Cloud Resource Provisioning and Bandwidth Management in Media-Centric Networks

Beheer van cloudbronnen en bandbreedte in media-centrische netwerken

Maryam Barshan

Promotoren: prof. dr. ir. F. De Turck, prof. dr. B. Volckaert
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017 - 2018

ISBN 978-94-6355-049-9
NUR 986, 988
Wettelijk depot: D/2017/10.500/84

Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

Promoters: prof. dr. ir. Filip De Turck
prof. dr. Bruno Volckaert

Jury members: prof. dr. ir. Hendrik Van Landeghem
prof. dr. ir. Filip De Turck
prof. dr. Bruno Volckaert
prof. dr. ir. Remi Badonnel
dr. ir. Glenn Van Wallendael
dr. Wouter Tavernier
Wim Sandra
prof. dr. ir. Steven Latré

Ghent University
Faculty of Engineering and Architecture

Department of Information Technology
Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium

Tel.: +32-9-331.49.00
Fax.: +32-9-331.48.99

Dissertation to obtain the degree of
Doctor of Computer Science Engineering
Academic year 2017-2018

*This dissertation is dedicated to my loving husband,
Reza, for his selfless support and encouragement
and my wonderful daughter, Elina, who was
born and raised during my PhD studies.*

Acknowledgment

At this important stage of my life, it is time to step back and appreciate the contributions of all those who stayed with me, supported me, inspired me and influenced me throughout this challenging yet fascinating journey. I would like to express my sincere appreciation and thanks to all who contributed during last four years to pursue a successful PhD degree.

First of all, I am deeply grateful to my supervisors, Prof. Filip De Turck and Prof. Bruno Volckaert for their immense knowledge, critical reviews and constructive feedback and I sincerely appreciate their continuous guidance, supports and encouragements. I would also like to acknowledge the assistance of Dr. Hendrik Moens, whose research ideas, thoughtful and challenging discussions helped me a long way in completing this thesis.

Besides my supervisors, I would like to thank the rest of my thesis committee for taking their time to critically review my thesis and for providing feedback to improve this book: Prof. Remi Badonnel, Dr. Glenn Van Wallendael, Dr. Wouter Tavernier, Prof. Steven Latré and Mr. Wim Sandra. I am also deeply grateful to Prof. Hendrik Van Landeghem to be the chair of my defense committee.

This work has been mainly conducted within the context of ICON MECaNO. I am thankful to the project partners, SDNsquare, Limecraft, VideoHouse, Nokia and VRT for hours of thoughtful discussions, a driving source towards the completion of this work.

It was great to work at IDLab–IMEC and the support provided was very helpful in completing my PhD studies. I was lucky to be surrounded by supportive staff and colleagues, right from the start of my studies, who helped me a lot in settling in and starting my research. I am thankful to Martine and Davinia and other staff for their constant support and I was fortunate to have Ankita, Pieter-Jan, Thomas, Wannes, Rafael, Lucas, Dries and Jose as my office mates.

I spent around four years of my life in Gent and these days will stay in my memory for the rest of my life. I would like to thank all friends I met there, Bahare, Sahel, Fatemeh, Nasrin, Leila, Amin, Mahdi, Mostafa, Hemen and Foad, who made my stay pleasant and enjoyable.

I consider myself very fortunate for having long-term best friends, among them I am extremely thankful to Maryam and Zeinab for their thoughtful discussions and long-distance inspirations.

I also want to express my deep appreciation to the expert team at Newtec, where I have started working for a few months. I feel especially grateful to Thomas Van Den Driessche, Kristof Geilenkotten, Dr. Bart Duysburgh, Jan Geirnaert, Marc De Baerdemaeker and Jan Van Puymbroeck for their kind support.

I would like to express my thanks and gratitude to my family members: my parents, parents-in-law, brother, brother- and sisters-in-law for encouraging and supporting me spiritually throughout my studies and my life in general.

Last but not the least, there would never be enough words to appreciate the contributions of my husband, Reza, in my success. He is my first listener and my most dependable friend. I am extremely lucky to have him in my life and I am truly obliged for his constant support and love. We are blessed with an adorable little angel, Elina, whose blissful presence is a true blessing for us and I am thrilled to have her in my life. Her smile has been a powerful encouragement for me to complete my doctoral studies. The beauty of her smile and the sound of her laughter make even the hardest of days easy.

Gent, October 2017
Maryam Barshan

Table of Contents

Acknowledgment **iii**

Samenvatting **xxix**

Summary **xxxiii**

1 Introduction **1**
1.1 Communication network revolution . . . 1
1.2 Media-centric networks . . . 2
1.3 Problem statement . . . 3
1.4 Terminology . . . 4
1.5 Research contributions . . . 5
1.6 Outline of this dissertation . . . 11
1.7 Publications . . . 12
1.7.1 Publications in international journals (listed in the Science Citation Index) . . . 13
1.7.2 Publications in international conferences (listed in the Science Citation Index) . . . 13
1.7.3 Publications in book chapters . . . 15
1.7.4 Publications in other international conferences . . . 15
References . . . 16

2 Deadline-aware Advance Reservation Scheduling Algorithms for Media Production Networks **19**
2.1 Introduction . . . 20
2.2 Related work . . . 21
2.3 Media production network architecture . . . 23
2.4 Advance reservation scheduling model . . . 25
2.4.1 Decision variables . . . 26
2.4.2 Objective function . . . 26
2.4.3 Flow constraints . . . 28
2.4.4 Interdependent requests . . . 29
2.4.5 On-line model . . . 30
2.5 Advance reservation algorithms . . . 30
2.5.1 Static & dynamic reservation . . . 30

2.5.2 ILP based advance reservation algorithms (ILP) 31
2.5.2.1 ILP based Static Advance Reservation Algorithm ($SARA_{ILP}$) 31
2.5.2.2 ILP-based Dynamic Advance Reservation Algorithm ($DARA_{ILP}$) 32
2.5.3 Sequential Priority Based advance reservation algorithms (SPB) 32
2.5.3.1 Sequential Priority Based Static Advance Reservation Algorithm ($SARA_{SPB}$) 33
2.5.3.2 Sequential Priority Based Dynamic Advance Reservation Algorithm ($DARA_{SPB}$) 37
2.6 Experimental Results 37
2.6.1 Evaluation Setup 37
2.6.2 Comparing the SPB algorithms to the ILP-based algorithms 40
2.6.3 Evaluation of the SPB algorithms 41
2.6.3.1 Impact of available bandwidth 41
2.6.3.2 Impact of time slot granularity 43
2.6.3.3 Impact of network load 45
2.7 Conclusion 46
References 49

3 Design and Evaluation of a Dual Dynamic Adaptive Reservation Approach in Media Production Networks 53
3.1 Introduction 54
3.2 Related work 56
3.2.1 Advance resource reservation 56
3.2.2 Resilient reservation 56
3.2.3 Media production networks 57
3.3 Runtime adaptation approach in media production networks . . . 58
3.3.1 Envisioned media production network 58
3.3.2 Runtime adaptation (RA) methodology 61
3.3.3 First phase: Periodic update 63
3.3.4 Second phase: Periodic adaptation 64
3.3.5 Modeling of the runtime adaptation methodology 64
3.4 Runtime Adaptation (RA) algorithms 65
3.4.1 Periodic update algorithms 65
3.4.2 Periodic adaptation algorithms 65
3.4.3 Clarifying examples 68
3.5 Performance evaluation 74
3.5.1 Evaluation Setup 75
3.5.2 Impact of different failure rates, fixed backup demand . . 76
3.5.2.1 Impact of available bandwidth 76
3.5.2.2 Impact of network load 78
3.5.3 Impact of different backup demands, fixed failure rate . . 78

3.5.3.1 Impact of available bandwidth 78
3.5.3.2 Impact of network load 80
3.5.3.3 Stressed versus non-stressed network conditions 85
3.5.4 Impact of different backup demands, varying failure rates . 85
3.5.5 Evaluation of execution times 86
3.6 Conclusions 86
References 92

4 A Flexible, Reliable and Adaptive Timeslot-based Advance Bandwidth Reservation Mechanism for Media-centric Networks 97
4.1 Introduction 98
4.2 Related work 100
4.3 Advance Reservation for media delivery services 101
4.3.1 Type of reservation requests 101
4.3.1.1 STSD (Specified Time, Specified Duration) and UTSD (Unspecified Time, Specified Duration) requests 102
4.3.1.2 STUD (Specified Time, Unspecified Duration) and UTUD (Unspecified Time, Unspecified Duration) requests 103
4.3.2 Time domain classification in AR approaches 103
4.3.3 The resilient AR scheduling approach 104
4.3.4 Runtime adaptation (RA) approach 104
4.3.4.1 First phase: Periodic update 105
4.3.4.2 Second phase: Periodic adaptation 105
4.4 Problem description 106
4.4.1 Flexible approach 106
4.4.2 Optimized resilient approach 107
4.4.3 Combining dynamic, flexible, resilient and RA approaches 108
4.4.3.1 Impact of RA on advance reservation approaches 109
4.5 Advance bandwidth reservation architecture 110
4.5.1 FixedTimeSlot module 110
4.5.2 FlexibleTimeSlot module 112
4.5.3 Runtime adaptation module 112
4.6 Advance bandwidth reservation algorithms 113
4.6.1 FlexibleTimeSlot algorithm 114
4.6.2 BWallocationFBResilient algorithm 117
4.7 Evaluation setup 118
4.8 Simulation results and discussion 119
4.8.1 Comparing DARA fixed and DARA flex 121
4.8.2 Resilient DARA fixed vs. resilient DARA flex 121
4.8.3 Resilient DARA fixed+RA vs. resilient DARA flex+RA . 125
4.8.4 Discussion 130
4.9 Conclusions 131
References 132

5 Algorithms for Network-Aware Application Component Placement for Cloud Resource Allocation **137**
5.1 Introduction 138
5.2 Related Work 140
5.3 Modeling of a large-scale cloud environment 142
5.4 Formal ILP-based problem formulation 145
5.4.1 Introduction to the model 145
5.4.2 Decision variables 145
5.4.3 Objective function 147
5.4.4 Constraints 148
5.4.4.1 Physical node limitations 148
5.4.4.2 Physical link limitations 149
5.4.4.3 Quality of service requirements 149
5.4.4.4 Well-connected mapping Constraints 149
5.4.4.5 Full deployment constraints 150
5.4.4.6 Anti-collocation constraints 150
5.4.4.7 Additional constraints 150
5.5 Algorithm descriptions 151
5.5.1 ILP-based algorithm 151
5.5.2 Heuristic algorithm 151
5.5.2.1 Centralized Cloud Mapping Algorithm (CCMA) 152
5.5.2.2 Hierarchical Cloud Mapping Algorithm (HCMA) 152
5.6 Evaluation Details 155
5.6.1 Comparing CCMA to the state-of-the-art solutions 158
5.6.1.1 Evaluation Set up 158
5.6.1.2 Evaluation Results 158
5.6.2 Comparing the CCMA to the ILP-based algorithm 158
5.6.2.1 Evaluation Set up 158
5.6.2.2 Evaluation Results 160
5.6.3 Comparing the hierarchical algorithm to the centralized approach 161
5.6.3.1 Evaluation Set up 163
5.6.3.2 Evaluation Results 164
5.6.4 Large scale scenarios 169
5.6.4.1 Evaluation Set up 169
5.6.4.2 Evaluation Results 169
5.6.5 Evaluation discussion 169
5.7 Conclusions 171
References 172

6 Conclusion **179**
6.1 Advance reservation in media-centric networks 179
6.2 Flexible and fixed time slot size AR approaches 180
6.2.1 Fixed timeslot sizes 180

6.2.1.1 Request characteristics in media production industries . . . 180
6.2.1.2 Predefined size of fixed-size time slots . . . 181
6.2.2 Flexible timeslot sizes . . . 182
6.2.2.1 Dependency to the network load . . . 182
6.2.2.2 Irregular network devices' reconfiguration . . . 183
6.2.2.3 Impractical timeslot duration . . . 183
6.2.3 Discussion . . . 183
6.3 Resilient advance reservation approaches . . . 184
6.4 Impact of runtime adaptation approach . . . 185
6.5 Cloud-based application placement . . . 186
6.6 Future research . . . 186
6.6.1 Alternative use cases . . . 187
6.6.2 Timeslot-based technique . . . 187
6.6.3 The underlying technology . . . 187
6.6.4 Live migration of application components . . . 188

A Single-path versus Multi-path Advance Reservation in Media Production Networks 189
A.1 Introduction . . . 190
A.2 Related work . . . 191
A.3 AR scheduling model . . . 192
A.3.0.1 Additional decision variable . . . 193
A.3.0.2 Additional constraints . . . 193
A.4 AR scheduling algorithm . . . 193
A.5 Experimental results . . . 195
A.5.1 Evaluation Setup . . . 196
A.5.2 ILP evaluation of single-path versus multi-path . . . 196
A.5.3 Comparison of ILP-based model with SPB approach . . . 196
A.5.4 Evaluation of single-path and multi-path in SPB approach 197
A.6 Conclusion . . . 200
References . . . 201

List of Figures

1.1 Media production network infrastructure in Flanders region in Belgium. 6
1.2 A use case scenario in the media production industry. 6
1.3 Timeslot-based planning, scheduling and reservations. 7
1.4 Priority-based reservation of active requests in the current timeslot. 8
1.5 Comparing the resilient advance bandwidth reservation algorithm in theory, in practice and further re-optimization. 9
1.6 Schematic overview of the different chapters in this dissertation . 11

2.1 Media production network architecture and components. 24
2.2 Different components of the Sequential Priority Based Advance Reservation Algorithm (SPB). 32
2.3 Interactions between media production actors in the three considered use case scenarios. 38
2.4 The smaller Media production network topology used in the evaluation. 39
2.5 The Larger Media production network topology used in the evaluation. The ATT North America topology consists of 25 nodes and 56 links. 39
2.6 Impact of bandwidth capacity and percentage of known requests on admission rate. 40
2.7 Impact of percentage of known requests on admission rate in ILP-based algorithms. 41
2.8 Impact of percentage of known requests on admission rate in SPB algorithms. 42
2.9 The execution time of SPB and ILP-based approaches. 42
2.10 Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 12-node topology. 42
2.11 Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 25-node topology. 43
2.12 Impact of timeslot granularity on request admission rate for 12-node topology. 44
2.13 Impact of timeslot granularity on request admission rate for 25-node topology. 44

2.14 The average execution times of different time slot sizes and percentage of known requests for 12-node topology. 45
2.15 The average execution times of different time slot sizes and percentage of known requests for 25-node topology. 45
2.16 Impact of network load on request admission rate for 12-node topology. 46
2.17 The execution times of different number of scenarios and percentage of known requests for 12-node topology. 46
2.18 Impact of network load on request admission rate for 25-node topology. 47

3.1 Different components of media production network. 58
3.2 The primary and backup schedules provided by DARA. 60
3.3 Dependency of backup demand on the reserved primary paths. (Blue: Primary reservation, Red and dashed: Backup reservation) . 60
3.4 Comparing the DARA algorithm in theory, in practice and the RA approach contribution. 61
3.5 The collaboration between different components in every timeslot in the RA approach. 62
3.6 Algorithms used in periodic update and periodic adaptation phases of the RA approach. Narrow arrows show invocations. 66
3.7 Comparing a snapshot view of bandwidth reservations for 3 requests as output of the DARA algorithm and as input of the AO algorithm (Full lines: primary reservations, dashed lines: backup reservations, Open arrows: file-based transfers). 69
3.8 Differences between bandwidth allocation algorithms in the DARA and RA approaches. 70
3.9 The impact of video streaming requests activation / deactivation on file transfer finish time in the periodic adaptation phase of the RA approach. 71
3.10 Single timeslot reservations made by the DARA algorithm and multiple re-invocations of the AO algorithm during the periodic adaptation phase of the RA approach. 73
3.11 The impact of invocations of algorithms in periodic update and periodic adaptation phases of the RA approach on performance of reservation system. 74
3.12 Media production network topologies used for evaluation. 75
3.13 Bandwidth contention per link for 20 scenarios in 8-node topology and 50 scenarios in 25-node topology. 77
3.14 Impact of network capacity and failure rates on the performance of using the RA approach for the 8-node topology. 78
3.15 Impact of network capacity and failure rates on the performance of using the RA approach for the 25-node topology. 79
3.16 Impact of network load and failure rates on the performance of using the RA approach for the 8-node topology. 79

3.17 Impact of network load and failure rates on the performance of using the RA approach for the 25-node topology. 80
3.18 Impact of available bandwidth and backup demands on the performance of using the RA approach for the 8-node topology. 81
3.19 Impact of available bandwidth and backup demands on the performance of using the RA approach for the 25-node topology. 82
3.20 Impact of network load and backup demands on the performance of using the RA approach for the 8-node topology. 83
3.21 Impact of network load and backup demands on the performance of using the RA approach for the 25-node topology. 84
3.22 Final state of requests in stressed and non-stressed 8-node topology with a failure rate of 2h. 87
3.23 Final state of requests in stressed and non-stressed 25-node topology with a failure rate of 2h. 88
3.24 Comparing the success rate of the DARA and the RA approaches with different backup demands for the 25-node topology. 89
3.25 Evaluation of execution times for the 8-node topology. 90
3.26 Evaluation of execution times for the 25-node topology. 90

4.1 Types of advance reservation requests. 102
4.2 Time domain classification of AR approaches. 103
4.3 Comparing the original and modified versions of resilient fixed size timeslot-based advance bandwidth reservation. 108
4.4 Comparing the original and modified versions of resilient flexible timeslot-based advance bandwidth reservation. 109
4.5 A comprehensive overview of the control plane of the adaptive advance bandwidth reservation system. 111
4.6 Comparing the impact of network capacity in the flexible and fixed size DARA approaches in the 8-node topology. 120
4.7 Comparing the impact of network capacity in the flexible and fixed size DARA approaches in the 25-node topology. 120
4.8 Comparing the performance of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 8-node topology. Network capacity is 200Mbps. 122
4.9 Comparing the performance of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 25-node topology. Network capacity is 100Mbps. 122
4.10 Comparing the execution time of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 8-node topology. Network capacity is 200Mbps. 123
4.11 Comparing the execution time of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 25-node topology. Network capacity is 100Mbps. 123
4.12 Comparing the impact of backup demand in the flexible and fixed size resilient DARA approaches in the 25-node topology. 124

