refining technique such as "trace back" is used to obtain a near minimum weight tree for each multicast session after the pruning process. The weight here depends on which metric is to be minimized for the algorithm. Two common wavelength assignment methods, namely, first-fit (FF) and random (Rand) are used in the first three comparison algorithms. For FF, the smallest available wavelength number is selected, and for Rand, any available wavelength is selected. The fourth comparison algorithm is a delay constrained multicast tree routing algorithm proposed in [CTL06]. We shall refer the algorithm from [CTL06] as *chen et al.*. For sparse splitting and sparse wavelength conversion scenarios, all the algorithms being compared have to satisfy the splitting and wavelength conversion constraints.

6.4.3.1 Minimum Delay Tree (MDT)

In MDT, the objective is to obtain minimum delay so that there is a higher probability for a receiver to receive data within the specified delay-constraint. Hence, the smallest delay link to the next nearest node is appended to the tree

Chapter 6

for each step of the tree construction. “Trace back” is performed to obtain near optimal minimum delay tree by considering link delay as the link weight.

6.4.3.2 Maximum Capacity Residue Tree (MCRT)

In MCRT, the objective is to obtain maximum capacity residue for reducing the blocking probability due to unavailability of free wavelength channels in the tree links. Hence, the largest available wavelength channels link to the next nearest node is appended to the tree for each step of the tree construction. “Trace back” is performed to obtain a near optimal maximum capacity residue tree by considering capacity residue as the link weight.

6.4.3.3 Minimum Wavelength Conversion Tree (MWCT)

In MWCT, the objective is to obtain minimum wavelength conversions for reducing the number of wavelength converters required in a transparent optical network; or for reducing the number of O-E-O conversions in a translucent optical network, and these in turn help to reduce cost and improve transmission efficiency. Hence, the link to the next nearest node which requires no wavelength conversion will have the priority to be appended to the tree for each step of the tree construction. “Trace back” is not necessary here because no redundant nodes in terms of wavelength conversions is produced. This is so because priority is already given to connections that do not require any conversion, therefore the constructed tree is already with minimum wavelength conversions.

6.4.3.4 Algorithm from Chen et al.

The authors in [CTL06] proposed trees to be constructed based on generation, refinement and conversion. Trees are generated by means of Prim's algorithm on a graph of minimum delay paths (MDPs). Minimum cost paths (MCPs) algorithm is used to refine the generated tree. Finally, each tree is converted such that the number of light-trees is minimized. It considers delay constrained dynamic tree routing with limited light splitting in large capacity networks. Lightpaths are used to route traffic to destinations which cannot be reached due to splitting limitation.. In addition, in order to ensure low SBP, the link cost metric used for this algorithm is the link load. Thus, minimizing cost here has the effect of avoiding congested links and hence minimizing SBP. In wavelength assignment phase, assignments are carried out such that the total number of lightpaths is minimized. To do this, each path in the tree searches for a common smallest wavelength. If the wavelength exists, it will be assigned for all the links in the path. If not, the first-fit assignment strategy will be adopted for each of the link in the path. Due to the dynamic nature of the environment considered, MDP and MCP between any two nodes are constructed in such a way that links with no available wavelength are not considered. Hence, the construction of MDPs and MCPs occur as long as one link has its available wavelength distribution changes. Note that this algorithm from *chen et al.* adopts the centralized approach where information of MDPs and MCPs are to be constructed and maintained in nodes before being used to construct trees. Hence, naturally, the blocking performance obtained should be better than the distributed approach. However, this is achieved at the expense of a much larger overhead and much longer tree construction time.

6.4.4 Simulation Results

6.4.4.1 For scenarios where all nodes are MC and WCC

For figure 6-4 to figure 6-6, we fix $W=3$, $\Delta=20$ and vary L . For figure 6-7 to figure 6-9, we fix $L=20$ Er, $\Delta=20$ and vary W . For figure 6-10 to figure 6-12, we set $L=20$ Er, $W=3$ and vary Δ .

Figure 6-4: Average SBP vs. Traffic Load ($W=3$, $\Delta=20$)

It can be seen from figure 6-4 that the average SBP for all the algorithms increases with an increase in L because with an increase in L and no change in capacity, more links will be congested and this lead to a higher blocking due to insufficient wavelength channels. *Chen et al.* has the minimum average SBP because it adopts the centralized approach with the objective of avoiding congested links. MDT has the minimum average SBP among the distributed algorithms as it minimizes blocking due to delay-constraint. The fact that

Chapter 6

MCRT has a higher average SBP shows that blocking is mainly due to the delay constraint. This is because MCRT maximizes the capacity residue and produces least blocking due to insufficient wavelength channels available. The average SBP of our proposed algorithm ML is very close to that of *Chen et al.* and MDT. This is because we consider the delay residue component in the selection function of our algorithm.

Figure 6-5: Average SWC vs. Traffic Load ($W=3, \Delta=20$)

It can be seen from figure 6-5 that the average SWC for all the algorithms increases slightly with an increase in L because with an increase in L , more wavelength channels are occupied and hence require more conversions in some links. MWCT has the minimum average SWC because it minimizes the wavelength conversions. The average SWC of our proposed algorithm ML is very close to MWCT and much smaller than that of *Chen et al.*. This is because

Chapter 6

we consider the wavelength conversion cost component in the selection function of our algorithm. From figure 6-6, the average ASRD for all the algorithms remain more or less the same with an increase in L . This is because there is no change in the network topology and hence no drastic change in the delay of tree path from the source to each receiver. MDT should have the minimum average ASRD because it minimizes the average delay between the source and each node in the tree. However, figure 6-6 shows that our proposed algorithm ML produces an even smaller average ASRD than MDT and much smaller than that of *Chen et al.*. This indicates that the selection function in ML is effective in keeping the delay between the source and receiver minimal.