4.13 Comparing the impact of failure rates in the flexible and fixed size resilient DARA approaches in the 8-node topology. 124
4.14 Comparing the impact of runtime adaptation (RA) in the flexible and fixed size resilient DARA approaches in the 8-node topology. 125
4.15 Comparing the impact of runtime adaptation (RA) in the flexible and fixed size resilient DARA approaches in the 25-node topology. 126
4.16 Comparing the percentage of succeeded requests in the flexible and fixed size DARA approaches in the 8-node topology. 127
4.17 Comparing the success rate of admitted requests in the flexible and fixed size DARA approaches in the 8-node topology. 127
4.18 Comparing the percentage of succeeded requests in the flexible and fixed size DARA approaches in the 25-node topology. 128
4.19 Comparing the success rate of admitted requests in the flexible and fixed size DARA approaches in the 25-node topology. 128
4.20 Final state of requests with a failure rate of 10h in the 8-node topology. Backup demand is 50% and network capacity is 300Mbps. . 129
4.21 Final state of requests with a failure rate of 10h in the 25-node topology. Backup demand is 50% and network capacity is 200Mbps. 129
4.22 Final state of requests with a failure rate of 2h in the 8-node topology. Backup demand is 50% and network capacity is 300Mbps. . 129
4.23 Comparing the impact of backup demand on the final state of requests with a failure rate of 10h in the 8-node topology. Network capacity is 300Mbps. 130

5.1 The process of application component placement with the anti-collection placement requirement. Redundant components are not allowed to be allocated on the same host as non-redundant components. 139
5.2 The architecture of physical infrastructure and the management plane (LLM: Low Level Manager, MLM: Mid Level Manager, RLM: Root Level Manager). 144
5.3 The process of application component placement onto a cluster of cloud servers for two types of components (database server and computational server). 153
5.4 Different messages for interacting between the managers in GCMA (GCMA: Global Cloud Mapping Algorithm, CCMA: Centralized Cloud Mapping Algorithm). 155
5.5 An illustrative $20-$component application (Type 3). 157
5.6 Comparing the percentage of servers used in the CCMA and the ACUNA algorithm as a function of number of application placement requests (20 iterations). 158
5.7 Comparing the percentage of fully mapped applications in the CCMA and the ACUNA algorithm (20 iterations). 159

5.8 Comparing the percentage of anti-collocation application placement fulfillment in the CCMA and the ACUNA algorithm (20 iterations). . . . 159
5.9 Comparing the number of servers used (as bar charts) and the application mapping success rate (as line charts) in the CCMA to the ILP-based algorithm for 5-component applications (20 iterations). 160
5.10 Comparing the number of servers used (as bar charts) and the application mapping success rate (as line charts) in the CCMA to the ILP-based algorithm for 10-component applications (20 iterations). 161
5.11 Comparing the execution times of the CCMA to the ILP-based algorithm for 5-component applications (20 iterations). 162
5.12 The percentage of servers used (Case study 1 with 1000 servers and 20 iterations). 163
5.13 The percentage of used servers relative to the CCMA (Case study 1 with 1000 servers and 20 iterations). 164
5.14 The percentage of fully placed applications (Case study 1 with 1000 servers and 20 iterations). 164
5.15 The percentage of fully placed applications (Case study 1 with 1000 servers and 20 iterations). 165
5.16 The percentage of servers used (Case study 2 with 4096 servers and 20 iterations.) 166
5.17 The percentage of fully placed applications (Case study 2 with 4096 servers and 20 iterations). 166
5.18 The percentage of fully placed applications relative to the CCMA (Case study 2 with 4096 servers and 20 iterations). 167
5.19 The percentage of servers used (Case study 2 with 4096 servers and 20 iterations). 167
5.20 The percentage of fully placed applications (20 iterations). Number of physical servers $= \beta^3$. 168
5.21 The execution time per application (20 iterations). Number of physical servers $= \beta^3$. 168
5.22 The number of fully placed applications (10 iterations). Number of physical servers $= \beta^3$. 170
5.23 The execution time per application (10 iterations). Number of physical servers $= \beta^3$. 170

A.1 Components of the Sequential Priority Based Advance Reservation Algorithm (SPB). 194
A.2 Media production network topologies used in the evaluation. . . . 196
A.3 Comparing single-path versus multi-path in ILP-based approach. . 197
A.4 Comparison of optimal ILP with SPB approach. 197
A.5 Impact of bandwidth capacity, time slot granularity and network load on admission rate in 8-node topology 198
A.6 Impact of bandwidth capacity, time slot granularity and network load on admission rate in a 25-node topology. 199

List of Tables

1.1 An overview of the contribution characteristics per chapter in this dissertation. 12

2.1 Media production video request taxonomy. 25
2.2 Symbols and notations used in the formal models. 27
2.3 Details of the use case requests 39

3.1 Maximum and average number of concurrent failures for different failure rates in 8-node and 25-node topologies. 76
3.2 Execution time per algorithm invocation (ms) of main algorithms of the RA approach. Failure rate is 2h. 86

4.1 Maximum and average number of concurrent failures for different failure rates. 119

5.1 The Management plane parameters. 144
5.2 Symbols and notations used in the formal model. 146
5.3 The physical network parameters. 157
5.4 The Physical Infrastructure Specifications. 161
5.5 Management plane infrastructure. 162
5.6 Application specifications. 162
5.7 The number of physical devices based on different β values. . . . 168
5.8 The management plane parameters based on different β values. . . 168

6.1 The summary of the benefits and disadvantages of flexible and fixed-size timeslot based advance reservation approaches in media-centric industries. 184

A.1 Symbols and notations used in the formal model. 192

List of Acronyms

A

AR Advance Reservation
AO Adaptive Optimization
API Application Programming Interface

B

BFS Breadth-First Search

C

CCMA Centralized Cloud Mapping Algorithm

D

DARA Dynamic Advance Reservation Algorithm
DC Default Computational Server
DD Default Database Server

F

FB File Based

G

GCMA	Global Cloud Mapping Algorithm

H

HCMA	Hierarchical Cloud Mapping Algorithm

I

IEEE	Institute of Electrical and Electronics Engineers
ILP	Integer Linear Programming
IR	Immediate Reservation
IP	Internet Protocol

L

LAN	Local Area Network
LLM	Low Level Manager

M

ML	Management Layer
MLM	Mid Level Manager
Mbps	Megabit per second
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair

N

NSS	Next Server Selection

Q

QoS	Quality of Service

R

RLM	Root Level Manager
RA	Runtime Adaptation

S

SARA	Static Advance Reservation Algorithm
SS	Supported Server
SLA	Service Level Agreement
SPB	Sequential Priority Based
SBPP	Shared Backup Path Protection
SDN	Software Defined Network
STSD	Specified Time Specified Duration
STUD	Specified Time Unspecified Duration

T

TDD	Time Division Duplex
TDMA	Time Division Multiple Access

U

UTSD	Unspecified Time Specified Duration
UTUD	Unspecified Time Unspecified Duration

V

VM Virtual Machine
VS Video Streaming

W

WAN Wide Area Network
WDM Wavelength-Division Multiplexing

List of Symbols

$\|ML\|$	The number of management levels
$\|LLM\|$	The number of low level mangers
$\|MLM\|$	The number of mid level mangers
$\|SS\|$	The number of supported servers
μ	The branch factor of each tier
S_u	Available storage capacity of physical node *u*
M_u	Available memory capacity of physical node *u*
C_u	Available CPU capacity of physical node *u*
$D_{e_{uv}}$	Delay of physical link e_{uv}
$BW_{e_{uv}}$	Bandwidth capacity of physical link e_{uv}
$type_{e_{uv}}$	whether Phy. link e_{uv} is a LAN or WAN link
$Ccost_u$	Cost of each CPU unit of physical node *u*
$Mcost_u$	Cost of each memory unit of physical node *u*
$Scost_u$	Cost of each storage unit of physical node *u*
$BWcost_{e_{uv}}$	Cost of each BW unit of physical link e_{uv}
$fcost_u$	The fixed cost of using physical node *u*
$fcost_{e_{uv}}$	The fixed cost of using physical link e_{uv}
$AppNo$	Number of applications
$CompNo_a$	Number of components of application *a*
S_{type}	Set of types of application components
γ_{ai}^t	has value 1 if *ai* is of type *t*
c_{ai}	Computation demand of application *a*, comp. *i*
s_{ai}	Storage demand of application *a*, comp. *i*
m_{ai}	Memory demand of application *a*, comp. *i*
e_{aij}	Link between comp. *i* and *j* of application *a*
$bw_{e_{aij}}$	Bandwidth demand of link e_{aij}

$d_{e_{aij}}$	Max. allowed delay of link e_{aij}
d^n	Destination node of request r^n
t_s^n	Start time for the request r^n. Decision variable when not specified
t_e^n	Deadline for the request r^n. Decision variable when not specified
i^n	Duration of request r^n
b^n	Required bandwidth of r^n
v^n	Volume of r_f^n for file-based requests (in bit)
$\beta^{n,e,k}$	Decision variable. Dedicated bandwidth over link *e*, request r^n and time interval *k*
$x_u^{a,i}$	Binary decision variable which shows the accomplished mapping between component *i* of application *a* and physical node *u*, regardless of the type of component
$f_{e_{uv}}$	Binary decision variable which indicates success of mapping between physical link e_{uv} and the link between components *i* and *j* of application *a* (e_{aij})
T_u^t	Binary decision variable to determine whether node *u* is used to host components of type *t*
B_u	Binary decision variable. 1 iff physical node *u* is used, either as a routing node or a used server in the entire mapping
$B_{e_{uv}}$	Binary decision variable to indicate whether physical link e_{uv} is used in the mapping scheme or not
M_a	Binary decision variable which indicates whether the application *a* is fully mapped or not
$P^{n,e,k}$	Binary decision variable. 1 iff there is any reservation for request r^n over link *e* in time interval *k*
$SU^{n,k}$	Binary decision variable. 1 iff in time slot k any reservation is done for request n, 0 otherwise
A^n	Binary decision variable. 1 iff request r^n is admitted, 0 otherwise
A^s	Binary decision variable. 1 iff scenario s is admitted, 0 otherwise
I	Duration of each time interval (in second)
t_s^{min}	Minimum start time of all reservations
t_e^{max}	Maximum end time of all reservations
B^e	Bandwidth capacity of link *e*
E_v^{out}	This collection contains all edges starting from node v (egress)
E_v^{in}	This collection contains all edges ending in node v (ingress)

Samenvatting
– Summary in Dutch –

In de loop der jaren zijn mediacentrische industrieën complexer en gegevensintensiever geworden omdat ze te maken hebben met enorme hoeveelheden (hoge bitrate) data zoals volledige ultra-HD video-opnames en daarbovenop een geografisch verdeeld personeelsbestand. Grote hoeveelheden gegevens kunnen door verschillende samenwerkende divisies / bedrijven worden geanalyseerd, verwerkt en opgeslagen in (data-)magazijnen op verschillende locaties. Traditioneel werd de verspreiding van media over het algemeen uitgevoerd door ofwel mensen die de inhoud op een fysiek opslagmedium vervoeren, wat zeer inefficiënt is, of getransfereerd over toegewijde point-to-point hoge snelheid optische verbindingen, die duur zijn. Het verbinden van de verschillende actoren die betrokken zijn bij het productieproces door middel van een gedeelde netwerklaag zou de investeringen aanzienlijk verminderen en de netwerkbron-benutting kunnen verhogen. Momenteel worden dergelijke gedeelde (media-centrische) IP-gebaseerde netwerken uitgerold om media-samenwerking en uitwisseling van mediamateriaal tussen verscheidene geografisch verspreide actoren te faciliteren.

Mediacentrische omgevingen zijn zeer dynamisch door gelijktijdige aankomst en vertrek van meerdere media-overdrachten (zoals ruwe en gecodeerde audio- en videobestanden en streaming-sessies) met verschillende groottes en vereisten. Elke overdracht kan een vaste of ongespecificeerde start- en eindtijd hebben. In mediacentrische netwerken kunnen meerdere overdrachten ook afhankelijk zijn van elkaar. In mediaproductieprocessen bijvoorbeeld zal de starttijd van het versturen van bewerkt materiaal naar de zender afhangen van de eindtijd waarop opgenomen materiaal naar het productiebureau verstuurd werd. Deze onderlinge afhankelijkheid tussen verzoeken impliceert dat een verzoek niet kan worden gestart tenzij de andere verzoeken waarvan dit verzoek afhangt, al werden uitgevoerd. In deze studie overwegen we een reeks onderling afhankelijke en deadline-beperkte netwerkoverdrachten.

Een belangrijk kenmerk van het dataverkeer in media-centrische netwerken is de voorspelbaarheid ervan. De tijdsvereisten, locatie en bandbreedte van gegevensoverdrachten zijn vaak uren en soms zelfs dagen vooraf gekend. Zo zou het gebruik van bandbreedtereserveringstechnieken (Advance Reservation - AR) kunnen leiden tot sterk verhoogd bandbreedtegebruik/efficiëntie en verminderde kosten. In AR-netwerken verzoeken gebruikers toekomstige data-overdrachten, over het algemeen voorzien van een starttijd (ofwel onmiddellijk of op een gegeven tijdstip in de toekomst), een deadline en de totale gegevensoverdracht (of benodigde bandbreedte bij streamen). Vervolgens wijst een scheduling-algoritme de nodige netwerkbronnen toe om ervoor te zorgen dat alle toegestane aanvragen worden afgerond voor hun gespecificeerde deadline, terwijl gelijktijdig zoveel mogelijk transfer-verzoeken ingepland worden. Het is duidelijk dat AR verschillende voordelen heeft voor de volgende generatie media-afleveringsnetwerken. Het stelt netwerkbeheerders in staat om het gebruik van de netwerkbronnen beter te plannen, wat leidt tot sterk verhoogd gebruik van de middelen en gegarandeerde kwaliteit van dienstverlening (QoS). Als eerste bijdrage in dit proefschrift hebben we de levensvatbaarheid bewezen van het gebruik van AR planning in mediaproductienetwerken, om aanzienlijk de netwerkefficiëntie en transferverzoek-inwilligingsratio te verbeteren.

In de voorafgaande reserveringsbenaderingen gaan we om met dynamische tijdsafhankelijke reservaties. Om voorspelbare complexiteit, eenvoudiger implementatie en periodieke herconfiguratie van netwerkapparaten te kunnen voorzien, werden de AR-algoritmen ontworpen om gebruik te maken van vaste tijdslots. Onze resultaten laten zien dat in sommige gevallen echter een vaste tijdslot-aanpak de kwaliteit van het algoritme en de uitvoeringssnelheid negatief kan beïnvloeden. Als gevolg werd de mogelijkheid onderzocht om een flexibel tijdslot-gebaseerd netwerkreservatiemechanisme voor mediacentrische bedrijven te gebruiken. We analyseerden de voordelen en nadelen van het gebruik van flexibele tijdslots, en vonden dat een flexibele benadering zeer gunstig is bij het omgaan met bursty dataverkeer in netwerken met een relatief klein aantal aanvragen en periodes zonder netwerktransfers. De flexibele aanpak kan het aantal benodigde tijdsloten potentieel aanzienlijk verminderen, resulterend in een verbetering van de uitvoertijd. We hebben bandbreedtereservatie-algoritmes gebaseerd op zowel vaste als flexibele tijdssloten ontworpen, geïmplementeerd en geëvalueerd en de kwaliteit en complexiteit van deze aanpakken grondig vergeleken. We tonen aan dat flexibele tijdssloten van nature meer compatibel zijn met transferverzoeken in mediacentrische netwerken, aangezien bij flexibele tijdssloten de reservaties beter afgestemd kunnen worden op de timing van inkomende verzoeken.

Bandbreedtereservaties resulteren in hoger bandbreedtegebruik en verbeterde netwerkprestaties. In onbetrouwbare netwerken kan dit echter mislukken. Wanneer een bestand wordt getransfereerd in overeenstemming met het AR-schema, kunnen twee scenario's voorkomen: het bestand is volledig overgebracht binnen de geplande tijdsperiode, of door onzekere netwerkomstandigheden, zoals bijvoorbeeld plotse veranderingen in netwerkconfiguratie, netwerkfluctuaties, storingen, etc. is het bestand niet volledig overgedragen. Daarom is de betrouwbaarheid van de overdrachten ook een aandachtspunt voor dit proefschrift. Bij AR-mechanismen kunnen fouttolerantie-mechanismen toegevoegd worden om prestaties te verbeteren bij voorkomen van onvoorziene fouten en zo betrouwbare reservaties aan te bieden. In eerste instantie, zorgt het toepassen van beschermingsmechanismen ervoor dat het AR schema geldig blijft wanneer het systeem operationeel is. We hebben een resiliënte reservatiemethode voorgesteld geoptimaliseerd voor mediadistributienetwerken gebruik makend van backup-paden en redundante reservaties. Dit stelt het reservatiesysteem in staat om in aanwezigheid van fouten toch betrouwbare en consistente prestaties te leveren.

Het gebruik van redundantie zorgt echter voor een aanzienlijke prestatie-impact en extra kosten. In onstabiele netwerkomstandigheden is het dan ook belangrijk om het AR schema up-to-date te brengen naarmate de tijd verloopt, volgens de huidige toestand van het netwerk en de lopende transfers. Om de eerder genoemde negatieve impacten te verzachten hebben we een dynamische event-gedreven simulator ontwikkeld en geëvalueerd die runtime monitoring, adaptatie en heroptimalisatie kan simuleren. Constante monitoring en adaptatie zijn nodig om het transferschema dynamisch aan te passen aan de wijzigende omstandigheden / context. Het schema afkomstig van het geavanceerde reservatie-algoritme wordt periodiek aangepast, rekening houdend met de huidige toestand van netwerk en lopende transfers. Additioneel exploiteren we hierbij onderbenutte (backup-) netwerkcapaciteiten om meer data te transfereren dan wat ingepland was, zolang er geen storing gedetecteerd wordt. We tonen aan dat zowel in stabiele als foutgevoelige netwerken deze aanpak significant de netwerkutilisatie en hoeveelheid geaccepteerde transfers kan verhogen.

Mediacentrische netwerken gebruiken datacentra vaak om gegevens op te slaan en hun applicaties uit te voeren. Een datacenter herbergt computerfaciliteiten en kan cloud-gebaseerde infrastructuur aanbieden als een dienst. Huidige raamwerken hebben echter geen ondersteuning voor een brede waaier aan applicatieplaatsingsvereisten, zoals een combinatie van schaalbare componentplaatsing die rekening houdt met harde vereisten op vlak van netwerkfaciliteiten en beperkingen op vlak van bvb dienst-collocatie. Om dit probleem aan te pakken stellen we een optimaal wiskundig model voor, evenals gecentraliseerde en hiërarchische heuristische oplossingen, waarin elke applicatie wordt weergegeven als een verzameling van interactieve componenten met een onderscheid tussen componenttypes. Onze aanpak houdt rekening met de kenmerken van het onderliggende netwerk en schaalt tot grote cloud-gebaseerde datacenters.