Figure 6-6: Average ASRD vs. Traffic Load ($W = 3, \Delta = 20$)

From figure 6-7, the average SBP for all the algorithms decreases with an increase in W . This is because with an increase in W and no change in traffic

Chapter 6

load, fewer links will be congested and this leads to lower blocking due to insufficient number of available wavelength channels. *Chen et al.* and MDT perform the best in terms of minimum average SBP because there is no change in the delay constraint condition. The average SBP of ML is very close to that of *Chen et al.* and MDT. This again shows that the selection function in our proposed algorithm is effective.

Figure 6-7: Average SBP vs. W ($L = 20E_r$, $\Delta = 20$)

From figure 6-8, the average SWC for all the algorithms generally increase with an increase in W because with an increase in W , there are more options for wavelength conversions. ML again has a much smaller average SWC than *Chen et al.*. There is a slight decrease in the average SWC for MCRT-FF, ML and MWCT. This indicates a smaller chance for a particular wavelength channel to be occupied and hence less chance of requiring conversions. This is possible due to the effective wavelength assignment strategy introduced in

Chapter 6

MCRT-FF, ML and MWCT.

Figure 6-8: Average SWC vs. W ($L = 20E_r$, $\Delta = 20$)

Figure 6-9: Average ASRD vs. W ($L = 20E_r$, $\Delta = 20$)

Chapter 6

It can be seen from figure 6-9 that the average ASRD for all the algorithms generally decrease with an increase in W . This is because with an increase in W , there are more options for wavelength conversions which in turn lead to higher chance of having smaller delay links in the tree path between the source and each receiver. *Chen et al.* has the highest average ASRD because each tree is refined to be the minimum cost (link load) tree. ML has the smallest average ASRD which indicates that it is highly robust even in the event of strict delay constraint.

Figure 6-10: Average SBP vs. Δ ($L = 20 E_r$, $W = 3$)

From figure 6-10, the average SBP for all the algorithms decrease with an increase in Δ . This is because there is lesser blocking due to the delay constraint. Again ML is very close to MDT which has the minimum average SBP among all the distributed algorithms. *Chen et al.* has the minimum average SBP for the reasons given to explain the results in figure 6-4.

Chapter 6

From figure 6-11, the average SWC for all algorithms except MWCT and ML generally increase with an increase in Δ . This is because with a higher delay-constraint, lesser blocking occurs due to delay constraint criteria not met and hence allows more conversions to take place. For MWCT and ML, the average SWC decreases with an increase in Δ because of the effective wavelength assignment strategy. MWCT minimizes wavelength conversions, hence it will assign wavelengths in such a manner that the overall conversions is minimal. ML considers both routing and wavelength assignment concurrently, hence it too can make use of the more relaxed delay constraint to route and assign wavelengths in such a way that the overall conversions is minimal. ML has a smaller average SWC than *Chen et al.*, indicating a much smaller overall conversion cost.

Figure 6-11: Average SWC vs. Δ ($L = 20E_r$, $W = 3$)

Chapter 6

From figure 6-12, the average ASRD for all algorithms generally increase with an increase in Δ because with a more relax delay constraint; the tree path delay between the source and receiver can be relaxed without violating the delay constraint criteria. Again ML produces the smallest ASRD. Again, this shows the selection function used in ML is highly effective in maximizing delay residue such that the overall delay between the source and receiver is low.

Figure 6-12: Average ASRD vs. Δ ($L = 20E_r$, $W = 3$)

6.4.4.2 For scenarios where only a fraction of the nodes are MC and WCC

In this section, we compare the performance of ML with the other four algorithms for sparse splitting and sparse wavelength conversions scenarios. We fix $W = 3$, $\Delta = 25$, $L = 20$ and varies the number of MC nodes (X_{mc}) and WCC nodes (X_{wcc}) in the network. Since it has been shown earlier that the first-fit and random wavelength assignment model achieves almost similar results, we present in this section only the results of MWCT, MDT, MCRT and Chen et

Chapter 6

al. using the first-fit wavelength assignment strategy. Figure 6-13 shows that the average SBP of ML is close to that of *Chen et al.* and much lower than the other 3 algorithms for the different number of MC and WCC nodes employed.

Figure 6-13: Average SBP vs. MC and WCC nodes in Network ($L = 20Er$, $\Delta = 25$, $W = 3$)

Figure 6-14 shows that the average SWC of ML is very close to the minimum average SWC achieved by MWCT-FF. Recall that MWCT-FF minimizes wavelength conversions and hence expected to have the least average wavelength conversions compared to the rest. Figure 6-15 shows that the average ASRD of ML is even lower than that of MDT-FF, which supposes to achieve the least average ASRD since it minimizes the average delay between the source and receivers. Hence, from figure 6-13 to figure 6-15, it can be concluded that the good performance of ML over the other algorithms being studied is consistent even for scenarios where only a fraction of the network

Chapter 6

nodes are MC and WCC.

Figure 6-14: Average SWC vs. MC and WCC nodes in Network ($L = 20Er$, $\Delta = 25$, $W = 3$)

Figure 6-15: Average ASRD vs. MC and WCC nodes in Network ($L = 20Er$, $\Delta = 25$, $W = 3$)

6.5 Summary

In this chapter, we propose an algorithm which makes use of a generic layered auxiliary graph model to perform both wavelength assignment and tree routing concurrently for the dynamic MC-RWA problem to construct delay constrained minimum hop dynamic trees with minimal session wavelength conversions and session blocking probability. We explain the rationale of our algorithm and why it is able to perform wavelength assignment and tree routing concurrently while the existing algorithms are unable to do so. Through intensive simulations on randomly generated networks and comparison with the existing algorithms for scenarios where all nodes are MC and WCC, we show that our proposed algorithm ML, with the use of a highly effective selection function, is able to construct trees with low average session blocking probability, low average session wavelength conversions and low average delay between the source and each receiver for all requests. This is unlike the rest of the algorithms where one of the metrics is optimized at the expense of the other metrics. This is because ML appends the link which gives the least wavelength conversion cost and largest delay residue to the tree during construction. Lower wavelength conversion cost will increase the probability that a particular wavelength is available and hence reduces the overall blocking due to wavelength continuity constraint. Larger delay residue will increase the number of nodes to be appended to the tree without violating the delay-constraint and hence reduces the overall blocking as well. To make our study more complete, we also perform simulations for scenarios where only a fraction of the network nodes are MC and WCC. The results show that the good performance of ML over the other studied algorithms is consistent with the other results shown.