Summary

Over the years, media centric industries have become more complex and more data intensive as they are dealing with enormous amounts of high bitrate data such as ultra-HD video recordings and a geographically distributed workforce. Large quantities of data may be analyzed and processed by various collaborating parties and stored at multiple warehouses on different locations. Traditionally, the distribution of media content is generally performed by either people transporting the content on a physical storage medium, which is highly inefficient, or over dedicated point-to-point high-speed optical links, which is expensive. Connecting the different actors involved in the media production process to a shared network substrate would greatly reduce capital expenditures and increase network resource utilization. Currently, such shared media centric networks, connecting many actors across a large geographical area, are being deployed to build collaborations over IP media contents and support the exchange of different media content.

Media-centric environments are highly dynamic due to the arrival and departure of several concurrent transfers of different sizes and requirements, such as raw and encoded audio and video files and streaming transfers. Each transfer can have a fixed or unspecified start and end time. In media centric networks, multiple transfers may depend on each other. For instance, in media production processes the start time of sending edited material to the broadcaster depends on the end time of sending recorded material to the production office. This interdependence among requests implies that a request can not be started unless the other requests on which this request depends, have already been transferred. In this study, we consider a set of interdependent and deadline-constrained network transfers.

A key characteristic of traffic in media-centric networks, is its predictability. The timing, locality and bandwidth requirements of data transfers are often known hours and sometimes even days in advance. As such, the use of advance bandwidth reservation (AR) techniques would result in greatly increased bandwidth utilization and reduced costs. In AR networks, users submit requests for future data transfers, generally encompassing a start time (either immediately or at some point in the future), a deadline, and total data transfer size (or rate). Subsequently, a scheduling algorithm allocates the necessary bandwidth resources to ensure that all admitted requests finish before their specified deadline, while admitting as many requests as possible. Clearly, AR has several advantages for next generation of media delivery networks. It allows network operators to better plan resource usage, leading to greatly increased resource utilization and guaranteed Quality of Service (QoS). As the first contribution in this dissertation, we have proven the viability of using AR scheduling in media production networks to significantly improve bandwidth efficiency and request admittance ratio.

In the advance reservation approaches, we deal with dynamic time-dependent reservations. To offer predictable complexity, easier implementation and periodic reconfiguration of network devices, the AR algorithms were designed making use of fixed time slots. Our results showed that in some cases however, using a fixed timeslot approach can negatively affect algorithm quality and execution speed. As a consequence, the possibility of using a flexible timeslot-based advance bandwidth reservation approach in media centric industries has been investigated. We have analyzed the benefits and drawbacks of using flexible timeslots, finding that the flexible approach is highly beneficial when dealing with bursty traffic conditions in a low-demand network with long-term downtimes. The flexible approach has the potential to significantly reduce the number of timeslots needed, resulting in execution speed improvements. We have therefore designed, implemented and evaluated advance bandwidth reservation algorithms based on both fixed and flexible timeslot sizes and thoroughly compared the quality and complexity of these approaches. We show that flexible timeslots are by nature more compatible with requests in media-centric networks since flexible timeslots could make the timings fit better with the timing of incoming requests.

Advance bandwidth reservation results in higher bandwidth utilization and improved network performance. However, in unreliable networks, this may fail. When a file is actually transferred in accordance with the AR schedule, two scenarios may occur: either the file is completely transferred within the scheduled time period, or due to uncertain network conditions, such as e.g. sudden changes in network configuration, network fluctuations, failures, it is not fully transferred. As such, reliability of transfers is another concern of this dissertation. In AR mechanisms, to offer reliable reservations, fault-tolerance features can be incorporated to improve performance and deal with unforeseen failures. As the first provisional stage, deploying protection mechanisms ensures that the schedule remains valid when the system is in operation. We have proposed a resilient advance reservation approach by provisioning backup paths and redundant reservations. This enables the reservation system to deliver reliable and consistent performance in the presence of failures.

Using redundancy imposes significant performance overheads and extra costs. In unstable network conditions, updating the schedule over time according to the current state of the network and transfers is also important. In order to mitigate these side-issues, we have further designed and evaluated a dynamic event-driven simulator in which monitoring, adaptation and re-optimization is applied at runtime. Constant monitoring and modification is required in order to be capable of dynamically adapting the transfer schedule to changing conditions. The schedule provided by the advance reservation algorithm is periodically updated, taking into account the current state of network and transfers. In addition, this approach exploits underutilized network capacities to transfer more data than what has been scheduled as long as no failure is detected. We show that in both stable and failure-prone networks, deploying this approach noticeably increases network utilization and request admittance ratio.

Media centric networks often use data centers to store data and run their applications. A data center is a repository that houses computing facilities which can offer cloud-based infrastructures as a service. However, current frameworks lack crucial features for supporting a broad range of application placement requirements, such as combination of scalable component-level placement, guaranteeing network and constraint-awareness requirements. To address this issue, we propose an optimal mathematical model as well as centralized and hierarchical heuristic solutions, in which each application is represented as a collection of interacting components with a distinction between application component types. Our approach takes the characteristic of the underlying network into account and scales to the size of large cloud-based datacenters.

1
Introduction

"Learning and innovation go hand in hand. The arrogance of success is to think that what you did yesterday will be sufficient for tomorrow."

–William Pollard (1911 - 1989)

1.1 Communication network revolution

The emergence of communication networks can be traced back to more than 140 years ago [1], when Alexander Graham Bell made his first call, on March 10, 1876, to his assistant Thomas Watson: "Mr. Watson–come here–I want to see you". This was the basis of telephone system's circuit switching method, which establishes a direct and temporary circuit to send continuous voice messages between two locations. Another form of communication came up almost one century later when the ARPAnet, the first packet-switched computer network, was conceived in the late 1960s. In packet switching [2], a message is broken into smaller units, called packets. The ARPAnet did not connect each pair of nodes directly, instead the intermediate nodes could forward the packets toward their final destinations. This was the starting point of ever-growing computer network infrastructures in the past half century.

The Internet, as we know it, grew from a different architectural concept. In 1972, by introducing the open architecture networking [3], networks with diverse underlying technologies were able to be connected to each other. This paved the way for the Internet, which embodies this key underlying technical idea.

Today, a computer network in any form and scale, ranging from Local Area Network (LAN) to Wide Area Network (WAN) and the Internet, is a powerful medium for communication. Use of computer networks has several advantages as it allows for resource sharing with higher performance and lower cost. It also provides centralized control for geographically distributed systems, distribution of processing functions, centralized or decentralized management and network resource allocation.

With the significant advances in information and communications technology over the last years, there is an increasingly perceived vision that computing can be considered the 5th utility after water, electricity, gas, and telephony. A number of computing paradigms have been proposed to deliver the long-held dream of computing as a utility, of which the recent one is known as cloud computing [4, 5]. Cloud computing is a new style of offering applications and services over the Internet. Cloud computing offers a promising alternative to traditional IT businesses, which had to build and maintain their own infrastructures. Recently, many companies use cloud technologies to reduce costs, increase flexibility and to respond faster to customer needs.

1.2 Media-centric networks

Nowadays, the use of digital techniques is becoming increasingly ubiquitous. The formerly separated forms of analog data, such as video and audio recording, are transformed to digital formats. All these time-dependent recordings, being turned into digital representations, are called media.

While media was originally often being used for entertainment purposes, it now becomes a widely pervasive part of various critical applications in many industries such as online surgeries, distance learning, video conferencing, traffic management control room, etc. These media-centric applications are characterized by a high need for fast and reliable data transport between different sectors.

The lack of an efficient technical solution that allows geographically distributed sectors to work together forces the media-centric networks to co-locate their buildings, in which the distribution of media content was generally performed by people transporting the content on a physical storage media. An alternative solution was to invest in time and cost consuming transfers over dedicated point-to-point high-speed optical links. However, these are respectively highly inefficient and costly methods. Industries with geographically distributed digital-collaborative workforce are striving to decrease the cost of these deadline-constrained transport processes.

The availability of wide area networks (WANs) between the locations of multiple sectors provides an opportunity for technical improvement. In order to reduce capital expenditures, support decentralized collaboration and increase network resource utilization, media related environments tend to switch to cost-effective WAN approaches. Deploying a shared WAN solution enables existing media content owners and their collaborators to work together in a cost effective way, while new actors can more easily find new collaboration opportunities, thus fostering the whole industry's further growth.

1.3 Problem statement

In the field of media transport with strict quality-of-service (QoS) demands, providing suitable connectivity of geographically spread locations remains challenging or expensive. As the first step to offer a cost-efficient solution supporting professional media collaboration, physical transport can be eliminated by transporting over a network. Network capacity is a valuable resource and efficient bandwidth management is of crucial importance [6], particularly for media content transfers that are dealing with ever increasing sizes to offer high resolution formats. Further cost reductions can be achieved if one network link allows transferring concurrently, thereby optimizing the use of the network capacity. Moreover, more reliable transport reduces the overhead of re-transmission of lost data. Since deadlines within the media sectors are of prime importance, for time-critical media processes, time savings in transport of contents or faster access to the archived materials are very valuable. Media related companies are willing to go a long way to reduce the time spent in transport and increase the productive time.

Over-provisioning of network resources [7] has been a common practice to deal with the peak traffic demands in long term. In media related networks with large file transfers, this requirement is much higher compared to traditional IT applications. In bandwidth-limited networks, bandwidth scheduling of large transfers will significantly reduce the need for this over-provisioning. Bandwidth scheduling refers to bandwidth allocations with flexible options with regards to timing and bandwidth requirements in both on-demand and in-advance reservation disciplines. Bandwidth reservation systems are generally capable of both advance and immediate reservations. The former allocate resources ahead of time in future, while the latter reserve resources upon availability in the next time intervals. In media centric networks, bandwidth requirements, timing constraint and locality of network transfers are mostly known hours or even days in advance. Overall, deploying advance bandwidth reservation techniques decreases the need for over-provisioning and increases the requests' admittance ratio and network utilization.

As advance bandwidth reservation is an essential feature of any shared network in which network capacity needs to be co-allocated at predetermined times, there have been significant investigations in literature to date [8–11]. The media-centric networks carry a combination of requirements which are not supported by existing state-of-the-art solutions. To start with, different types of transfer requests have different requirements. For file transfers, the start time of transfer is generally flexible, the deadline is fixed, and the reserved bandwidth may vary over the lifetime of the reservation. For streaming requests, a constant duration and a fixed amount of reserved bandwidth is associated from source to sink of the requests. Deadlines are of great importance, however, for both file transfers and streams start time, end times and deadlines of transfers are not necessarily known. This is because media related industries mainly deal with a set of interdependent transfers, implying that each transfer can only start when other transfers, on which this transfer depends, have been finished. This combination of deadline-conscious interdependent transfers with flexible times and elastic bandwidth allocation has not received much attention in the literature to date [12].

Media centric networks often make use of data centers to store and run different types of applications and complex workloads. A datacenter is composed of networked servers and storage facilities used to organize, process and store large amounts of data. This datacenter can also offer cloud-based infrastructure and house cloud services. With the increasing demand for cloud computing services, methods for efficient placement of requested applications into available cloud resources has drawn enormous attention and became an essential research problem since this placement is further complicated by the issues of shared data, data interdependencies and user concerns about efficiency, security and reliability. Each application placement request has a list of different component types, along with potential placement constraints. For instance, an application with data safety-sensitive services requires hardware-based security and an application requiring high-level fault tolerance has an anti-collocation placement constraint to safeguard service availability. The component placement process has to ensure the entire placement of such a component-based application, known as "full deployment" constraint, meaning that either all or none of the application components have to be placed. Moreover, due to interconnections among these components, network requirements of collaborative components has to be taken into account. The combination of the network-awareness requirements, full deployment and anti-collocation placement constraints has been challenging and remained, to the best of our knowledge, unexplored.

1.4 Terminology

In this section, definitions are provided for the most important concepts used throughout this dissertation:

Request: a request refers to a transfer of two types: streaming or file transfers. The n^{th} request is denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$, comprising of the source of the request s^n, the destination node d^n, the time when the data for a file-based request is ready to transfer t_s^n (or fixed start time for a streaming request), the deadline for the transmission of the data of a file-based request t_e^n (or fixed end time for a streaming request), the duration of each request i^n and finally the bandwidth demand of the request b^n. Moreover the volume of the files are denoted by v^n.

Scenario: a scenario is defined as a collection of interdependent file and streaming transfers. Due to inter-dependencies, if one of these transfers is not successful, the whole scenario will be affected.

timeslot: a time interval or time slot is a period of time in which bandwidth reservations remain invariant. A timeslot can be of flexible or fixed size.

Schedule: refers to a 3-dimensional allocation among transfer requests, network links and timeslots, indicating how much bandwidth is allocated to each transfer request over each link on each timeslot.

Static approach: the static approach assumes all requests are known at the start of the resource reservation period. The schedule remains fixed during the whole lifetime of reservations.

Dynamic approach: in the dynamic or online approach requests enter to the system at any time. The dynamic system re-schedules the already-scheduled and newly arrived requests together while the system is in operation. Re-accommodation of the previously admitted requests is a must.

Anti-collocation placement constraint: in the context of component-based application placement, this constraint ensures that the components of a given application cannot be placed on the same server, mainly for fault tolerance and isolation purposes.

1.5 Research contributions

Media-centric networks impose requirements not supported by existing advance scheduling techniques, such as different types of video or audio transfers, flexible or unspecified start or end times, strict deadlines, interdependent requests, reliability, etc. In addition, the problem of constraint- and network-aware application placement in large scale cloud-based datacenters has not gained enough attention to date. Addressing these unexplored aspects is the main contribution of this dissertation.

We pay specific attention to media production networks as a typical comprehensive example of media-centric networks. As shown in Figure 1.1, multiple actors are involved in media production network including central production, remote production, post-production, broadcasting, archiving locations, etc. The map shows the physical infrastructure of media production industry in the Flanders region of Belgium.

Media-centric industries generally deal with a set of multiple transfers of various types, which we referred to as a *scenario*. For more clarification, Figure 1.2

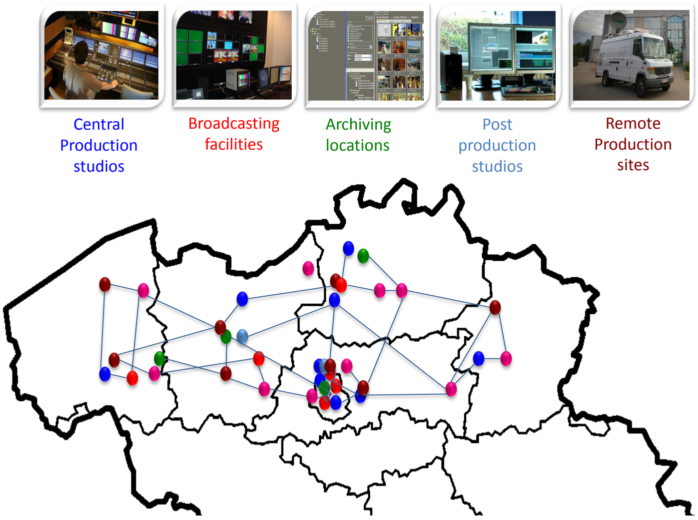


Figure 1.1: Media production network infrastructure in Flanders region in Belgium.

illustrates the process flow of a realistic scenario use case in a media production network, a weekly soccer after-game discussion program, consisting of multiple transfers. Each transfer is referred to as a *request*. As can be seen, first the soccer matches have to be recorded at the stadiums and along with on-site interviews sent to the production platform. Then, these raw contents and archived materials have to be transferred between different post-production locations for editing. With strict timing constraint, the live program will be broadcasted and then the recorded program will be sent to distinct locations to provide e.g. on demand access, archiving, etc. This use case scenario reveals the importance of timely transfer of inter-dependent requests.

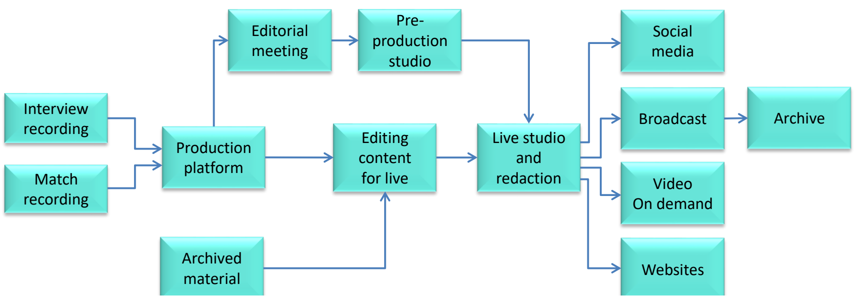


Figure 1.2: A use case scenario in the media production industry.

Scenario requests, supported in this dissertation, are of 4 different classes: independent streaming requests, independent file transfers, dependent streams and dependent file based transfers. The requests of independent type can be started at the specified start time but dependent requests have to wait until the requests upon which they are dependent have finished. Dependency among different transfers implies that either all or none of the interdependent requests must be admitted. We assume that volume for file-based requests, and duration for streaming requests must be specified. The allocated bandwidth for the streams must be equal to their required bandwidth demand, from the start time to the end time, because their demand is fixed. However, for file-based requests, the volume of file is the determinative factor. The file can be transferred whenever possible from the time the file is ready to be transferred till its deadline. The residual demand of file-based transfers is modified whenever a part of the file is transferred.

Dynamic network reservations can be discretized into time steps, referred as timeslots and bandwidth allocation algorithms can be applied efficiently on every snapshot of the network and produce a schedule which consists of distinct reserved capacities in every time interval for each admitted request. This timeslot-based planning, scheduling and reservation is shown in Figure 1.3. Timeslots can be based on flexible or fixed size ($T_i = T_j$ for any pairs of timeslots).