Chapter 7

Conclusion and Future Work

In this thesis, we focus on the study of performance efficient trees for multicast communications. We presented a list of important performance metrics for multicast communications. We then developed and analyzed heuristic algorithms for constructing performance efficient trees for multicast communications in both IP and optical networks. We conclude this thesis in this chapter by summarizing the major contributions and key findings. In addition, we suggest some future works motivated by the findings of this thesis.

7.1 Conclusion

In chapter 3, we study the construction of minimum cost trees in multi-group multicast. It is well known that the construction of cost optimal trees is NP-complete. Hence efficient heuristics are required to construct sub-optimal trees with a much shorter computation time. Existing heuristics are either based on multiple single-tree construction model (MSi-T) where each session's tree is required to be constructed from scratch, or based on multiple shared-tree construction model (MSh-T) where each session's trees from the same multicast group are constructed from the shared-tree of that multicast group. MSi-T requires each session's tree to be constructed from scratch and hence is able to construct trees with cost closest to the optimal but at the expenses of much more computation time and complexity which is not desirable. MSh-T

Chapter 7

requires each multicast group to construct a shared-tree after which all sessions originated from the same multicast group will have their trees constructed via pruning the shared-tree of the multicast group. Hence, MSh-T is able to construct trees faster than MSi-T but at the expense of higher cost which is not desirable.

In order to combine the advantages and eliminate the disadvantages of MSi-T and MSh-T, we first introduce the common shared-tree construction model (CSh-T) and propose two heuristics – MGST-T and DC-MGST-T – based on CSh-T to construct non-delay constrained and delay constrained trees with cost not too far off from the optimal without increasing the complexity and computation time of the tree construction process. We introduce the “trace back” feature to ensure that the “ill effect” of redundant nodes for employing shared-tree during construction is eliminated to reduce the overall tree cost. Simulation results show that MGST-T and DC-MGST-T are able to construct trees with cost not too far off from the approximate optimal with a large reduction in tree construction complexity and computation time. The results demonstrate the effectiveness of the proposed CSh-T model and the “trace back” feature implemented in the proposed heuristics in constructing cost-efficient trees.

In chapter 4, we study the construction of performance efficient trees for delay constrained dynamic membership multicast. Dynamic membership multicast allows members of each multicast group to join or leave the group. Hence, the receivers of each multicast session can join or leave the session. For cost-

Chapter 7

optimal tree of the session, the tree is required to be constructed for the updated list of receivers. However, the remaining receivers will experience disruption due to the re-construction or rearrangement of the tree. Thus, the construction of cost-efficient tree and minimization of disrupted receivers are two contradicting goals of the tree construction problem in dynamic membership multicast. In order to ensure that the tree cost is kept low without inflicting large number of disrupted receivers, existing heuristic algorithms were proposed to be used to trigger rearrangement for a region of the tree after a certain threshold of the region is exceeded due to the joining and leaving of receivers. Although such algorithms could help to keep the tree cost and number of disrupted receivers at a low level, the reactive nature of the tree rearrangement in the algorithms will result in large rearrangement time. This is not desirable since it can lead to race condition whereby a new request on receiver joining/leaving may arrive before a tree rearrangement is completed. This will affect efficient utilization of link resource because link resource supposed to be released may be held up due to tree rearrangement is in progress. In addition, there is no direct control in the number of disrupted receivers caused by the rearrangement. Some control is necessary to ensure that rearrangement only carried out if it does not cause too many receivers to be disrupted.

In view of the above, we propose a new proactive rearrangement heuristic, PRDCDMM, whereby disruption control and pre-construction of paths for rearrangement are implemented. In order to ensure trees can be rearranged with fewer receivers not able to satisfy the specified delay constraint, minimum

Chapter 7

hybrid cost paths are constructed during the initialization stage before the arrival of session requests. The links of these paths form the traffic-ready link graph whereby each link in the graph is uniformly assigned a number of channels to relay traffic when the link is used as part of the rearranged tree. PRDCDMM with uniform channel assignment is known as UA PRDCDMM. Since some links may not be used frequently to rearrange trees, therefore to eliminate the ill effect of assigning too many channels in some links which are not usually used, PRDCDMM (Non-UA PRDCDMM) with non-uniform channel assignment is proposed. The function of a path's hybrid cost is defined in such a way that the path's cost and delay residue are minimized and maximized respectively. PRDCDMM performs rearrangement for every new joining/leaving update request and hence is a proactive rearrangement process. During rearrangement, PRDCDMM makes use of those non-fully occupied links (i.e. links which have at least one free channel) from the pre-computed minimum hybrid cost paths to rearrange a single sub-tree which gives the maximum tree cost reduction and have an accepted number of disrupted receivers. If the number of allowable disrupted receivers is large enough, PRDCDMM is able to perform full rearrangement of the entire tree when the rearranged sub-tree is the tree itself. Despite performing rearrangement proactively, simulation results show that UA PRDCDMM is able to construct minimum cost trees with a large reduction in rearrangement time compared to the reactive algorithms. In addition, it is able to achieve total disruption control. To evaluate the performance of the algorithms in terms of both disruptions and cost, a new CC-DC performance index is introduced. UA PRDCDMM is shown to outperform the existing reactive algorithms under the new performance

Chapter 7

index. Non-UA PRDCDMM is shown to be able to achieve the equivalent good results of UA PRDCDMM with a much improved channel utilization rate and lower number of reserved links.

In chapter 5, we study the multicast-capable (MC) node and wavelength-conversion capable (WCC) node placement problems in the context of optical multicasting. The number of MC and WCC nodes to be placed in an optical network should be low due to the high cost and complexity of MC and WCC nodes. A certain number of MC and WCC nodes have to be placed optimally in the optical network so that the tree for each multicast session can be constructed to achieve a low receiver blocking probability. It has been shown in existing research that just a fraction of the network nodes need to be MC and all nodes to be WCC, or a fraction of the network nodes to be WCC and all nodes to be MC to achieve the minimum blocking probability possible. Existing heuristics such as the “add” heuristic from [Ali02] consider the placement of MC and WCC nodes independently. Although [YL03] and [CY05] place MC and WCC nodes jointly, the MILP and ILP optimal algorithms used are too computationally intensive. Hence, there are enough evidences in existing works that show that MC node and WCC node placements can be efficiently solved individually while assuming full wavelength conversion and full splitting respectively. However, there is a lack of research in considering the impact of both sparse-splitting and sparse-wavelength conversion to the blocking performance in the optical network.

Chapter 7

Thus, we first propose a new analytical model to approximate the average receiver blocking probability for a set of multicast session trees in an optical network with sparse-splitting and sparse-wavelength conversion. We then introduce the “removal” method for MC and WCC node placements in different node configurations. The different node configurations include sparse-splitting and full wavelength-conversion, full-splitting and sparse wavelength-conversion, full-splitting and full wavelength-conversion, and sparse-splitting and sparse wavelength-conversion. The proposed placement heuristics based on the “removal” method include the hybrid-mode where optimal placement of MC nodes are found first before WCC nodes and vice versa, and dual-mode where optimal placement of nodes with MC and WCC capabilities are found. Simulation results show that our proposed heuristics based on “removal” method is faster than the existing “add” method. The results also demonstrated that heuristics based on the proposed “removal” method enables just a fraction of the network nodes to be MC or WCC or both to achieve minimum blocking probability.

In chapter 6, we study the dynamic MC-RWA problem for optical multicast and seek to construct delay constrained minimum hop dynamic trees with minimal wavelength conversions and minimal session blocking probability. In the past, multicast routing and wavelength assignment in optical multicast are solved as two independent sub-problems. In addition, there is a lack of research in minimizing wavelength conversions for MC-RWA problem. Too much wavelength conversions will result in longer data transmission delays for translucent optical networks since more O-E-O conversions are required, and

Chapter 7

too much wavelength conversions will also require more wavelength converters to be deployed in transparent optical networks. Hence, minimizing wavelength conversions will minimize the data transmission delays as well as the number of expensive converters required in the network. Since multicast routing and wavelength assignment are mutually dependent on each other for optimal results in terms of blocking performance as well as wavelength assignment, we propose a new heuristic known as Multi-Layers Tree algorithm (ML) to make use of a generic layered auxiliary graph model (GLAGM) to solve multicast routing and wavelength assignment sub-problems jointly with the objective of minimizing session blocking probability, wavelength conversions and average delay between the source and receivers. The average delays between the source and receivers should be small so that only a small number of multicast sessions will be blocked due to at least one of the receivers not able to satisfy the specified delay-constraint.

Through simulations, we show that ML, with the implementation of “trace back” and with the use of a highly effective selection function to append the next node to the tree with the “best” link, is able to construct trees with a low average session blocking probability, low session wavelength conversions and low average delay between the source and each receiver for all requests. This is because ML appends the link which gives the least wavelength conversion cost and largest delay residue to the tree during construction. Lower wavelength conversion cost will increase the probability that a particular wavelength is available and hence reduces the overall blocking due to wavelength continuity constraint. Larger delay residue will increase the number of nodes to be

Chapter 7

appended to the tree without violating the delay-constraint and hence reduces the overall blocking. As ML is a distributed process whereby the next “best” hop is selected to append to the tree during construction, the tree construction time for ML is small. To show that the good performance of ML is not limited to scenario where all nodes are assumed to be MC and WCC, simulations are also performed for scenario where only a fraction of the nodes are MC and WCC. The results showed that the good performance of ML over the studied algorithms in such scenario is consistent with the scenario where all nodes are MC and WCC. This exhibits the flexibility of ML in any environment.

7.2 Future Research

One of the critical factors which allow multicast communication to be supported in an economical and efficient manner is the construction of performance efficient trees. This thesis contributed to the fast construction of performance efficient trees in both IP and optical multicast networks under non-delay and delay constrained conditions. The findings of this thesis have opened up some interesting new research directions as described below.

- In chapter 3, we proposed heuristics to construct non-delay and delay constrained trees based on a common shared-tree construction model. The heuristics will have to be modified if an optical network instead of an IP network is used. This is because the presence and absence of MC and WCC nodes in a transparent optical network have a direct impact on the construction of the common shared-tree. Also, the common shared-

Chapter 7

tree construction model will have to be modified to solve the MC-RWA problem efficiently for optical multicasting.

- In chapter 4, our proposed proactive rearrangement heuristic pre-computed minimum hybrid cost paths for future rearrangements due to the updates of receivers joining or leaving a multicast session. While the function defined for each path's hybrid cost is able to achieve excellent results for uniform delay-constraint scenario, it may not be able to achieve excellent results for non-uniform delay-constraint scenario by using the maximum specified delay-constraint among all receivers for all sessions in the computation of the hybrid cost. This is because some receivers may have a specified delay-constraint much smaller than the maximum and hence affects the computation of the hybrid cost which in turn leads to higher cost. Thus, a better hybrid cost function should be defined for the proactive rearrangement heuristic for non-uniform delay-constraint scenario.
- In chapter 5, we assume static membership multicast where receivers of each multicast session is fixed. Hence, trees spanning from the source to a fix set of receivers can be constructed efficiently from the optical network in which MC and WCC nodes are placed optimally using the proposed heuristics based on the "removal method". For dynamic membership multicast where receivers for each multicast session changes, the placement will have to be dynamic to achieve optimal

Chapter 7

results. Thus, algorithms for dynamic optimal placement of MC and WCC nodes need to be developed for dynamic membership multicast.