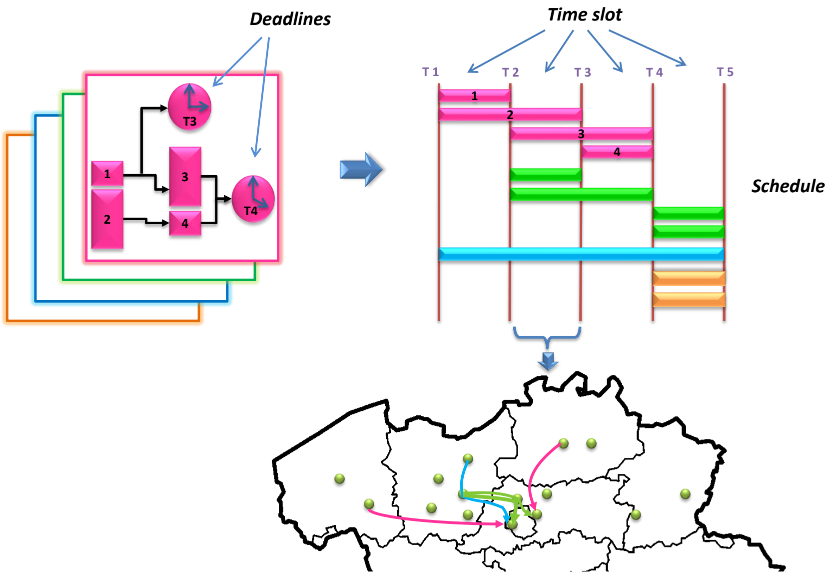


Figure 1.3: Timeslot-based planning, scheduling and reservations.

In each timeslot, the active requests are sorted according to their priorities. As has been shown in Figure 1.4, the prioritization process assigns priorities to the requests, taking the request deadline as the first determinative factor. This scheduling is employed by the reservation system as long as no new scenarios are submitted to the system. To improve the performance, strategies able to deal with unforeseen failures in the network are required. This reliability can be offered by provisioning backup paths, before any failure happens in the network. Dashed arrows in this figure indicate these provisioned backup reservations.

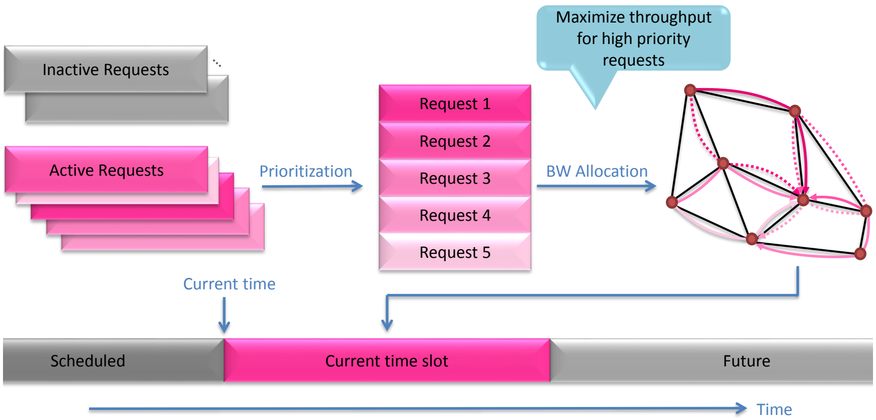
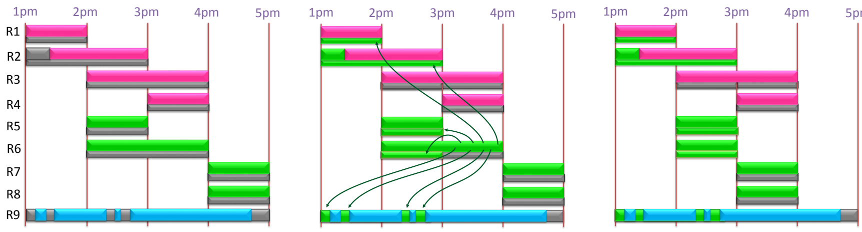


Figure 1.4: Priority-based reservation of active requests in the current timeslot.

To offer reliable reservations, the incorporation of fault tolerance features in bandwidth reservation strategies is a necessity. However, this incurs additional costs and extra performance overheads as network capacity remains unused to offer this protection. To reduce this waste, a constant monitoring, adaptation and re-optimization can be applied. As long as no failure is detected, underutilized network capacities can be exploited to transfer more data than what has been scheduled. This adaptation and optimization offers protection from failures using resilient advance reservation, while also improving the network utilization and request admittance ratio. To elaborate more on this, Figure 1.5 is depicted in which a fixed timeslot size of 1 hour is taken into account. In this figure, 8 file transfers (R1-R8) and one streaming request (R9) have been scheduled. The gray bars show the potential unused reservations, such as backup reservations as long as no failure is detected or time gap between streaming request resumes and play backs. This underutilized capacity can be exploited to transfer future data. For example, as shown in Figure 1.5b, in stable network conditions, R6 which has been scheduled to be transferred between 3pm and 4pm, can be transmitted earlier. Then, the schedule is periodically updated to adjust the transfer reservations according to the most recent network transfers and conditions, shown in Figure 1.5c.

Finally, we have targeted the issues of cloud based resource provisioning, in terms of scalability and combining network-awareness and placement constraints.



(a) Capacity remains unused in the reliable AR schedule. (b) Early transmission of future requests when possible. (c) Updating the schedule, freeing future capacity.

Figure 1.5: Comparing the resilient advance bandwidth reservation algorithm in theory, in practice and further re-optimization.

We assume that each application comprises a set of application components that should be placed in full on a cluster of servers while the communication requirements and the placement constraints of application components has to be respected.

To conclude, we present an overview of the main research contributions within this dissertation:

- Development of timeslot-based advance bandwidth reservation scheduling approaches, optimized for media production networks.
  - Proof of the viability of using AR scheduling in media production networks to significantly improve bandwidth efficiency and request admittance ratio.
  - Definition of a set of realistic use cases that serves as a basis for the evaluation use cases.
  - Design and implementation of static and dynamic, single-path and multi-path, Integer Linear Programming (ILP) formulations of the bandwidth scheduling problem based on fixed-size timelots, which takes into account the specific characteristics of media production networks.
  - Design and implementation of near-optimal static and dynamic, single-path and multi-path fixed size timeslot-based advance bandwidth reservation heuristic algorithms to address the complexity of optimal solutions.
  - Thorough evaluation of optimal and near-optimal scheduling approaches and quantitative analysis of the impact of time interval granularity, network capacity and network load on the quality and complexity of the proposed approaches.

- Design and implementation of the resilient advance bandwidth reservation and further optimization of this approach in order to improve network utilization and request admittance ratio.
- A thorough comparison analysis of the flexible and fixed timeslot sizes for the proposed advance bandwidth reservation approaches.
- Implementation and evaluation of the static and dynamic, resilient and non-resilient advance bandwidth scheduling algorithms based on flexible time slots.

• Design of a discrete-event-based simulator which improves both reliability and performance of advance reservation systems when the network is in operation.

- Design and implementation of a resilient advance bandwidth reservation system by incorporating fault tolerance related features in advance reservation strategies.
- Design and implementation of a dynamic adaptive runtime approach to mitigate the side-effect of extra overhead imposed by redundancy, continuously monitoring network conditions, dynamically adapting the transfer schedule and reacting to sudden changes in uncertain network conditions. This complementary component can be used in combination with both flexible and fixed size resilient timeslot-based advance reservation solutions.
- Quantitative evaluation of the dynamic adaptive solution in failure free and failure prone networks and analysis of the impact of backup demand, failure rate, stressed and non-stressed network conditions.

• A scalable hierarchical application placement approach in cloud-based environments.

- Network and constraint-aware ILP-based solution for cloud resource allocation. Application modeling is component-based with different component types, and interaction between those components affects the placement process.
- Design and implementation of near-optimal centralized and hierarchical application placement approaches to resolve the computational complexity associated with the optimal solutions.
- Comparison of the proposed algorithm to the optimal solution and a generic state-of-the-art approach.
- Evaluation on small to very large scale cloud-based infrastructures through simulation.

1.6 Outline of this dissertation

This dissertation is composed of a number of publications that were realized within the scope of this PhD. The selected publications provide an integral and consistent overview of the work performed. The different research contributions are detailed in Section 1.5 and the complete list of publications that resulted from this work is presented in Section 1.7.

Within this section we give an overview of the remainder of this dissertation and explain how the different chapters are linked together. Fig. 1.6 positions the different contributions that are presented in each chapter (Ch.) and appendix (App.). Table 1.1 shows the challenges that were studied in this dissertation and indicate which challenges were targeted per chapter.

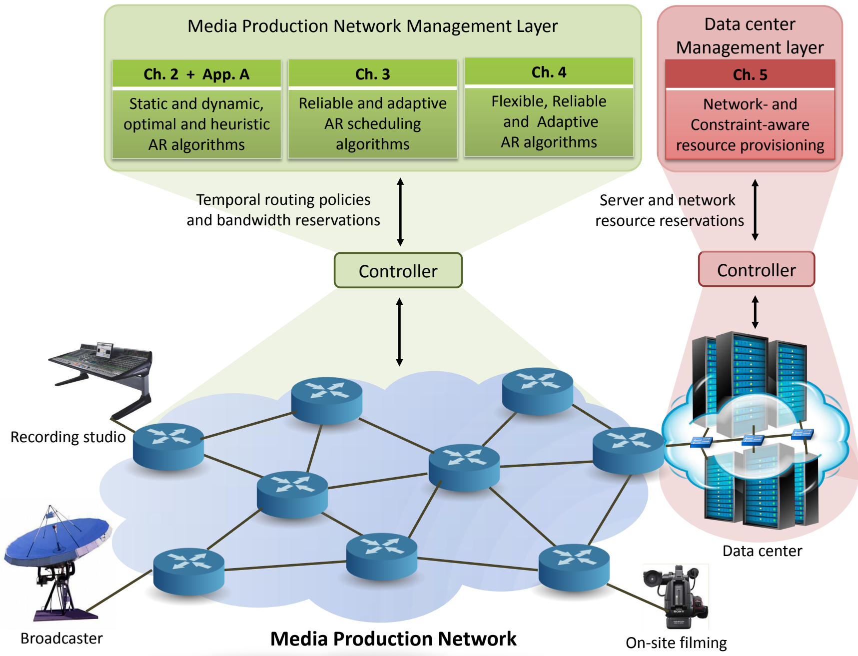


Figure 1.6: Schematic overview of the different chapters in this dissertation

Chapter 2 introduces multiple static and dynamic, optimal and near-optimal advance bandwidth reservation approaches, taking into account the specific characteristics of media production processes such as time-variable bandwidth reservation, flexible start times, request dependencies and splittable flows. In-depth performance analysis is performed to assess the viability of the proposed solutions. Appendix A also investigates and compares the impact of single-path and multi-path bandwidth allocation and routing processes of the AR solutions.

Table 1.1: An overview of the contribution characteristics per chapter in this dissertation.

	Type of solution		Type of approach		Reliability provisioning			Time domain		Routing scheme		Request dependencies	
	Static	Dynamic	Optimal	Heuristic	None	Resilient	Resilient+	Fixed	Flexible	Single-path	Multi-path	Inter-dependent	Independent
Ch. 2	•	•	•	•	•			•			•	•	•
Ch. 3		•		•		•		•			•	•	•
Ch. 4		•		•			•		•		•	•	•
Ch. 5	•		•	•	•					•		•	•
App. A	•	•	•	•	•			•		•		•	•

In Chapter 3, a resilient advance reservation approach optimized for media production networks is proposed which enables the reservation system to deliver reliable and consistent performance in the presence of failures. In order to mitigate the significant performance overheads and extra costs of provisioned redundancy, a dynamic event-driven approach has also been designed and evaluated aiming to increase network utilization and request admittance ratio. In this dynamic approach, monitoring, adaptation and re-optimization is applied at runtime. Underutilized network capacities are exploited to transfer more data than what was scheduled as long as no failure is detected.

Chapter 4, aims at investigating the possibility of using a flexible timeslot-based advance bandwidth reservation approach in media production or similar industries. A novel flexible algorithm is designed, implemented and evaluated to compare the quality and complexity of the proposed algorithm with our previously designed fixed size approach.

Chapter 5, presents optimal and near-optimal methodologies to map the application components on cloud environments. To have a scalable solution, a hierarchical near-optimal algorithm is designed and evaluated in order to be deployed in large scale cloud management systems. These algorithms take the characteristics of the underlying network into account and work with component-level applications, making a distinction between different component types.

In Chapter 6, conclusions and research perspectives are presented.

1.7 Publications

The obtained results related to this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my research.

1.7.1 Publications in international journals (listed in the Science Citation Index [1])

1. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *A Flexible, Reliable and Adaptive Timeslot-based Advance Bandwidth Reservation Mechanism for Media-centric Networks.* Submitted to International Journal of Network Management, Jun. 2017.

2. **Maryam Barshan**, Hendrik Moens, Jeroen Famaey, Filip De Turck. *Deadline-aware advance reservation scheduling algorithms for media production networks.* Published in Elsevier Journal of computer communications, 77:26–40, 2016. DOI:10.1016/j.comcom.2015.10.016

3. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *Design and Evaluation of a Dual Dynamic Adaptive Reservation approach in Media Production Networks.* Published in Elsevier Journal of Network and Computer Applications, 80:109-122, 2016. DOI: 10.1016/j.jnca.2016.12.003

4. **Maryam Barshan**, Hendrik Moens, Steven Latré, Bruno Volckaert, Filip De Turck. *Algorithms for Network-Aware Application Component Placement for Cloud Resource Allocation.* Submitted to Journal of Communications and Networks, Nov. 2016.

5. **Maryam Barshan**, Mahmood Fathy, Saleh Yousefi. *Improving the availability of P2P-based network management systems by provisioning fault tolerance property.* Published in Springer Journal of Supercomputing, 61:912-934, 2012. DOI: 10.1007/s11227-011-0659-4.

1.7.2 Publications in international conferences (listed in the Science Citation Index [2])

1. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *Design and Evaluation of a Flexible Advance Bandwidth Reservation Algorithm for Media Production Networks.* Published in proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2017, pages 142–150, Lisbon, Portugal, 2017.

[1] The publications listed are recognized as ‘A1 publications’, according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

[2] The publications listed are recognized as ‘P1 publications’, according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index - Science or Conference Proceedings Citation Index - Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

2. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *Design of a Dynamic Adaptive Reservation System in Media Production Networks.* Published in proceedings of the 8th IEEE/IFIP International Workshop on Management of the Future Internet (ManFI), 2016, pages 1149–1152, Istanbul, Turkey, 2016.

3. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert. *Dynamic Adaptive Advance Bandwidth Reservation approach in Media Production Networks.* Published in proceedings of the 2nd IEEE conference on network softwarization (Netsoft), 2016, pages 58–62, Seoul, Korea, 2016.

4. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *A Comparative Analysis of Flexible and Fixed Size Timeslots for Advance Bandwidth Reservations in Media Production Networks.* Published in proceedings of the 7th International conference on Network of the Future (NOF), 2016, pages 1–6, Buzios, Rio De Janeiro, Brazil, 2016.

5. Sahel Sahhaf, **Maryam Barshan**, Wouter Tavernier, Hendrik Moens, Didier Colle, Mario Pickavet. *A Comparative Analysis of Flexible and Fixed Size Timeslots for Advance Bandwidth Reservations in Media Production Networks.* Published in proceedings of the 12th International Conference on the Design of Reliable Communication Networks (DRCN), 2016, pages 130–137, Paris, France, 2016.

6. **Maryam Barshan**, Hendrik Moens, Jeroen Famaey, Filip De Turck. *Algorithms for advance bandwidth reservation in media production networks.* Published in proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pages 11–15, Ottawa, Canada, 2015.

7. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *Single-path versus Multi-path Advance Reservation in Media Production Networks.* Published in proceedings of the 6th International Conference On Network of the Future (NOF), 2015, pages 1–6, Montreal, Canada, 2015.

8. **Maryam Barshan**, Hendrik Moens, Steven Latré, Filip De Turck. *Algorithms for efficient data management of component-based applications in cloud environments.* Published in proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2014, pages 1–8, Krakow, Poland, 2014.

9. **Maryam Barshan**, Hendrik Moens, Filip De Turck. *Design and evaluation of a scalable hierarchical application component placement algorithm*

for cloud resource allocation. Published in proceedings of the 10th International Conference on Network and Service Management (CNSM), 2014, pages 175–180, Rio De Janeiro, Brazil, 2014.

10. **Maryam Barshan**, Mahmood Fathy, Saleh Yousefi. *Self Fault-managed and High Available P2P Architecture for Next Generation Network Management Systems.* Published in proceedings of the Second International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ), 2009, pages 63–69, Colmar, France, 2009.

1.7.3 Publications in book chapters

1. **Maryam Barshan**, Hendrik Moens, Bruno Volckaert, Filip De Turck. *Development and Validation of an Optimized Resilient Version of the Timeslot-based Advance Reservation for Media Deliver Services.* Published in proceedings of the 11th International Conference on Autonomous Infrastructure, Management and Security (AIMS), 2017, pages 72–93, Zurich, Switzerland, 2017.

2. **Maryam Barshan**, Maryam Shojaei, *Management Challenges and Solutions in Next-Generation Networks (NGN).* Published in Mobile communication and power engineering, Part of the Communications in Computer and Information Science book series, vol 296, pages 549–555, Springer, Berlin, Heidelberg, 2013.

3. **Maryam Barshan**, Mahmood Fathy, Saleh Yousefi. *Fault-Tolerant Architecture for Peer to Peer Network Management Systems.* Published in proceedings of the 9th International conference on Next Generation Wired/Wireless Networking (NEW2AN/ruSMART), St Petersburg, Russia, 2009. Part of the Lecture Notes in Computer Science book series (LNCS), vol 5764, pages 241–252, Springer, Berlin, Heidelberg.

1.7.4 Publications in other international conferences

1. **Maryam Barshan**, Mahmood Fathy, Saleh Yousefi. *A hierarchical P2P Architecture for Network Management Systems.* Published in proceedings of the International Conference on Wireless Networks (ICWN), 2009, pages 645–650, Las Vegas, Nevada, USA, 2009.

References

[1] M. Pióro and D. Medhi. *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[2] J. C. McDonald. *Fundamentals of digital switching*. Springer Science & Business Media, 2013.

[3] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. *A brief history of the Internet*. ACM SIGCOMM Computer Communication Review, 39(5):22–31, 2009.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation computer systems, 25(6):599–616, 2009.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. *A View of Cloud Computing*. Commun. ACM, 53(4):50–58, April 2010. Available from: http://doi.acm.org/10.1145/1721654.1721672, doi:10.1145/1721654.1721672.

[6] K. Nahrstedt and R. Steinmetz. *Resource management in networked multimedia systems*. Computer, 28(5):52–63, 1995.