- In chapter 6, we studied and compared the performance of our proposed algorithms for scenario where all nodes are MC and WCC; and for scenario where only a fraction of the nodes are MC and WCC. We assume random placement for a given number of MC and WCC nodes in the network for the latter. Hence, under sparse-splitting and sparse wavelength-conversion scenario, it will be interesting to study how the generic layered auxiliary graph model can be used to find the optimal placement for the given number of MC and WCC nodes.

As the sub-problems considered in the thesis are NP-complete problems, only computationally intensive optimal algorithms can be used to obtain optimal results. However, for the real-world networks, industry will face larger and more complex networks. The optimal algorithms will not be computationally possible for the larger networks. This thesis focuses in studying performance efficient trees for multicast communications, and developed a number of heuristics to achieve near optimal results using much lesser computations. In other words, this thesis enables the industry to achieve approximate optimal solutions for the larger and more complex real-world networks which cannot be solved using optimal algorithms. This helps the industry to save time, cost and resources. In addition, this thesis can also serve as a reference and benchmark so that the proposed heuristics within it can be modified appropriately in future for use in any real-world scenarios.

Author's Publications

Journals

1. Keen-Mun Yong, Gee-Swee Poo and Tee-Hiang Cheng, "Optimal Placement of Multicast and Wavelength Converting Nodes in Multicast Optical Virtual Private Network" *Elsevier Journal on Computer Communication*, Vol. 29, Issue 12, Aug. 2006, Pages 2169-2180.
2. Keen-Mun Yong, Gee-Swee Poo and Tee-Hiang Cheng, "Proactive Rearrangement in Delay Constrained Dynamic Membership Multicast", *Elsevier Journal on Computer Communication*, Vol. 31, Issue 10, June 2008, pp. 2566-2580.

Conferences

1. Keen-Mun Yong, Gee-Swee Poo, Tee-Hiang Cheng, "Optimal Placement of Multicast and Wavelength Converting Nodes in Multicast Optical Virtual Private Network", in *Proc. IEEE LCN 2005*, pp.505-506, Nov. 2005.
2. Keen-Mun Yong, Gee-Swee Poo, Tee-Hiang Cheng, "Efficient Heuristic for Minimum Cost Trees Construction in Multi-Groups Multicast", in *Proc. IFIP-Networking 2008*, pp. 215-226, May 2008.
3. Keen-Mun Yong, Gee-Swee Poo and Tee-Hiang Cheng, "Proactive Rearrangement in Delay Constrained Multicast with Dynamic Membership Support", in *Proc. IEEE ICCCN 2008*, pp.584-589, Aug. 2008.

Chapter 7

4. Keen-Mun Yong, Tee-Hiang Cheng and Gee-Swee Poo, “Dynamic Multicast Routing and Wavelength Assignment for Delay-Constrained WDM Networks”, *submitted as invited paper to OECC 2009*, Hong Kong.
5. Keen-Mun Yong, Tee-Hiang Cheng and Gee-Swee Poo, “Dynamic Multicast Routing and Wavelength Assignment with Minimal Conversions in Delay-Constrained WDM Networks”, to be presented at *ICCCN 2009*, USA.

Bibliography

A

[AD00] M. Ali and J. Deogun, "Power-efficient design of multicast wavelength-routed networks", *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1852-1862, Oct. 2000.

[AS00] A.S. Arora and S. Subramaniam, "Converter Placement in Wavelength Routing Mesh Topologies", in *Proc. IEEE ICC'00*, vol. 3, pp. 1282-1288, Jun. 2000.

[AZ01] T. Alrabiah, T. Znati, Delay-constrained, low-cost multicast routing in multimedia networks, (Special Issue, "Routing in Computer and Communication Systems"), *Journal Parallel Distrib. Comput.*, vol.61, no. 9, pp.1307-1336, Sept. 2001.

[Ali02] M. Ali, "Optimization of Splitting Node Placement in Wavelength-Routed Optical Networks", *IEEE J. Sel. Areas Commun.*, vol. 20, no.8, pp. 1571-1579, Oct. 2002.

[AC05] A. Boudani, B. Cousin, "Multicast tree in MPLS network", in *Proc. IEEE Globecom '05*, vol.2, pp. 1017-1022, Dec. 2005.

Bibliography

B

[BFC93] A. Ballardie, P. Francis, J. Crowcroft, “Core Based Trees (CBT) and Architecture for Scalable Inter-domain Multicast Routing”, in *Proc. ACM SIGCOMM’93*, pp. 85-93, Aug. 1993.

[BV97] F. Bauer, A. Verma, “ARIES: a rearrangeable inexpensive edge-based on-line Steiner algorithm”, *IEEE J. Sel. Areas Commun.*, vol. 15, pp.382-397, Apr. 1997.

C

[CCBLS00] T. Y. Chai, T. H. Cheng, Bose S.K., C. Lu, G. Shen, “Analytical model for a WDM optical cross-connect with limited conversion capability”, *IEEE Commun. Letters*, vol. 4, no. 11, pp. 369-371, Nov. 2000.

[CLX00] G. Chen, G. Li, G. Xue, “Optimal placement of wavelength converters in WDM optical networks with a general tree of rings topology”, in *Proc. IEEE ICCCN’00*, pp. 606-611, Oct. 2000.

[CM01] A. Chakrabarti, G. Manimaran, “A case for scalable multicast tree migration”, *IEEE GLOBECOM’01*, vol. 3, pp.2026-2030, Nov. 2001.

[CLC03] X. Chu, Bo Li, I. Chlamtac, “Wavelength Converter Placement under Different RWA Algorithms in Wavelength-Routed All-Optical Networks”, *IEEE Trans. Commun.*, vol. 51, no. 4, Apr. 2003.

Bibliography

[CY05] Y. Cao, O. Yu, “Optimal placement of light splitters and wavelength converters for multicast in WDM networks”, in *Proc. IEEE ICCAS'05*, vol. 1, pp. 585-589, May 2005.

[CLT06] M.T. Chen, S.S. Tseng, and B.M.T. Lin, “Dynamic multicast routing under delay constraints in WDM networks with heterogeneous light splitting capabilities”, *Computer Commun.*, vol. 29, no. 5, pp. 1492-1503, May 2006.

D

[Dijk59] E.W. Dijkstra, “A note on two problems in connection with graphs”, *Numerische Mathematik*, vol.1, pp.267-27, 1959.

[DPW88] S. Deering, C. Partridge, D. Waitzman, “Distance Vector Multicast Routing Protocol”, IETF RFC 1075, Nov. 1988

F

[Flo62] R.W. Floyd, “Algorithm 97: Shortest paths”, *Commun. ACM*, vol. 5, no. 6, p.345, June 1962.

[FT87] M. Fredman, R. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms”, *Journal of the ACM (JACM)*, vol. 34 no. 3, pp. 596-615, July 1987.

[FZG01] A. Fei, Z. Duan, M. Gerla, “Constructing shared-tree for group multicast with QoS constraints”, in *Proc. IEEE Globecom'01*, vol. 4, pp. 2389-2394, Nov. 2001.

Bibliography

[FB05] A. Farrel, I. Bryskin, “GMPLS: Architecture and Applications”, Morgan Kaufmann, Dec. 2005.

[For06] Behrouz A. Forouzan, “Data Communications Networking”, McGraw Hill Higher Education, May 2006.

[FK07] S. Fahmy, M. Kwon, “Characterizing Overlay Multicast Networks and Their Costs”, *IEEE/ACM Trans. on Networking*, vol. 15, no. 2, pp. 373-386, Apr. 2007.

G

[Gir90] A. Girand, “Routing and Dimensioning in Circuit-Switched Networks”, Addison-Wesley: Reading, MA, 1990.

H

[HR92] F. Hwang and D. Richards, “Steiner tree problems,” *Networks*, vol. 22, pp. 55–89, no. 1, Jan. 1992.

[HZ98a] W.S. Hu and Q.J. Zeng, “Multicasting optical cross connects employing splitter-and-delivery switch,” *IEEE Photonic Technol. Letter*, vol. 10, pp. 970-972, Jul. 1998.

[HZ98b] A. Hac, K. Zhou, “Multicasting algorithm in multimedia communication network”, in *Proc. IEEE Globecom’98*, vol. 2, pp. 747-752, Nov. 1998.

Bibliography

[HLP98] S. Hong, H. Lee, B.H. Park, “An efficient multicast routing algorithm for delay sensitive applications with dynamic membership”, in *Proc. IEEE INFOCOM’98*, pp.1433-1440, Mar. 1998.

[HH00] D.J. Higham, N.J. Higham, “MATLAB Guide”, Soc for Industrial & Applied Math, Jan. 2000.

[HL05] Q.-D. Ho, M.-S. Lee, “Dynamic converter-aware wavelength assignment in WDM networks with sparse conversion capability”, in *Proc. IEEE ICBN’05*, vol. 1, pp. 300-302, Oct. 2005.

[HM06] T. Hu, H. T. Mouftah, “Dynamic Constrained Multicast Routing in WDM Networks: Blocking Probability, QoS and Traffic Engineering”, in *Proc. IEEE ISCC’06*, pp. 974-980, Jun. 2006.

[HL07] C.-Y. Hsieh, W. Liao, “All optical multicast routing in sparse-splitting optical networks”, *IEEE J. Sel. Areas Commun.*, vol. 25, no.6, pp. 51-62, Aug. 2007.

I

[IW91] M. Imase and B. Waxman, “Dynamic Steiner tree problems,” *SIAM J. Disc. Math.*, vol. 4, no. 3, pp. 369–384, Aug. 1991.

Bibliography

J

[Jia98] X. Jia, “A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks”, *IEEE/ACM Trans. on Networking*, vol. 6, no. 6, pp. 828-837, Dec. 1998.

[JDH[01] X.H. Jia, D.Z. Du, X.D. Hu, “Integrated algorithm for delay bounded multicast routing and wavelength assignment in all optical networks”, *Computer Commun.*, vol. 24, no. 14, pp. 1390–1399, Dec. 2001.

K

[Kru56] J.B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem”, in *Proc. of the American Mathematical Society*, vol. 7, no. 1, pp. 48-50, Feb. 1956.

[Kar72] R.M. Karp, “Reducibility among combinatorial problems”, R.E. Miller, J.W. Thatcher (Eds.), *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-103, 1972.

[KMB81] L. Kou, G. Markowsky, L. Berman, “A fast algorithm for Steiner Trees”, *Acta Informatica*, vol. 15 pp. 141–145, 1981.

[Kel86] F.P. Kelly, “Blocking Probabilities in large circuit switched networks”, *Advance Appl. Probability*, vol. 18, pp. 473-505, 1986.

Bibliography

[KPP93a] V. Kompella, J. C. Pasquale, and G. C. Polyzos, “Multicast routing for multimedia communication,” *IEEE/ACM Trans. Networking*, vol. 1, pp. 286–292, June 1993.

[KPP93b] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, “Two distributed algorithm for multicasting multimedia information”, in *Proc. IEEE ICCCN’93*, pp.343–349, 1993.

[KK95] J. Kadirire, G. Knight, “Comparison of dynamic multicast routing algorithms for wide-area packet switched (asynchronous transfer mode) networks”, in *Proc. IEEE Infocom’95*, vol. 1, pp.212-219, Apr. 1995.

L

[LL98] H.-C. Lin, S.-C. Lai, “VTDM – A Dynamic Multicast Routing Algorithm”, in *Proc. IEEE Infocom’98*, vol. 3, pp.1426-1432, Apr. 1998.

[LMZ04] S. Li, R. Melhem, T. Znati, “An efficient algorithm for constructing delay bounded minimum cost multicast trees”, *Journal of parallel and distributed computing*, vol. 64, no. 12, pp.1399-1413, Dec. 2004.