[7] A. Neto, E. Cerqueira, M. Curado, P. Mendes, and E. Monteiro. *Scalable multimedia group communications through the over-provisioning of network resources*. In IFIP/IEEE International Conference on Management of Multimedia Networks and Services, pages 52–63. Springer, 2008.

[8] *ESnet: Energy sciences network*. http://www.es.net/. Accessed: 2017-05-1.

[9] *Internet2*. http://www.internet2.edu/. Accessed: 2017-05-1.

[10] B. Gibbard, D. Katramatos, and D. Yu. *TeraPaths: end-to-end network path QoS configuration using cross-domain reservation negotiation*. In Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on, pages 1–9. IEEE, 2006.

[11] D. Katramatos, S. Sharma, and D. Yu. *Virtual Network on Demand: Dedicating Network Resources to Distributed Scientific Workflows*. In Proceedings of the Fifth International Workshop on Data-Intensive Distributed Computing Date, DIDC '12, pages 53–62, New York, NY, USA, 2012. ACM. Available from: http://doi.acm.org/10.1145/2286996.2287006, doi:10.1145/2286996.2287006.

[12] N. Charbonneau and V. M. Vokkarane. *A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks*. Communications Surveys & Tutorials, IEEE, 14(4):1037–1064, 2012.

2

Deadline-aware Advance Reservation Scheduling Algorithms for Media Production Networks

In the media production process a substrate network can be shared by many users simultaneously when different media actors are geographically distributed.This allows sophisticated media productions involving numerous producers to be concurrently created and transferred. Due to the predictable nature of media transfers, the collaboration among different actors could be significantly improved by deploying an efficient advance reservation system. In this chapter, we introduce a model for the advance bandwidth reservation problem, which takes the specific characteristics of media production networks into account. Time variable bandwidth reservations, meeting delivery deadlines, supporting splittable flows and interdependent transfers and all types of advance reservation requests imposed by the media production transfers are incorporated into this model. In addition to the optimal scheduling algorithms, which are presented based on this model, near optimal alternatives are also proposed. In our evaluations, we observe that the proposed algorithms are scalable in terms of physical topology and granularity of time intervals and obtain a satisfactory performance, executing significantly faster than an optimal algorithm and the percentage of admitted requests remains within 8.78% of the optimal results.

⋆ ⋆ ⋆

M. Barshan, H. Moens, Jeroen Famaey and F. De Turck

Published in Elsevier Journal of Computer Communications, Vol. 77, pages 26–40, Mar. 2016.

2.1 Introduction

In the media production industry, a team of artists, editors, reporters and producers works simultaneously at geographically distributed locations producing and processing content, music, commentary, special effects, etc. Various producers and actors could then access these individual elements over a shared network to integrate them and thereby produce a complete product. In the media creation process, reliability of the transport is of crucial importance.

Predictability is a key feature of traffic in media production networks. Traffic characteristics in terms of bandwidth requirements, the time when the contents are ready, and the deadline for the data to be completely transferred to the destinations, are mostly known several hours ahead of time. The predictable nature of these transfers makes it possible to use resource reservation techniques. Therefore, a management system can efficiently manage the transmission. In general, two types of resource reservation can be distinguished [1]: Immediate Reservation (IR) and Advance Reservation (AR). While just-in-time reservation is applied in IR, the principle behind AR relies on the resource reservation times before the actual time when the resource is used. Assuming prior knowledge of the network structures and different requests, advance reservation makes it possible to schedule network requests optimally.

In computer networks, bandwidth is a valuable resource. Particularly for multimedia transfers, where large amounts of content, such as video files, have to be transmitted, efficient bandwidth management is an important factor [2]. In bandwidth-limited networks, an efficient bandwidth reservation mechanism needs to be defined to meet the QoS requirements and deadlines. The next generations of media production networks are expected to efficiently support advance reservation systems for different delivery services, so the desired QoS requirement and resource utilization could be ensured.

In this paper, we propose a set of novel AR scheduling algorithms, optimized for media production networks. Such networks impose requirements not supported by existing AR scheduling techniques. First, the start time of requests is generally flexible, the deadline is fixed, and the reserved bandwidth may vary over the lifetime of the reservation. This combination of flexible start times and elastic bandwidth allocation has not received much attention in research to date [3]. Second, in media production networks, multiple requests may depend on each other. Until now to the authors' knowledge, this aspect has remained unexplored. Third, it should be possible to split requests over multiple paths, in order to further optimize bandwidth utilization.

We propose a centralized advance reservation system which based on our evaluation scales to the size of realistic media production networks and demand patterns. We present a model to solve this variant of the AR scheduling problem, and propose various advance reservation algorithms based on our designed model. This model is an instance of time variable scheduling which is known as multicommodity over time problem. In this context, commodity corresponds to a telecommunication traffic demand between two media actors. It has been proven that the complexity of multicommodity flow over time without caching is strongly NP-hard [4]. Therefore, we also came up with efficient and near-optimal heuristic solutions which are more practical. In both optimal and heuristic approaches the main goal is threefold: 1) delivery of the requests before their deadline. 2) maximizing the number of admitted requests. 3) processing requests as quickly as possible.

Both approaches can be used in static and dynamic settings. The Static Advance Reservation Algorithm (SARA) assumes all requests are known at the start of the reservation period. By contrast, the Dynamic Advance Reservation Algorithm (DARA) supports rescheduling in order to incorporate new requests at runtime. The dynamic advance reservation system tries to admit new arrival requests while rescheduling the previously admitted requests is a must. We provide a thorough analysis of the algorithms based on in-depth simulation results. They are compared and the impact of their parameters on the solution quality is evaluated.

The remainder of this paper is organized as follows. Section 2.2, reviews the related work. Section 2.3 describes the scenarios as well as architecture and components of the proposed media production network. Section 2.4 explains the concepts, assumptions and AR scheduling formulation for media production networks. The proposed algorithms are explained in Section 2.5. A comparison of optimal and near-optimal algorithms and a performance evaluation of offline and online settings are provided in Section 2.6. Finally, Section 2.7 concludes the paper.

2.2 Related work

There has been a large number of theoretical as well as practical experimental work [5–8] related to the advance reservation problem. Here we study some of the most relevant work. The authors in [9] and [10] focus on re-routing in advance reservation networks. Our formal model is inspired by their ILP-based solutions called GILP and DILP [9]. The GILP assumes that the entire set of requests is known beforehand and the DILP is designed to work in an online setting. However our approach is different as their models assume only streaming requests with fixed time intervals and dedicated bandwidth remains fixed and equal to the demand during the entire reservation. Dependencies among the requests are also ignored. [10] is the extension of [9] in realistic multi-domain networks which addresses the implementation challenges related to advance reservation solutions.

Charbonneau et al. [3], survey the literature on advance reservation routing and scheduling algorithms, specifically focused on WDM networks. It has defined four types of advance reservation requests based on whether their start time and their duration are specified or not. All these classifications, which will be discussed in detail later, are supported in our approach. In addition, our approach is elastic which means that the allocated bandwidth is variable over time. According to this reference only two AR scheduling algorithms have been proposed that support elastic reservations [11, 12]. However, they both assume a fixed start time. Sharma et al. [13] present an algorithm called RRPC which addresses multiple flexible requests for bandwidth reservation between two end points. RRPC is deadline-aware in which any reservation that meets the deadline is acceptable. However all the requests have a same source and destination, flow splitting is not allowed and a single path is chosen for all the requests. Another work [14] focuses on dynamically transporting of large volume of data in e-science networks. The optimization consists of two steps admission control and scheduling. Periodically the central controller gathers all the new requests, runs admission control, and then schedules new and unfinished jobs.

Furthermore, the problem addressed in this work is related to the multicommodity flow problem (MCFP). The multicommodity flow problem can be described as follows: a set of individual flows have to be transferred in a dimensioned network without violating the capacity limits [15]. The resource allocation algorithm should find an optimal routing path to transfer the flows through the network. In [16], unsplittable flow and single path MCFPs are studied. Comprehensive surveys on the approaches to solve multicommodity flow problems (MCFP) and their variants are provided in [17, 18]. Our approach further deals with the problem of flow variation over time and solves an MCFP as a subproblem. In network flow problems, having variable flows over time is crucial. Dynamic flows or flow variation over time are primarily introduced by Ford and Fulkerson [19, 20]. They introduced variable flows over time as equal as static flow problem, building another temporal dimension over the network. This makes use of time-expanded networks. A time-expanded network is a copy of network in each discrete time step. Also, Fleischer et al. [21] have mentioned that in literature hardly any results on multicommodity over time are noted.

Jiancong et al. in [22] have stated that the single-path approach on which the Internet routing protocols is based, could not meet the delay requirements when the video streams are transferred over bandwidth-limited networks. They proposed a multipath routing of video contents over bandwidth limited network. However the main focus of their work is on delay and over the Internet, and therefore no reservation is considered.

Balman et al. [23] have focused on advance bandwidth reservation for on-demand data transfer in scientific applications. However, their work differs from our approach as they purely focus on data transfers, not video streaming sessions, and the routing mechanism is based on single-path in contrary to our multi-path approach. In addition, our approach considers dependencies among requests. To the best of our knowledge, dependencies among requests which is explicitly incorporated in our algorithms, have not received adequate attention in the literature by state of the art approaches.

This work is an extension of our previous work [24], in which only the static and dynamic ILP-based models are introduced. The main focus of this previous work was to investigate the viability of AR mechanisms in media production networks and to find the optimal solutions, determining the most appropriate objective function in our optimal models. We defined two objective functions and compared their performance. We found that the so called ASAP objective function, which in addition to maximizing the number of admitted requests also tries to schedule requests in earlier time slots leads to better results. In this chapter several new heuristic approaches are proposed which are near-optimal and computationally less-complex compared to ILP-based approaches. In our evaluation, their performance is compared with the highest quality optimal algorithms (i.e. algorithms based on the ASAP objective function).

2.3 Media production network architecture

The envisioned media production network is depicted in Figure 2.1. The different actors and locations involved in the media production process, such as for example recording studios, on-site filming crews, broadcasters, and storage datacenters, are connected to a shared wide-area network, consisting of interconnected switches. The network supports the exchange of raw and encoded multimedia content between an arbitrary set of actors, both in the form of file transfers and streaming. The management layer provides a reservation interface, that allows the users of the network to reserve bandwidth over certain time periods in the future. The AR scheduling algorithms are responsible for reserving the required amount of bandwidth resources for all requests. With each request, they associate one or multiple paths from source to sink with a specific amount of reserved bandwidth. In case the deadline of a transfer cannot be guaranteed, the reservation interface rejects it. When multiple transfers depend on each other, either all or none of them are admitted.

The output of the scheduling algorithms takes the form of a set of temporal routing policies (i.e., the paths associated with all requests over time) and band-

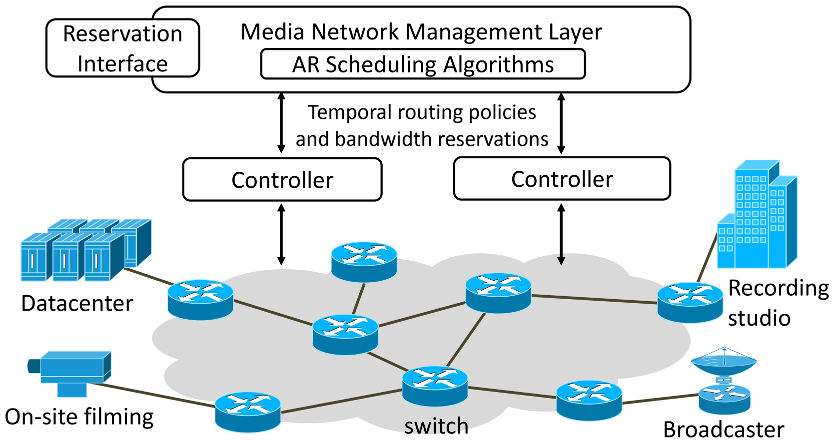


Figure 2.1: Media production network architecture and components.

width reservations (i.e., the amount of bandwidth resources to associate with each flow over time). This information can be transferred to the network controllers, that use it to configure the switches in the media production network. The controllers keep track of the temporal aspects of the policies, adjusting configurations when necessary.

In the media production industry multiple actors, which are involved in one production project, are interacting and transferring media content. If one of those transfers is not successfully done the whole project can be affected. This forms dependencies among different transfers. We refer to the set of all transfers of a project as a scenario. The scenario consists of several interdependent video transfers. We refer to each single transfer as a request. A request can have a fixed start time, end time and/or duration, or may depend on the other requests.

The video transfer types which are supported in this work are of two types which can either be video streams (VS) or file based videos (FB). We assume that for FB requests, volume and for VS requests duration is always known. As stated by [3, 25, 26] advance reservation requests are classified into four individual categories.

- STSD: Start time of the request is specified, its duration is also specified.
- STUD: Start time of the request is specified, but its duration is unspecified.
- UTSD: Start time of the request is unspecified, but its duration is specified.
- UTUD: Start time of the request is unspecified, its duration is also unspecified.

Table 2.1: Media production video request taxonomy.

Request types	Specified start time	Specified duration	Dependent VS	Dependent FB	Independent VS	Independent FB
STSD	yes	yes			X	
STUD	yes	no				X
UTSD	no	yes	X			
UTUD	no	no		X		

In this chapter all four classes are taken into account. As for VS requests the duration is always specified, if the start time of a VS request depends on other requests, this stream belongs to the UTSD class. The class of independent VS requests is STSD. In case of file transfers, the reserved resources for a request may vary over time, as long as the delivery deadline is satisfied. Moreover, even if the start time of FB request is specified, it refers to the time when the file is ready to be transferred. The time when the file transfer starts could be in future time slots. Since for file transfers duration and start time might be undefined and fluid, independent and dependent file based requests are classified as flexible UTUD and flexible STUD respectively. This is illustrated in Table 2.1.

During the scheduling process, four statuses are defined for each scenario: 1) Submitted: When a scenario entered to the system, but the admission process has not been started yet. 2) Pending: When a scenario is being processed and it is waiting for the admission decision. 3) Admitted: When all the requests of the scenario could be scheduled and transferred. This status implies the transmission guarantee. 4) Rejected: When the scheduler is not able to respond to any of the scenario requests' demands.

The remainder of this paper focuses on the AR scheduling algorithms.

2.4 Advance reservation scheduling model

We first present a formal model for the advance reservation scheduling of network bandwidth. The model can be used to schedule collections of requests, that consist of multiple interdependent and deadline-constrained network transfers. The network is represented as a graph with network nodes N and edges E.

Requests are grouped into scenarios, contained in the set S, that represent a complex workflow. These workflows must be executed in their entirety, so when a scenario is admitted, all requests must be executed. The model only admits those scenarios for which sufficient bandwidth can be guaranteed during the reservation period. When a scenario is rejected, none of its requests are executed. The various requests within a scenario may depend on each other, meaning that one request can only start when other requests have been finished.

The requests of all scenarios are stored in R. The model supports two types of network transfers: video streaming and large file transfers. Consequently R consists of both types. To make distinction between two types R_f which refers to file-based flows and R_s which refers to the streaming requests are defined.

In this model the n^{th} request is denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$ comprising of the source of the request s^n, the destination node d^n, the time when the data for file-based request is ready to transfer t_s^n (or fixed start time for video streaming request), the deadline for the transmission of the data of file-based request t_e^n (or fixed end time for video streaming request), the duration of each request i^n and finally the bandwidth demand of the request b^n. In particular, r_f^n and r_s^n refers to file-based and video streaming requests respectively. Moreover, the volume of the files are denoted by v^n and the time slot size by I. Table 2.2 lists the notations which have been used to define the model.

2.4.1 Decision variables

The goal of the model is to determine when and how requests are transferred over the network. Binary decision variables A^s and A^n are used to represent whether or not scenario s or request n are admitted. When the scenario is admitted, a collection of decision variables $\beta^{n,e,k}$ determines the amount of bandwidth for a request n that is sent over edge e during time slot k.

$$
\begin{aligned}
&A^n \in [0,1] && \forall r^n \in R \\
&A^s \in [0,1] && \forall s \in S \\
&\beta^{n,e,k} \in \mathbb{R}^+ && \forall r^n \in R, \forall e \in E, k \in [t_s^{min}, t_e^{max}]
\end{aligned}
$$

For some requests their start and end times are not specified and depend on the start or end time of other requests. In this case, the t_s^n, t_e^n or both of a request n may become decision variables of which the value is determined during the optimization process. To support these kinds of scenarios additional decision variables and constraints need to indicate whether a request is active during a given time slot. Therefore, we define the binary time slot use decision variable $SU^{n,k}$ that takes on value 0 when a request n is inactive during time slot k. These variables are defined for all requests where t_s^n, t_e^n or both are decision variables, but not for requests of which start and end time are known.

$$
\begin{aligned}
&SU^{n,k} \in [0,1] && \forall r^n \in R, k \in [t_s^{min}, t_e^{max}] \\
&t_s^n \in \mathbb{R}^+ && \forall r^n \in R \text{ if start time is variable} \\
&t_e^n \in \mathbb{R}^+ && \forall r^n \in R \text{ if end time is variable}
\end{aligned}
$$

2.4.2 Objective function

The objective function, shown in Expression 2.1, maximizes the number of admitted requests, but also tries to schedule requests as soon as possible. This is done by

Table 2.2: Symbols and notations used in the formal models.

Variable	**Description**
N	Set of network nodes.
E	Set of network links ($e \in E$).
S	Set of all scenarios ($s \in S$).
R_f	Set of file-based video requests.
r_f^n	The n^{th} request of set R_f.
R_s	Set of video streaming requests.
r_s^n	The n^{th} request of set R_s.
R	Set of all requests ($R_f \cup R_s$).
R_o	Set of all old requests.
r^n	The n^{th} request of set R, denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$.
s^n	Source node of request r^n.
d^n	Destination node of request r^n.
t_s^n	Start time for the request r^n. Decision variable when not specified.
t_e^n	Deadline for the request r^n. Decision variable when not specified.
i^n	Duration of request r^n.
b^n	Required bandwidth of r^n.
v^n	Volume of r_f^n for file-based requests (in bit).
$\beta^{n,e,k}$	Decision variable. Dedicated bandwidth over link *e*, request r^n and time interval *k*.
$SU^{n,k}$	Binary decision variable. 1 iff in time slot k any reservation is done for request n, 0 otherwise.
A^n	Binary decision variable. 1 iff request r^n is admitted, 0 otherwise.
A^s	Binary decision variable. 1 iff scenario s is admitted, 0 otherwise.
I	Duration of each time interval (in second).
t_s^{min}	Minimum start time of all reservations.
t_e^{max}	Maximum end time of all reservations.
B^e	Bandwidth capacity of link *e*.
E_v^{out}	This collection contains all edges starting from node v (egress).
E_v^{in}	This collection contains all edges ending in node v (ingress).

adding a second factor to the objective function that achieves higher values when requests are scheduled in earlier timeslots. This second term is normalized to ensure it will not interfere with the primary objective of maximizing the number of accepted requests.

$$\max \sum_{r^n \in R} A^n + \frac{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{\beta^{n,e,k}}{k}}{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{B^e}{k}} \tag{2.1}$$

2.4.3 Flow constraints

Requests are scheduled over a network, which means they are subject to capacity and network flow constraints. The capacity constraint, shown in Expression 2.2, ensures that the cumulative bandwidth reservation over each link does not exceed its bandwidth capacity. This constraint is specified for every edge, and for every time slot.

$$\sum_{r^n \in R} \beta^{n,e,k} \leq B^e \qquad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.2}$$

All network nodes that are not source or sink of a flow are subject to a flow conservation constraint, shown in Expression 2.3, to ensure the incoming flow equals outgoing flow. The network entering and leaving the source and sink of the flow is dependent on the type of request. For a file transfer request, an entire volume v^n must be transferred between the start and end times, shown in Expression 2.4. For these requests, the amount of data transferred can vary between timeslots. Video streaming requests are handled differently, as they require a constant amount of resources during all time intervals between the start and end time of the request. This is shown in Expression 2.5. To minimize the occurrence of loops within the network, constraints preventing incoming flow in the source node and outgoing flow in the sink node are added. These constraints are shown in Expressions 2.6 and 2.7.

$$\sum_{e \in E_v^{out}} \beta^{n,e,k} = \sum_{e \in E_v^{in}} \beta^{n,e,k} \tag{2.3}$$
$$\forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}], \{\forall v \in N | v \notin \{s^n, d^n\}\}$$

$$\sum_{k\in[t_s^{min},t_e^{max}]}\sum_{e\in E_{s^n}^{out}}\beta^{n,e,k}\times I = v^n \times A^n \quad \forall r_f^n \in R_f \tag{2.4}$$

$$\sum_{e\in E_{s^n}^{out}}\beta^{n,e,k} = b^n \times A^n \quad \forall r_s^n \in R_s, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.5}$$

$$\sum_{e\in E_{s^n}^{in}}\beta^{n,e,k} = 0 \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.6}$$

$$\sum_{e\in E_{d^n}^{out}}\beta^{n,e,k} = 0 \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.7}$$

2.4.4 Interdependent requests

Start and end times of requests may either be input variables or decision variables. Dependencies between different requests are handled by Expressions 2.8 up to 2.14. First, Expression 2.8 ensures either all or none of the requests of a scenario get admitted.

$$A^n = A^s \quad \forall r^n \in R \tag{2.8}$$

Expression 2.9 is defined to connect $\beta^{n,e,k}$ and $SU^{n,k}$ values, which is needed if either the start or end time of a request is a decision variable. This constraint ensures that $SU^{n,k}$ can only become zero if $\beta^{n,e,k} = 0$.

$$\beta^{n,e,k} \leq SU^{n,k} \times B^e \quad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}], \forall r^n \in R \tag{2.9}$$

If the start time is known and predefined as an input variable, then Expression 2.10 ensures that no bandwidth is dedicated to request r^n before t_s^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in [t_s^{min}, t_s^n) \tag{2.10}$$

If the start time is not specified and depends on other requests, then t_s^n is a decision variable. In that case, the constraint shown in Expression 2.11 is used to ensure $SU^{n,k}$ becomes 0 for values of $k < t_s^n$, ensuring nothing is transferred. Dependencies between time variables can then be added as shown in Expression 2.12, which ensures that the request n is started only when all the requests on which request n depends are finished.

$$t_s^n \leq k + (1 - SU^{n,k}) \times t_e^{max} \quad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.11}$$

$$t_s^n \geq t_e^{n'} + 1 \quad \{\forall r^n \in R | r^n \ depends \ on \ r^{n'}\} \tag{2.12}$$

When the end time is an input variable, then Expression 2.13 ensures that no bandwidth is dedicated to request n after t_e^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in (t_e^n, t_e^{max}] \tag{2.13}$$

If the end time is not specified, t_e^n is a decision variable. In this case, a constraint is added to ensure $SU^{n,k}$ becomes 0 for values of $k > t_e^n$, ensuring nothing is transferred after the end time. This constraint is shown in Expression 2.14.

$$t_e^n \geq k - (1 - SU^{n,k}) \times t_e^{max} \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \tag{2.14}$$

2.4.5 On-line model

The model described in the previous section can be used to statically compute a schedule for the execution of a collection of scenarios, provided all scenarios are known beforehand. In practical media production networks, the requests however arrive on-line over time. Therefore, a dynamic, on-line approach is needed that adapts the schedule at runtime. We present this on-line model as an extension of the previously discussed static model, meaning is implemented with the previously defined decision variables and objective functions.

The on-line model assumes that a previous schedule exists, and that one or more requests are added that must be scheduled. This results in a new schedule that contains both the original requests, and the new requests. We assume that a request may not be canceled once it has been accepted, meaning that while old requests may be rescheduled, they may not fail. Besides the constraints of the original model, one additional constraint (shown in Expression 2.15) is therefore added to ensure that previously admitted requests remain accepted.

$$A^s = 1 \qquad \forall r^n \in R_o \tag{2.15}$$

2.5 Advance reservation algorithms

This section, first describes the static and dynamic reservation schemes. The second part implements the models which have been defined in the previous section. In the third part the heuristic algorithms, which in general we refer to as Sequential Priority Based (SPB), are proposed to resolve the high computational complexity and scalability issue of the ILP solutions.

2.5.1 Static & dynamic reservation

The algorithms provided in this section are either static or dynamic, which can be used “offline” or “online”. The static algorithms, which we refer to as Static Advance Reservation Algorithm (SARA), can be used to generate a schedule when all requests are known in advance. However, in practice, some requests may not be known from the start of the scheduling, making it impractical to use the SARA. Therefore, a dynamic version of the resource reservation algorithm is needed.

When not all requests are known from the start, and new ones are added throughout the day, the Dynamic Advance Reservation Algorithm (DARA) can be used. When new scenarios enter to the reservation system, the DARA re-optimize the reservation by re-routing existing reservations in order to accommodate new scenarios' requests. This re-optimization is performed for the entire schedule starting from the next time slot. In DARA, we assume that new incoming scenarios have lower priority as the previous requests are already admitted and rejecting them violates the agreed SLA. Therefore, in DARA requests are divided in three categories based on their progress:

- **Scheduled:** When a request is scheduled, it will start to execute during some time slot in the future. As the request is not yet running during the trigger point, no special considerations are needed.
- **Finished:** A request is considered finished when it has finished executing at the time of the trigger point. The request itself can therefore be removed from the on-line model input. If the start or end times of other requests depend on the end time of this request, the final end time can be added as an input to the model.
- **In progress:** A request is in progress when it has started, but has not finished yet at the time of the trigger point. These requests must still be considered in the on-line model input, but the amount of data that was already transferred must be removed from the total request demand.

2.5.2 ILP based advance reservation algorithms (ILP)

In this section we define two algorithms based on the model presented in the previous section. The first algorithm is based on the static model. In an "online" setting the second approach, which makes use of the on-line version of the model, can be used.

2.5.2.1 ILP based Static Advance Reservation Algorithm ($SARA_{ILP}$)

This algorithm is based on the static formal model which assumes that all the scenario arrivals are known beforehand, which results in an optimal schedule.

Using the previously defined constraints, the multi-path model is likely to result in feasible but undesirable solution, as cycles may potentially occur in intermediate network nodes. As the model is implemented using an ILP, these cycles will never impact the optimality of the result as specified by the objective function. There are two possible approaches to address these cycles. 1) Firstly, it would be possible to modify the model by changing the objective function, adding an additional factor that minimizes the edge use. This would however increase the complexity of the model and consequently lead to an increase in execution duration.

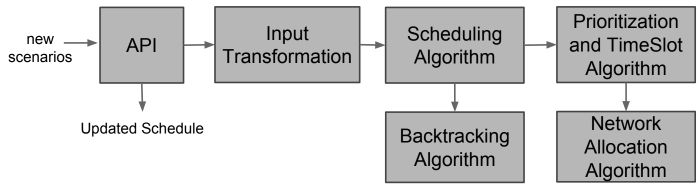


Figure 2.2: Different components of the Sequential Priority Based Advance Reservation Algorithm (SPB).

Furthermore, this would make it more difficult to balance the different optimization objectives. 2) Alternatively, the results of the algorithm can be post-processed by removing the cycles after the ILP has been solved. This approach has the advantage of limiting the complexity of the ILP model, and as stated previously has no impact on its optimality.

Because of these considerations, we use a post processing algorithm after the ILP optimization. During this post-processing phase, we look for cycles in each reserved path and remove them.

2.5.2.2 ILP-based Dynamic Advance Reservation Algorithm ($DARA_{ILP}$)

The $DARA_{ILP}$ invokes the ILP formulation of the model multiple times whenever new scenarios arrive. In this algorithm, an initial schedule is generated using the static model, which is then iteratively updated using the on-line model as new scenarios arrive. The input of the on-line model must however be modified at every trigger point to take into account the work that has already been executed. The demands of previously admitted, unfinished and in progress requests are updated based on the data that has already been transferred. Then new requests as well as modified requests are scheduled together.

2.5.3 Sequential Priority Based advance reservation algorithms (SPB)

In general, the ILP-based algorithms have a high computational overhead, particularly with fine-grained time slot sizes and large physical networks. The Sequential Priority Based (SPB) advance reservation algorithm is a heuristic approach which is proposed to resolve the scalability limitations of the ILP solutions. The admission control process is also integrated into the algorithm and once a scenario is admitted, it will never be denied by the scheduler in future. In this approach, the scenarios are sequentially admitted and scheduled.

Different components of the SPB advance reservation algorithm are illustrated in Figure 2.2. As can be observed from this figure, the new scenarios enter to the reservation system through an API (Application Programming Interface). Then any transformation can be applied. For example in dynamic approach before the scheduling algorithm invocation, the previously admitted scenarios' demand needs to be updated. In the next step the scenario requests are prioritized and in each time slot the algorithm sequentially calls the network allocation algorithm for each scenario request. If this process is successfully terminated the new scenario is admitted, and the schedule is updated. Otherwise, the previous scheduling and network state remain untouched and the scenario is rejected. Again we discern two algorithm variants: $SARA_{SPB}$ and $DARA_{SPB}$.

2.5.3.1 Sequential Priority Based Static Advance Reservation Algorithm ($SARA_{SPB}$)

In this algorithm, first the scenarios are sorted and then sequentially processed. This sorting is based on the earliest average start time of the scenario's requests. If two scenarios have the same value, the one requiring more resources is chosen. As can be seen in Algorithm 1, the network resource usage, the requests information and the current scheduling are saved for each scenario.

algorithm 1: The SARA Sequential Priority Based (SPB) algorithm.

input: scenarios' requests, network infrastructure
sortedQueue ← AverageStartSort(all scenarios);
for *(scenario ∈ sortedQueue)* **do**
    Set scenario status as Pending;
    currentstate ← Save the current system state;
    Prioritization(scenario's requests);
    sysReqList.Add(scenario's requests);
    feasible ← TimeSlot(sysReqList);
    if *(feasible)* **then**
        Update the scheduling;
        Set scenario status as Admitted;
    else
        Set current system state to CurrentState;
        Set scenario status as Rejected;
    end
end

Then each scenario in the sorted list is processed as follows. The prioritization algorithm is another component which assigns priorities to the scenario's requests. In the prioritization step two factors play a role: the estimated hard deadline and the volume. Since the deadline may not be specified for all the requests, the hard deadline (i.e. the latest possible deadline) for those with no specific deadline should be estimated. This time is calculated by assuming that all requests on which the request depends use the entire network at once. This gives the latest possible deadline for the request. In the prioritization algorithm, the main factor is the estimated hard deadline: the sooner the deadline, the higher the priority. The second factor, volume, comes into consideration only when the hard deadlines are equal, the higher the demand, the higher the priority.

Then the scenario's requests are added to the list of system requests and this list is given to the TimeSlot algorithm. Based on the result of TimeSlot algorithm, SPB decides to admit or reject the scenario. If the TimeSlot algorithm achieves a feasible schedule, the previous reschedule is updated, otherwise the algorithm has to backtrack to the previous feasible situation.

The TimeSlot algorithm, which is shown in Algorithm 2, iterates over the time slots and consists of 5 sub-algorithms for each time interval.

1. TimeSlotRequests: First, the algorithm has to determine which unserved requests can be served in the current time slot. For independent requests the algorithm looks at the start time. If the current interval is greater or equal the request start time, these requests are eligible to be added to the list of current requests. For requests with start time dependencies, the algorithm checks whether the requests on which this request depends are finished or not. If all the requests on which the request depends are fulfilled, this request can be started. Then it will be added to the list of current requests.

2. Limit: In order to avoid the extra reservation for the requests, a limitation needs to be defined for each request. The limit for the video streams is their required demand, because their demand is fixed and non-variable. The limit of file-based requests is their residual demand which is modified whenever a part of video file is transferred.

3. Sorting: In this step the requests of different scenarios selected in the previous step are sorted based on their priorities.

4. BWallocation: The most important sub-algorithm in TimeSlot is the bandwidth allocation algorithm. We have defined two different bandwidth allocation algorithms for video streams and video files. These algorithms are presented later in detail.

5. UpdateAndCheckFeasibility: based on the provided result of the previous step, BWallocation, and by calculating the residual demands, the requests requirements are updated and the feasibility of the results is checked. If the hard deadline of a request is reached, but part of the request has not been transferred yet and the residual demand is not zero, the hard deadline has not been met. In this step rescheduling is infeasible and the TimeSlot algorithm returns false.

algorithm 2: The TimeSlot algorithm.

```
input: sysReqList, timeSlotGraphs
for (t ∈ Time Intervals) do
    currentRequests←TimeSlotRequests(t,sysReqList);
    if (currentRequests ≠∅) then
        Limit(currentRequests);
        sortedList ← Sort(currentRequests);
        reservation ← BWallocation(sortedList);
        if (!UpdateAndCheckFeasibility(reservation)) then
            return false;
        end
    end
end
return true;
```

algorithm 3: The BWallocation algorithm.

```
input: sortedReqList
costAllocation(Links);
for (req ∈ sortedReqList) do
    if (req is FB) then
        reservation ← BWallocationFB(req);
    else
        reservation ← BWallocationVS(req);
    end
end
return reservation;
```

The Bwallocation algorithm, which is shown in Algorithm 3, determines the maximum possible bitrate and associated paths for the requests. This algorithm first assigns cost to the network links using the CostAllocation algorithm. There are several approaches to find a path between source and destination of a flow. Since in advance reservation the other flows and all their specifications are often known, the most logical way is to take their preferred deployment into account. As a result in this step the algorithm tries to find the most desired paths and give them the highest cost. The cost of each link is the sum of desirability of this link for all requests. To find how important a link is for a request, the maximum flow (maxflow) from source to sink of each request is determined. Having the maxflow and the utilization per link, the desirability of the link is measured by dividing the link utilization by the maxflow. The maxflow is measured using the Edmonds-Karp maximum flow approach [27].

Then according to the type of the request either BWallocationFB or BWallocationVS algorithm, shown in Algorithm 4 and Algorithm 5 respectively, is invoked.

The BWallocationFB algorithm is in charge of the FB requests and is based on maxflow and least-cost path algorithms. In order to transfer the files as soon as possible, first the maxflow is applied based on the Edmonds-Karp algorithm. If the maxflow is lower than the request limit, all the maxflow paths are reserved for this request. Otherwise, the algorithm forms a graph out of the maxflow paths and the k-shortest path is the second alternative. In this step finding the least-cost path is repeated till the total bandwidth offered by the paths is sufficient for the request. The shortest path applied here is a modified version of the Dijkstra shortest path algorithm [28] in which the cost of the links are taken into account instead of path length.

algorithm 4: The BWallocationFB algorithm for file based requests.

```
input: an FB request
maxFlow ← EdmondsKarp.getMaxFlow(graph);
if (maxFlow = 0) then
|   Return;
else if (maxFlow ≤ Limit(req)) then
|   reservation(req, maxFlowPath);
else
|   graph(maxFlowPath);
|   path ← LeastCostPath(graph);
|   while (path ≠ ∅) do
|   |   minBW ← minBandwidth(path);
|   |   flow ← flow+minBW;
|   |   if (flow ≥ Limit(req)) then
|   |   |   minBW ← minBW−(flow−Limit(req));
|   |   |   reservation(req, path, minBW);
|   |   |   update the residual graph(minBW, path);
|   |   |   feasibility ← true;
|   |   |   return reservation;
|   |   else
|   |   |   reservation(req, path, minBW);
|   |   |   update the residual graph(minBW, path);
|   |   end
|   |   path ← LeastCostPath(graph);
|   end
end
```

The BWallocationVS algorithm deals with video stream requests. This algorithm is partially similar to the second part of the BWallocationFB algorithm where the maxFlow is higher than the request limit. The algorithm iteratively looks for the least cost path on the whole graph and sums up the minimum available bandwidth of the paths. In each step the network capacities are modified and the path is reserved. These steps are repeated over the residual graph while the total bandwidth provided by the paths fulfills the request demand. If no more paths are left, the request could not be served and feasibility of rescheduling is false.

algorithm 5: The BWallocationVS algorithm for video streaming requests.

input: a VS request
path ← LeastCostPath(graph); **while** *(path ≠ ∅)* **do**
    minBW ← minBandwidth(path);
    flow ← flow+minBW;
    if *(flow ≥ Limit(req))* **then**
        minBW ← minBW−(flow−Limit(req));
        reservation(req, path, minBW);
        update the residual graph(minBW, path);
        feasibility ← true;
        return reservation;
    else
        reservation(req, path, minBW);
        update the residual graph(minBW, path);
    end
    path ← LeastCostPath(graph);
end
feasibility ← false;

2.5.3.2 Sequential Priority Based Dynamic Advance Reservation Algorithm ($DARA_{SPB}$)

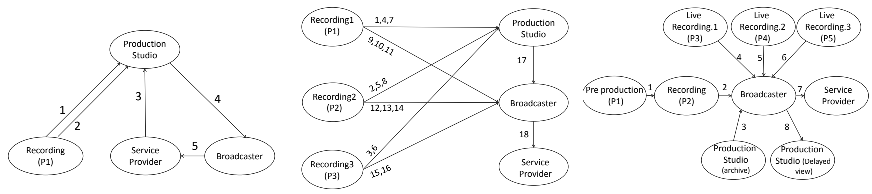
The $DARA_{SPB}$ is invoked several times whenever new scenarios are submitted to the reservation system. The dynamic approach is partly similar to the $SARA_{SPB}$. However in addition to the static approach and same as dynamic ILP-based algorithm, it has to update the previously admitted requests' demands based on whether the request is scheduled, is finished or is in progress.