[LLL04] Xiongfei Li, Xiangqun Li, Xiangni Li, “A dynamic algorithm used in multicast routing”, *IEEE CACWD’04*, vol. 2, pp. 1-6, May 2004.

[LA05] Yong-Jin Lee, M. Atiquzzaman, “Least cost heuristic for the delay-constrained capacitated minimum spanning tree problem”, *Computer Commun.*, vol. 28, no. 11, pp.1371-1379, July 2005.

Bibliography

M

[ME96] E. Modiano, A. Ephremides, “Efficient Algorithms for Performing Packet Broadcasts in a Mesh Network”, *IEEE/ACM Trans. on Networking*, vol. 4, no. 4, pp. 639-648, Aug. 1996.

[Muk97] B. Mukherjee, “Optical Communication Networks”, New York: McGraw-Hill, 1997.

[MZQ98] R. Malli, X. Zhang, and C. Qiao, “Benefits of multicasting in all-optical networks”, in *Proc. SPIE All-Optical Networking*, vol. 2531, pp. 209-220, Nov. 1998.

[MN99] W.M. Moh, B. Nguyen, “An optimal QoS-guaranteed multicast routing algorithm with dynamic membership support”, in *Proc. IEEE ICC’99*, vol. 2, pp. 727-732, June 1999.

[Muk06] B. Mukherjee, “Optical WDM Networks”, Springer, Jan. 2006.

[Mai08] M. Maier, “Optical Switching Networks”, Cambridge University Press, Feb. 2008.

P

[Prim57] R.C. Prim, “Shortest connection networks and some generalizations”, *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

Bibliography

[PZG98] M. Parsa, Q. Zhu, J. Garcia-Luna-Aceves, “An iterative algorithm for delay-constrained minimum-cost multicasting”, *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 461-474, Aug. 1998.

[PGA03] I. Pepelnjak, J. Guichard, J. Apcar, “MPLS and VPN Architectures, Volume II”, Cisco Press, June 2003.

R

[Ram96] S. Ramanathan, “Multicast tree generation in networks with asymmetric links”, *IEEE/ACM Trans. on Networking*, vol. 4, no. 4, pp.558-568, Aug. 1996.

[RS98] R. Ramaswami and K. Sivarajan, “Optical Networks: A practical Perspective”, Academic Press, 1998.

[RMM98] S. Raghavan, G. Manimaran, C. Siva Ram Murthy, “Preferred link based delay constrained least cost routing in wide area networks”, *Computer Commun.*, vol. 21, no. 18, pp.1655-1669, Dec. 1998.

[RC99] D. Rizzetto, C. Catania, “A voice over IP service architecture for integrated communications”, *IEEE Network*, vol. 13, no. 3, pp. 34-40, May 1999

[RMM99] S. Raghavan, G. Manimaran, C. Siva Ram Murthy, “A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees”, *IEEE/ACM Trans. on Networking*, vol. 7, no. 4, pp. 514-529, Aug. 1999.

Bibliography

[Rou03] G.N. Rouskas, "Optical layer multicast: Rationale, building blocks, and challenges", *IEEE Network*, vol. 17, no. 1, pp. 60-65, Feb. 2003.

[RK05] A.A.Rehman, A. Karim, "Modeling and Performance Analysis of WDM Switching Networks with a Limited Number of Wavelength Converters", in *Proc. IEEE INMIC'05*, pp. 1-6, Dec. 2005.

[RZVZ06] Z. Rosberg, A. Zalesky, H.L. Vu, M. Zukerman, "Analysis of OBS Networks With Limited Wavelength Conversion", *IEEE/ACM Trans. on Networking*, vol. 14, no. 5, pp. 1118-1127, Oct. 2006.

[RM06] N. Rammohan, C.S.R. Murthy, "On-line multicast routing with QoS constraints in WDM networks with no wavelength converters", *Computer Networks*, vol. 50, no. 18, pp.3666-3685, Dec. 2006.

S

[SK97] A. Shaikh, S. Kang, "Destination-driven routing for low-cost multicast", *IEEE J. Sel. Areas Commun.*, vol. 15, no.3, pp.373-381, Apr. 1997.

[SRV97a] H.F. Salama, D.S. Reeves, Y. Viniotis, "A distributed algorithm for delay-constrained unicast routing", in *Proc. IEEE INFOCOM'07*, vol.1, pp.84-91, Apr. 1997.

[SRV97b] Salama, H.F., Reeves, D.S., Viniotis, Y., "The delay-constrained minimum spanning tree problem", in *Proc IEEE ISCC'97*, pp.699-703, July 1997.

Bibliography

[SL98] Q. Sun, H. Langendoerfer, “An efficient delay-constrained multicast routing algorithm”, *Journal High-Speed Networks*, vol. 7, no. 1, 1998.

[SAS99] S. Subramaniam, M. Azizoglu, A.K. Somani, “On the optimal placement of wavelength converters in wavelength-routed networks”, *IEEE/ACM Trans. on Networking*, vol. 17, no. 5, pp. 754-766, Oct. 1999.

[SM99] L.H. Sahasrabudhe and B. Mukherjee, “Light-trees: Optical multicasting for improved performance in wavelength-routed networks,” *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 67-73, Feb. 1999.

[SB99] T.E. Stern and K. Bala, “Multiwavelength Optical Networks – A Layered Approach”, Reading, MA: Addison-Wesley, 1999.

[Str00] B. Stroustrup, “The C++ Programming Language: Special Edition”, Addison-Wesley Professional, Feb. 2000.

[SBCLC01] G. Shen, S.K. Bose, T.H. Cheng, C. Lu and T.Y. Chai, “Performance analysis under dynamic loading of wavelength continuous and non continuous WDM networks with shortest path routing”, *Int. J. Commun. Syst.*, vol. 14, no. 4, pp. 407-418, Apr. 2001.