2.6 Experimental Results

In this section, we first evaluate the SPB static and dynamic algorithms by comparing their performance with the optimal results provided by the ILP models as a benchmark. Then, we make an extensive evaluation on the SPB algorithms, determining the influence of the available bandwidth, the percentage of requests known in advance, the number of scenarios, and the time granularity.

2.6.1 Evaluation Setup

Based on interviews with several Belgian media production actors, including a broadcaster, service provider, and recording facility provider, a set of use case sce-



(a) Use case 1: Soccer discussion program (b) Use case 2: Infotainment show (c) Use case 3: News broadcast

Figure 2.3: Interactions between media production actors in the three considered use case scenarios.

narios was defined that serve as a basis for the evaluation. Figure 2.3 depicts the interactions between actors in the three defined use cases. Use case 1 represents a soccer after-game discussion program and comprises 5 different file transfer requests. Use case 2 is a 30 minute infotainment show and consists of 18 file transfer requests. Finally, use case 3 is a news broadcast, consisting of 4 file transfer and 4 video streaming requests. Several instances of each use case are generated, based on randomized input parameters. A detailed overview of the randomized variables of each use case and its requests is shown in Table 2.3. The variable names used in the table header are defined in Table 2.2. $\#t_s^n dep.$ refers to the number of requests on which the start time of the request (i.e., t_s^n) depends. If a request does not depend on others, $t_s^n/dep.On$ is defined as the start time of the request, otherwise it points to those interdependent requests. The variables $\#t_s^e dep.$ and $t_s^e/dep.On$ are similarly defined for the end time of the request. The variable s used in the table represents the earliest time on which the file-based request could be started. In addition, st, d, and et deal with the streaming requests and refer to the start time of the broadcast on television, the deadline of the request to get started, and the end time of the request respectively

Because of the limited scalability of ILP-based algorithms, two different topologies are used for the evaluation of media production network reservation system. The smaller size, depicted in Figure 2.4, consists of media production actor sites, switches and bidirectional WAN links. This topology contains 12 nodes, 8 of which are devoted to different media production actors e.g. the production studio, broadcaster, service provider and recording locations. The 4 remaining nodes are the intermediate switches, connected in a full mesh topology. The larger test topology, shown in Figure 2.5, is only used to evaluate the performance of the more scalable SPB algorithms. This network is the well-known ATT North America topology which consists of 25 nodes and 56 links [29].

Each simulation run covers a 24 hour period. When using the SARA algorithm,

Table 2.3: Details of the use case requests

Use case 1	$Type$	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1	r_f	$P1$	Production studio	0	$rand(s+1hrs, s+5hrs)$	1	$Req3$	$90min$	$200Mbps$
Req2	r_f	$P1$	Production studio	0	$rand(s, s+6hrs)$	1	$Req3$	$90min$	$200Mbps$
Req3	r_f	Broadcaster	Production studio	2	$Req1,2$	1	$Req4$	$90min$	$200Mbps$
Req4	r_f	Production studio	Broadcaster	3	$Req1,2,3$	0	st	$180min$	$15Mbps$
Req5	r_f	Broadcaster	Service provider	0	$st+3hrs$	0	$24hrs$	$180min$	$15Mbps$
$P1 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand(1, 9) hrs; $st =$ rand(17, 19) hrs									
Use case 2	$Type$	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1,9	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req2,10	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req3,11	r_f	$P3$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req4,12	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req5,13	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req6,14	r_f	$P3$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req7,15	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req8,16	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req17	r_f	Production studio	Broadcaster	16	$Req1..16$	1	$Req18$	$60min$	$200Mbps$
Req18	r_f	Broadcaster	Service provider	1	$Req17$	0	st	$60min$	$15Mbps$
$P1, P2, P3 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand(1, 15) hrs; $st =$ rand(18, 22) hrs									
Use case 3	$Type$	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1	r_f	$P1$	$P2$	0	$rand(s, 9hrs)$	1	$Req2$	$(30-50)min$	$200Mbps$
Req2	r_f	$P2$	Broadcaster	1	$Req1$	0	$rand(10,12)hrs$	$(30-50)min$	$200Mbps$
Req3	r_f	Production studio	Broadcaster	0	$rand(s, 9hrs)$	0	$rand(10,12)hrs$	$(30-50)min$	$200Mbps$
Req4	r_s	$P3$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req5	r_s	$P4$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req6	r_s	$P5$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req7	r_s	Broadcaster	Service provider	0	st	0	st+0.5hrs	$30min$	$15Mbps$
Req8	r_f	Broadcaster	Production studio	0	st+0.5hrs	0	$24hrs$	$30min$	$15Mbps$
$P1, P2, P3, P4, P5 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand(1, 7) hrs; $st =$ rand(12, 16) hrs; $d = (st + 0.5 - i^n)$ hrs; if $(i^n < I)$ then $(et = T_s^n+1)$ else $(et = T_s^n + i^n)$									

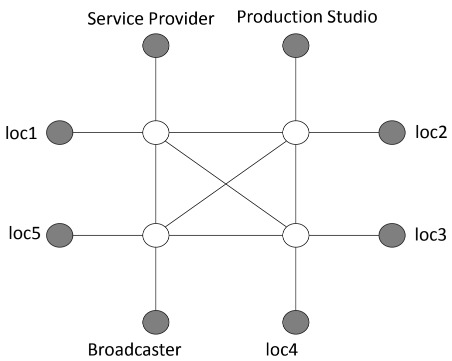


Figure 2.4: The smaller Media production network topology used in the evaluation.

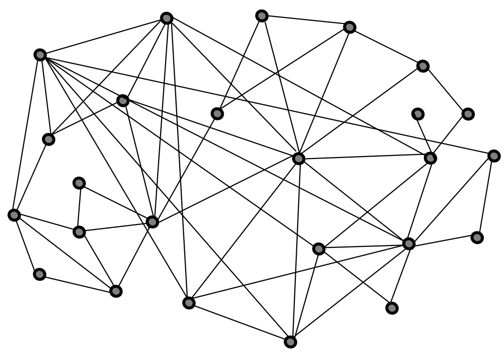


Figure 2.5: The Larger Media production network topology used in the evaluation. The ATT North America topology consists of 25 nodes and 56 links.

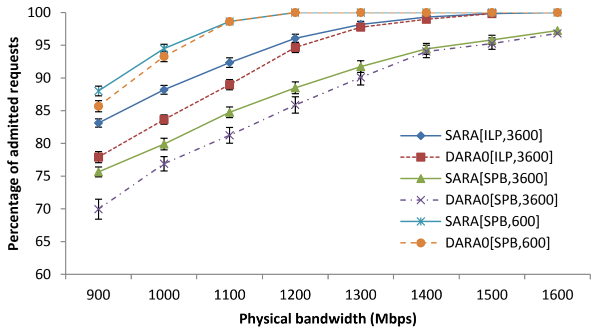


Figure 2.6: Impact of bandwidth capacity and percentage of known requests on admission rate.

it is assumed that all scenarios are known in advance. When using DARA some use case instances are assumed to be known only throughout the day, at least one hour before t_s^n of its earliest request. Throughout this section, DARA*XX*%[*YY*, *ZZ*] denotes that *XX*% of the use case instances are known at the start of the simulated day and the algorithm *YY* is used (i.e., ILP or SPB). *ZZ* is optional and refers to the time slot size in seconds, if the *ZZ* is not specified, the default time slot size of 3600 seconds has been used. Furthermore, the number of use case instances equals 20 (in total 209 requests) and 50 (in total 519 requests) for all the experiments with the smaller and larger topologies respectively. All results are averaged over 50 runs with different randomized inputs. Error bars denote the standard error.

All algorithms in this section are implemented in Java 7. ILP models are solved using the IBM ILOG CPLEX 12.6 optimization software package.

2.6.2 Comparing the SPB algorithms to the ILP-based algorithms

Figure 2.6 compares the ILP solutions with the SPB algorithms, where for the latter either 1 hour or 10 minute time slot sizes are shown. In this evaluation, all or none of the requests are known in advance. First for the same time slot size of 1 hour, this figure shows that the result of the SPB algorithms is close to the optimal values. The performance of $SARA_{SPB}$ and $DARA_{SPB}$ are within 8.29% of $SARA_{ILP}$ and 8.78% of $DARA_{ILP}$ respectively. For a time slot size of 1 hour, the SPB algorithms reach to 100% admittance only when 2300 Mbps physical bandwidth is available. For the 6-times shorter time interval size (i.e. 10 minutes) and 1200 Mbps capacity, both the static and dynamic SPB algorithms admit all 20 scenarios, whereas the static ILP-based algorithm only achieve 96.05% on average.

Figure 2.7 and Figure 2.8 are provided to compare the performance of ILP-

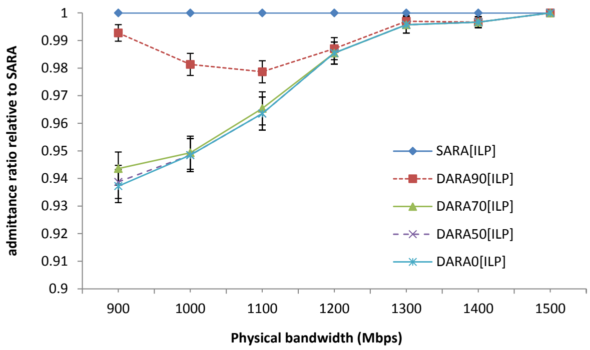


Figure 2.7: Impact of percentage of known requests on admission rate in ILP-based algorithms.

based and SPB algorithms respectively, showing the influence of the percentage of known requests in advance on the solutions. Knowing more requests gives more freedom to schedule them and makes it easier to determine the subset of requests to reject. Thus, static algorithms outperform the dynamic ones. Therefore, to have a clear distinction, in both figures the result are shown relative to the $SARA$. As expected, more known requests significantly increase the performance. The results show that when requests are not known at the start of the day, $SARA_{ILP}$ outperforms $DARA_{ILP}$ by up to 5.22% while when 90% are known this is reduced to 1.97% at most. For the SPB algorithms the same trend can be observed. The percentage of admitted requests in $DARA_{SPB}$ is within 5.7% and 1.37% of $SARA_{SPB}$ for the 0% and 90% of known requests respectively.

Furthermore, to have a comparison of the execution times Figure 2.9 is depicted. This figure compares the execution duration of the static ILP and SPB algorithms. As can be observed from this figure, the ILP is the most complex and the slowest, and the static SPB algorithm with the same time interval size is between 128 up to 520 times faster than the ILP solution.

2.6.3 Evaluation of the SPB algorithms

2.6.3.1 Impact of available bandwidth

We now assess the impact of physical network capacity on the SPB algorithm performance. Figure 2.10 compares the percentage of admitted requests of SARA to DARA, where for the latter various ranges of use case instances are known in advance. The network capacities vary from 600 up to 2300 Mbps and a time slot size of 1 hour is used. This figure shows that when no request is known in advance, SARA shows up to 5.7% higher acceptance rate compared with the dynamic approach.

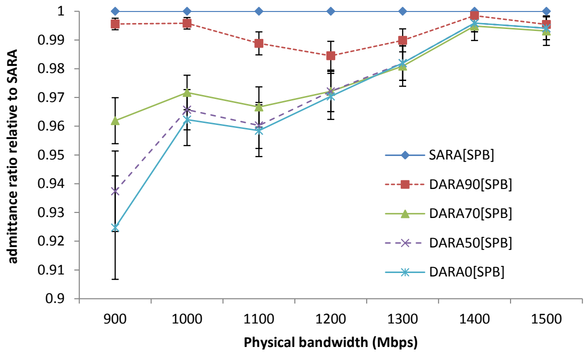


Figure 2.8: Impact of percentage of known requests on admission rate in SPB algorithms.

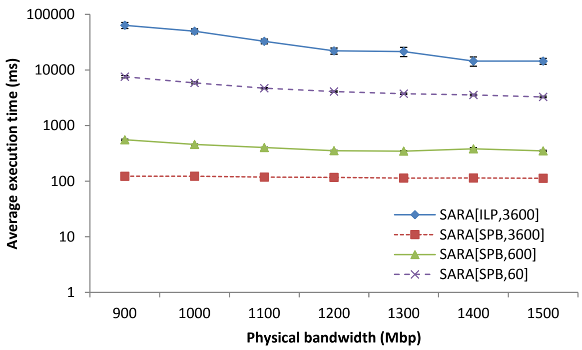


Figure 2.9: The execution time of SPB and ILP-based approaches.

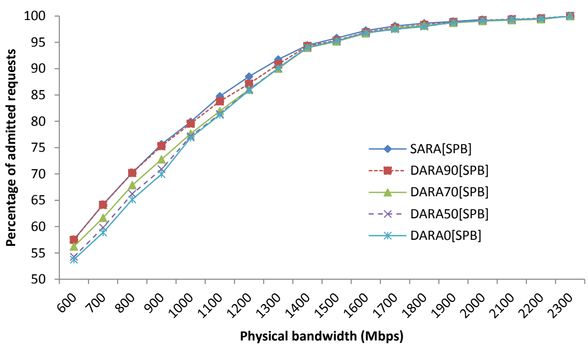


Figure 2.10: Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 12-node topology.

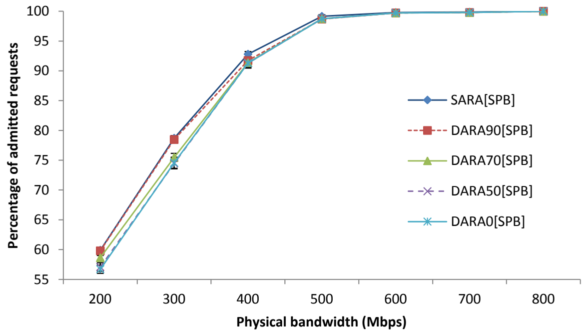


Figure 2.11: Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 25-node topology.

In Figure 2.11 the same results are shown for the larger infrastructure. The available bandwidth varies from 200 up to 800 Mbps. As can be observed from this figure, when none of requests are known beforehand, the admittance ratio in $DARA$ is within 4.1% of $SARA$ and when 90% of requests are known the result is within 1.1%.

2.6.3.2 Impact of time slot granularity

Figures 2.12 and 2.13 study the impact of time slot granularity on SARA and DARA for the 12-node and 25-node topology respectively. For the smaller network the link capacity of 700 Mbps and for the larger topology the link capacity of 200 Mbps is used. As shown in both figures, the fine-grained experiment with shortest time slot size results in the best performance. However, although more granularity increases the performance of the algorithm, the complexity of the algorithm significantly increases as well. In the case of DARA, the algorithm is invoked several times throughout the day.

In Figure 2.12, we observe that an interval size of 60 seconds yields 11.5% better results than a size of 3600 seconds for SARA. However, the execution time of the algorithm also increases. For DARA, a similar trend is seen. Moreover, it should be noted that in case fewer requests are known in advance, the complexity of a single DARA algorithm invocation decreases significantly. This can be seen in Figures 2.14 and 2.15 for the smaller and the larger topologies respectively. For example in Figure 2.14, when none of requests are known and the size of time slot is 1 minute, the execution time is on average 18.33 times shorter than when all requests are known in advance. In the former case, the algorithm needs to be executed 19.6 times on average, while in the latter only once. Based on the results, a time slot size of 600 seconds optimises the trade-off between optimality and complexity. This interval size is not the most optimal value for all possible

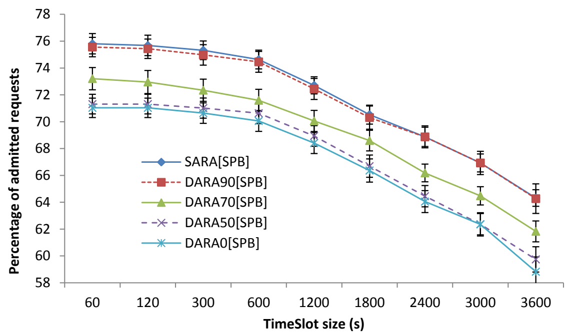


Figure 2.12: Impact of timeslot granularity on request admission rate for 12-node topology.

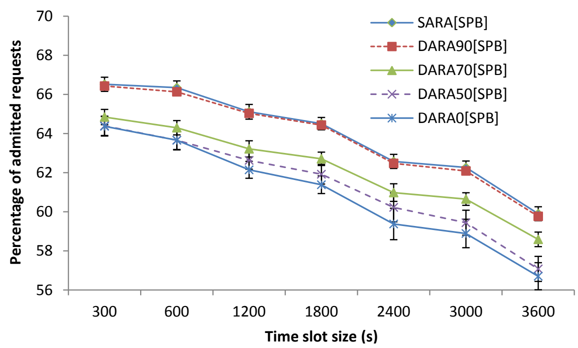


Figure 2.13: Impact of timeslot granularity on request admission rate for 25-node topology.

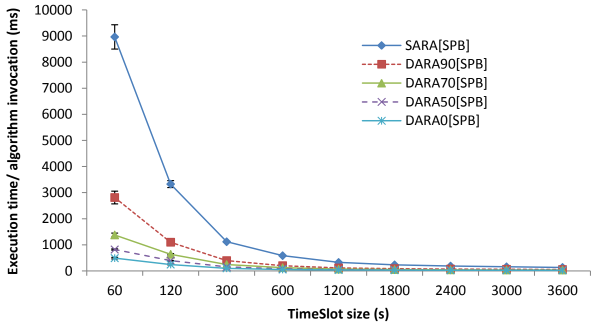


Figure 2.14: The average execution times of different time slot sizes and percentage of known requests for 12-node topology.