[SM02] A. Striegel, G. Manimaran, “A survey of QoS Multicasting Issues”, *IEEE Comm. Mag.*, pp.82–85, June 2002.

Bibliography

[SSM06] N.K. Singhal, L.H. Sahasrabudde, B. Mukherjee, “Optimal Multicasting of Multiple Light-Trees of Different Bandwidth Granularities in a WDM Mesh Network With Sparse Splitting Capabilities”, *IEEE/ACM Trans. on Networking*, vol. 14, no. 5, pp. 1104-1117, Oct. 2006.

T

[TM80] H. Takahashi and A. Matsuyama, “An approximate solution for the Steiner problems in graphs”, *Math Japonica*, vol. 24, pp. 573-577, 1980.

[TS00] T. Tripathi, K.N. Sivarajan, “Computing approximate blocking probabilities in wavelength routed all-optical networks with limited-range wavelength conversion”, *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 2123-2129, Oct. 2000.

[TS06] T. De, S. Sen, “Multicast routing and wavelength assignment in sparse splitting all optical networks”, in *Proc. IFIP WOCN’06*, 5 pp. , Apr.2006.

U

[UZ02] H. Ural, K. Zhu, “An efficient distributed QoS based multicast routing algorithm”, in *Proc. IEEE IPCCC’02*, pp. 27-36, Apr. 2002.

V

[VSK99] K.R. Venugopal, M. Shivakumar, P.S. Kumar, “A heuristic for placement of limited range wavelength converters in all-optical networks”, in *Proc. IEEE INFOCOM’99*, vol. 2, pp. 908-915, Mar. 1999.

Bibliography

W

[Win87] P. Winter, “Steiner problem in networks: A survey,” *Networks*, vol. 17, no. 2, pp. 129–167, 1987.

[Wax88] B.M. Waxman, “Routing of multipoint connections”, *IEEE J. Sel. Areas Commun.*, vol.6, pp. 1617-1622, Dec. 1988.

[Will99] B. Williamson, “Developing IP Multicast Networks, Volume I”, Cisco Press, Oct. 1999.

[WLF99] P.-J. Wan, L. Liu, O. Frieder, “Optimal placement of wavelength converters in trees and trees of rings”, in *Proc. IEEE ICCCN’99*, pp. 392-397, Oct. 1999.

[Witt00] R. Wittmann, “Multicast Communication: Protocols, Programming, and Applications”, Morgan Kaufmann, May 2000.

[WLW01] Y. Wang, L. Li, S. Wang, “A new algorithm of design protection for wavelength-routed networks and efficient wavelength converter placement”, in *Proc. IEEE ICC’01*, vol. 6, pp. 1807-1811, Jun. 2001.

[WWY01] K.-D. Wu, J.-C. Wu, C.-S. Yang, “Multicast routing with power consideration in sparse splitting WDM networks”, in *Proc. IEEE ICC’01*, vol. 2, pp. 513-517, Jun. 2001.

Bibliography

[WQY06] J. Wang, X. Qi, M. Yang, “Routing and wavelength assignment for core-based tree in WDM networks”, *Computer Commun.*, vol. 29, no. 11, pp.1896-1904, July 2006.

[WWL08] X. Wang, S. Wang, L. Li, “Multicasting on translucent multicast capable WDM mesh networks”, *AEU – Int. J. Electronics and Commun.*, vol. 62, no. 5, pp. 338-348, May 2008.

X

[Xue99] G. Xue, “Provably good approximation to minimum cost delay-constrained multicast trees”, in *Proc. ICCCN’99*, pp. 610-614, Oct. 1999.

[XR04] Y. Xin, G.N. Rouskas, “Multicast routing under optical layer constraints”, in *Proc. IEEE INFOCOM’04*, vol. 4, pp. 2731-2742, Mar. 2004.

Y

[Yen71] J. Y. Yen, “Finding the k shortest loopless paths in a network,” *Manage. Sci.*, vol. 17, no. 11, pp. 712–716, 1971.

[YRL99] J. Yates, M. Rumsewicz, and J. Lacey, “Wavelength converters in dynamically reconfigurable WDM networks”, *IEEE Commun. Surveys & Tutorials*, vol. 2, no. 2, pp. 2-15, Second Quarter 1999.

[YAJ01] S. Yan, M. Ali, J. Deogun, “Route optimization of multicast sessions in sparse light-splitting optical networks”, in *Proc. IEEE GLOBECOM’01*, vol. 4, pp. 2134-2138, Nov. 2004.

Bibliography

[YL03] D.N. Yang, W. Liao, "Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks," in *Proc. IEEE INFOCOM'03*, vol. 1, pp. 32-41, Apr. 2003.

[YY05] M. Yang, Y. Yang, "Constructing minimum cost dynamic multicast trees under delay constraint", in *Proc. ICCCN'05*, pp. 133-138, Oct. 2005.

[YPC08] K.-M. Yong, G.-S. Poo, T.-H. Cheng, "Efficient Heuristic for Minimum Cost Trees Construction in Multi-Groups Multicast", in *Proc IFIP-Networking'08*, pp. 215-226, May 2008.

Z

[ZWQ00] X. Zhang, J.Y. Wei, C. Qiao, "Constrained multicast routing in WDM networks with sparse light splitting", *IEEE J. Lightwave Tech.*, vol. 18, no. 12, pp. 1917-1927, Dec. 2000.

[ZAM02] Taieb F. Znati, Tawfig Alrabiah, Rami Melhem, "Low-cost, delay-bounded point-to-multipoint communication to support multicasting over WDM networks", *Computer Networks*, vol. 38, no. 4, pp. 423-445, Mar. 2002.

[ZPCSZ08] Y. Zhou, G.-S. Poo, S. Chen, P. Shum, and L. Zhang, "Dynamic multicast routing and wavelength assignment using generic graph model for wavelength-division-multiplexing networks," *IET Commun.*, vol. 3, no. 7, pp. 951-959, Aug.2008.