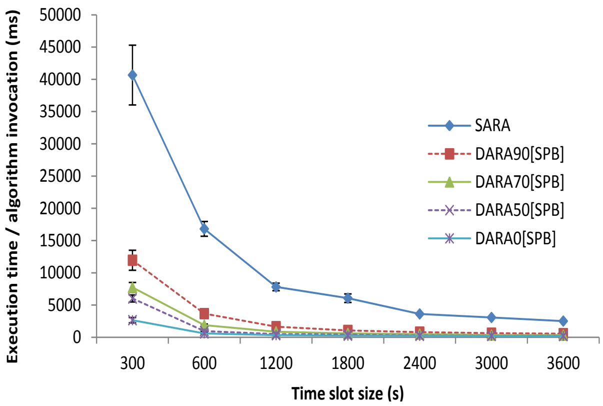


Figure 2.15: The average execution times of different time slot sizes and percentage of known requests for 25-node topology.

configurations, but is within 1.6% of the optimum in all evaluated cases.

2.6.3.3 Impact of network load

Figures 2.16, 2.17 and 2.18 compare the influence of number of use case instances and percentage of known requests. In Figures 2.16 and 2.17 the smaller topology with network capacity of 600 Mbps and 1 hour time slot size are used. The number of scenarios varies from 1 to 20. As can be seen in Figure 2.16 by increasing the demands, the percentage of admitted requests decreases. $SARA$ outperforms $DARA$ up to 5% when 0% of requests are known in advance, and up to 3.8% when 50% are known. As depicted in Figure 2.18 the same trend can be observed for the larger topology with 200 Mbps capacity. $SARA$ yields up to 5.8% better results than $DARA$.

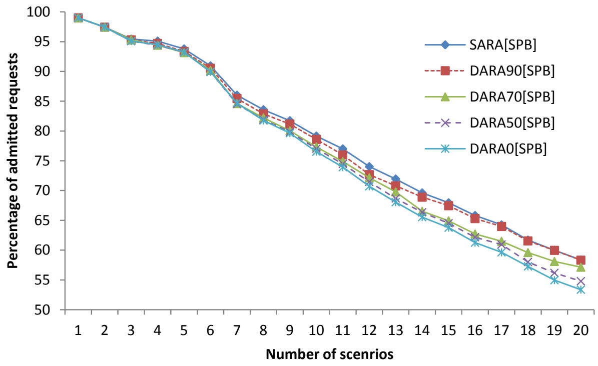


Figure 2.16: Impact of network load on request admission rate for 12-node topology.

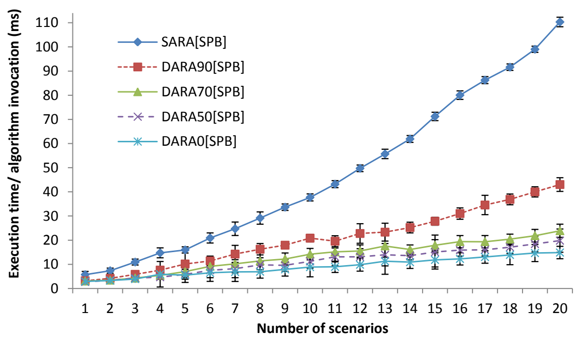


Figure 2.17: The execution times of different number of scenarios and percentage of known requests for 12-node topology.

In Figure 2.17 the execution times for each algorithm invocation for a range of percentage of known requests is assessed. Based on this result the DARA is up to 7.4 times faster, when the number of scenarios is 20. However, the algorithm is invoked 9.48 times on average.

2.7 Conclusion

In this chapter, an optimal model and a set of novel scheduling algorithms were presented for advance bandwidth reservation in media production networks. Specifically the SPB approach is proposed to resolve the computational complexity associated with the optimal solutions. The bandwidth scheduling algorithms take the specific characteristics of media production processes into account, for example time-variable bandwidth reservation, flexible start times, request dependencies

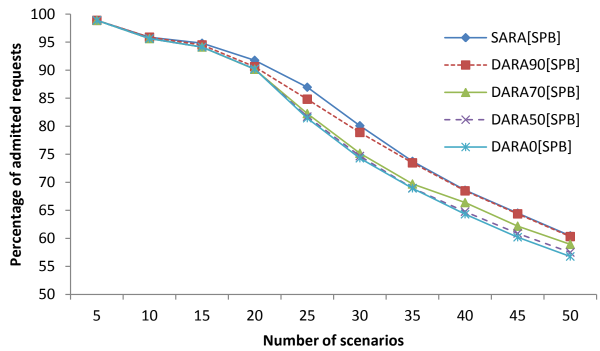


Figure 2.18: Impact of network load on request admission rate for 25-node topology.

and splittable flows. In our approaches all four types of advance reservation requests are supported. Furthermore, the proposed algorithms operate in both offline and online manners. A detailed performance analysis is conducted to assess the viability of ILP-based and SPB solutions. The influence of the available bandwidth, the percentage of requests known in advance, the network load, the time granularity and the execution time have been evaluated.

Our evaluation showed that the SPB results at least within 8.78% of the optimal admittance rate. Also, when a significant portion of requests is known in advance, AR significantly increases bandwidth efficiency and request admittance. Concretely, in case all requests are known beforehand, request admittance of the optimal and heuristic solutions can be increased up to 5.22% and 5.7% respectively. Additionally, the results showed that time granularity increases algorithm accuracy and optimality in terms of request admittance. SPB can achieve higher scalability in terms of the size of physical network as well as time slot sizes. The size of time intervals can be fine-grained up to 1 minute. Comparing to the ILP-based approaches, the SPB algorithms offer lower operational overhead in terms of problem complexity and execution time.

Future work includes determining the impact on quality and performance of variable time intervals, and adding resilience to improve the robustness of the schedules generated by the advance reservation system.

Acknowledgment

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government – department EWI. The research leading to these results was performed within the context of ICON MECaNO. It is a project co-funded by iMinds, a digital research institute founded by the Flemish Government. Project partners are SDNsquare, Limecraft, VideoHouse, Alcatel-Lucent, and VRT, with project support from IWT under grant agreement no. 130646.

References

[1] E. M. Varvarigos, V. Sourlas, and K. Christodoulopoulos. *Routing and scheduling connections in networks that support advance reservations*. Computer Networks, 52(15):2988–3006, 2008.

[2] K. Nahrstedt and R. Steinmetz. *Resource management in networked multimedia systems*. Computer, 28(5):52–63, 1995.

[3] N. Charbonneau and V. M. Vokkarane. *A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks*. Communications Surveys & Tutorials, IEEE, 14(4):1037–1064, 2012.

[4] A. Hall, S. Hippler, and M. Skutella. *Multicommodity Flows over Time: Efficient Algorithms and Complexity*. In J. Baeten, J. Lenstra, J. Parrow, and G. Woeginger, editors, Automata, Languages and Programming, volume 2719 of *Lecture Notes in Computer Science*, pages 397–409. Springer Berlin Heidelberg, 2003. Available from: http://dx.doi.org/10.1007/3-540-45061-0_33, doi:10.1007/3-540-45061-0_33.

[5] C. Guok, E. N. Engineer, and D. Robertson. *ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)*. Internet2 Joint, 2006.

[6] B. Gibbard, D. Katramatos, and D. Yu. *TeraPaths: end-to-end network path QoS configuration using cross-domain reservation negotiation*. In Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on, pages 1–9. IEEE, 2006.

[7] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. *StorNet: Integrated Dynamic Storage and Network Resource Provisioning and Management for Automated Data Transfers*. In Journal of Physics: Conference Series, volume 331, page 012002. IOP Publishing, 2011.

[8] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. *StorNet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers*. In Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference on, pages 121–126. IEEE, 2011.

[9] C. Xie, H. Alazemi, and N. Ghani. *Rerouting in advance reservation networks*. Computer Communications, 35(12):1411–1421, 2012.

[10] H. Alazemi, F. Xu, C. Xie, and N. Ghani. *Advance reservation in distributed multi-domain networks*. IEEE Systems Journal, 2013.

[11] S. Naikstam and S. Figueira. *Elastic reservations for efficient bandwidth utilization in LambdaGrids*. Future Generation Computer Systems, 23(1):1–22, 2007.

[12] L.-O. Burchard, H.-U. Heiss, and C. De Rose. *Performance issues of bandwidth reservations for Grid computing*. In Symposium on Computer Architecture and High Performance Computing, pages 82–90, 2003.

[13] S. Sharma, D. Katramatos, D. Yu, and L. Shi. *Design and Implementation of an Intelligent End-to-end Network QoS System*. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pages 68:1–68:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press. Available from: http://dl.acm.org/citation.cfm?id=2388996.2389089.

[14] K. Rajah, S. Ranka, and Y. Xia. *Advance Reservations and Scheduling for Bulk Transfers in Research Networks*. IEEE Trans. Parallel Distrib. Syst., 20(11):1682–1697, November 2009. Available from: http://dx.doi.org/10.1109/TPDS.2008.250, doi:10.1109/TPDS.2008.250.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Technical report, DTIC Document, 1988.

[16] H. Masri, S. Krichen, and A. Guitouni. *A multi-start variable neighborhood search for solving the single path multicommodity flow problem*. Applied Mathematics and Computation, 251:132–142, 2015.

[17] A. Ouorou, P. Mahey, and J.-P. Vial. *A survey of algorithms for convex multicommodity flow problems*. Management science, 46(1):126–147, 2000.

[18] J. L. Kennington. *A survey of linear cost multicommodity network flows*. Operations Research, 26(2):209–236, 1978.

[19] L. R. Ford Jr and D. R. Fulkerson. *Constructing maximal dynamic flows from static flows*. Operations research, 6(3):419–433, 1958.

[20] L. Ford and D. R. Fulkerson. *Flows in networks*, volume 1962. Princeton Princeton University Press, 1962.

[21] L. Fleischer and M. Skutella. *Quickest flows over time*. SIAM Journal on Computing, 36(6):1600–1630, 2007.

[22] J. Chen, S.-H. Chan, and V. O. Li. *Multipath routing for video delivery over bandwidth-limited networks*. Selected Areas in Communications, IEEE Journal on, 22(10):1920–1932, 2004.

[23] M. Balman, E. Chaniotakis, A. Shoshani, and A. Sim. *A flexible reservation algorithm for advance network provisioning*. In High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for, pages 1–11. IEEE, 2010.

[24] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Algorithms for Advance Bandwidth Reservation in Media Production Networks*. In the integrated network management. IM2015. IFIP/IEEE International symposium on. IEEE, 2015. to appear.

[25] J. Zheng and H. T. Mouftah. *Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks*. In Communications, 2002. ICC 2002. IEEE International Conference on, volume 5, pages 2722–2726. IEEE, 2002.

[26] E. He, X. Wang, V. Vishwanath, and J. Leigh. *CAM03-6: AR-PIN/PDC: Flexible Advance Reservation of Intradomain and Interdomain Lightpaths*. In Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE, pages 1–6. IEEE, 2006.

[27] J. Edmonds and R. M. Karp. *Theoretical improvements in algorithmic efficiency for network flow problems*. Journal of the ACM (JACM), 19(2):248–264, 1972.

[28] T. Cormen. *Introduction to Algorithms*. MIT Press, 2009. Available from: http://books.google.be/books?id=Jwr8jwEACAAJ.

[29] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. *The internet topology zoo*. Selected Areas in Communications, IEEE Journal on, 29(9):1765–1775, 2011.

3

Design and Evaluation of a Dual Dynamic Adaptive Reservation Approach in Media Production Networks

In the previous chapter, we exploited the predictability of media transfer and use advance bandwidth reservation services to achieve greater bandwidth utilization and service guarantees. To offer reliable reservations, the incorporation of fault-tolerance related features in bandwidth reservation strategies is a necessity, although this imposes a waste of capacity and extra performance overhead. In this chapter, we first introduce a resilient bandwidth reservation mechanism, to ensure that the reservations remain valid when the system is in operation. To maximally utilize the network, and to ensure there is a quick response in a dynamic network environment, constant monitoring and optimization is needed. Therefore, we then propose an efficient complementary optimization approach, referred to as Runtime Adaptation (RA) approach, which can be used in combination with the resilient solution. When a schedule is produced by a resilient advance reservation algorithm, the generated schedule is continually updated over time using the RA approach in order to be capable of dynamically adapting the network to changing conditions and mitigating the side effects of provisioned reliability. This step uses the interconnecting network links' leftover capacity, resulting in an increased performance both in steady and unsteady network conditions. Our evaluations show that in *failure-prone environments, the combination of resilient advance bandwidth reservation and proposed approach leads to significant increase in the success rate of admitted requests, up to 6.77 times, compared to the resilient advance reservation algorithms.*

⋆ ⋆ ⋆

M. Barshan, H. Moens, B. Volckaert and F. De Turck

Published in Journal of Network and Computer Applications, Vol. 80, pages 109–122, Feb. 2017.

3.1 Introduction

Media production processes have become more complex and more data / network-intensive as they are increasingly dealing with high bitrate videos, deadline constrained network transfers and geographically distributed media production teams. Large quantities of data must be processed by multiple collaborating parties at different geographical locations. Media production environments are highly dynamic due to the arrival and departure of several requests of different sizes and requirements. In order to provide high-performance collaboration between different sites, next generation network reservation systems have to provide predictable performance and efficient bandwidth utilization. To ensure that bandwidth needs, delivery deadlines and requirements of different transfers are met, advance bandwidth reservation is needed. In general, advance reservation benefits the network operators as knowledge of future transmissions can be used to improve the admission control and provisioning to increase network utilization. It is also beneficial for the user as the network can provide better QoS to (future) requests with declared arrival and holding times [1], guaranteeing that the needed network capacity will be available. Advance reservation approaches can be either static or dynamic. While in a static approach all requests are known before scheduling, requests arrive one by one over time in a dynamic model.

In the media production industry, advance reservation scheduling of network transfers [2] is very important in order to make correct decisions on rejection or acceptance of future requests. In uncertain network conditions, such as sudden changes in network configuration, network fluctuations, failures, etc., additional precautions must be taken to guarantee successful transfers. The reliability of transfers in the media production networks is of prime importance and it can be enhanced using protection mechanisms. However, there are arguments against this redundancy as a large portion of network capacity will be wasted if the capacity assigned for this redundancy cannot be reused. As such, making use of idle network capacity and updating the resilient schedule over time, based on the current state of the network and running and planned transfers is of great advantage.

This work has been performed within the context of ICON MECaNO project [3], which provides solutions for the transmission of large media contents over an IP-based infrastructure, tailored to the quality and timing requirements of current and future media production process requests. In our previous work [4] and [5], we proposed both static and dynamic advance reservation scheduling approaches for a couple of interdependent requests of two types, video streams (VS) and file-based video transfers (FB). We have further presented the resilient version of these approaches based on a protection mechanism to improve the reliability of the advance reservation system [6]. The proposed scheme is capable of covering single link failures using pre-reserved disjoint backup paths. In this chapter, we make a tradeoff between reliability and resource usage in 3 ways: 1) the percentage of redundancy is defined for each individual request based on an input parameter provided by the customer, to influence the importance of reliability for each individual connection, 2) shared backup path protection (SBPP) [7, 8] is used, significantly reducing the bandwidth requirements for backup purposes, and 3) redundant reservations and network leftover capacity are reused as long as those are not being used for their primary purpose.

The main contribution of this paper is to make use of backups and idle bandwidth capacities to push more data into the network as long as advance reservations are partially unused as well as rapid reaction to sudden changes in uncertain network conditions using an event-based approach. Based on the resilient advance reservation approach, backups are ready for use, but are only activated when failures occur, leaving capacity unused. In addition, we have found that reservations made for video streams, are not completely utilized throughout the requested time. Video streams can be resumed and played-back multiple times during the reserved period, which causes idle reservations between resumes and playbacks. In our proposed approach, these unused capacities can be exploited to transfer additional data. This means that we use these reserved capacities as double-purpose, prioritizing their original purpose. In doing so, as long as these reserved capacities are idle, additional data can be transferred and as soon as for example a video stream becomes active, an event will be raised to prioritize the advance reservation made for this streaming request over the extra data transfers.

The proposed approach consists of two sequential processes. First, the network and transfers status are being continually monitored and the advance reservations are periodically updated. Second, the backup and unused network capacities, e.g. unused video stream reservations, are re-utilized to transfer more data than the schedule made by the resilient advance reservation algorithms. In unreliable networks, as soon as any failure is detected, an event will trigger our proposed algorithm to adapt the ongoing network transfers according to the current state of the network. This leads to a better utilization of substrate network resources, higher success rate and rapid reaction to sudden changes when the network is in operation.

The rest of this chapter is organized as follows. Section 3.2, describes background and related work. The envisioned media production network and the runtime adaptation approach are explained in Section 3.3. The proposed algorithms are described in Section 3.4. Section 3.5 provides simulation results and Section 3.6 concludes the chapter.

3.2 Related work

3.2.1 Advance resource reservation

Advance network resource reservation has applications for both wide-area and grid networks and has been studied frequently in recent years [9–15]. Current research mostly focuses on optical networks in combination with wavelength division multiplexing [1]. Advance reservation requests can be classified in 4 individual categories [1]. This classification is also valid for different types of requests in media production networks and all classes are supported in our work. In optical networks, the static advance reservation problem is first introduced by Kuri et al. [16, 17], who focus on requests with specified start time and duration and proposed heuristics and meta-heuristics to solve the static problem. The authors in [18, 19] were the first to propose dynamic advance reservation in fixed time-slotted networks. By introducing the percentage of known requests in [4], both static and dynamic traffic models are considered in our approach. Xie et al. in [20] proposed ILP-based models and heuristic approaches on re-routing in advance reservation networks in order to maximize admittance of new requests. The authors in [21, 22] focused on advance bandwidth reservation for on-demand data transfer in scientific applications. These approaches, however, purely focus on data transfers, not video streaming requests, dependencies among different transfers are ignored and no fault tolerance techniques are considered for possible failures.

3.2.2 Resilient reservation

Adding resilience into a reservation system can be achieved through restoration or protection failure recovery mechanisms [23]. In [24] a resilient advance reservation mechanism is proposed in optical grids. Due to the lower cost of restoration mechanisms, they use the latter. Burchard et al. in [25] consider a recovery mechanism for advance reservations in grid environments. The idea is to re-schedule failed but unstarted requests whenever failure occurs, but the main focus is on estimating the downtime. The authors in [26] have also focused on a proactive approach by taking resource statistical failure information into account. Their method relies on failure prediction and avoiding vulnerable resources. The authors in [27] present a fault-tolerant job scheduling approach for grid environments using adaptive task replication, which is a recovery approach. Providing resiliency in optical WDM networks through shared path protection has been proposed in [28–31]. Since meeting strict deadlines and QoS requirements is of great importance in our approach, using protection mechanisms tends to be more reliable.

3.2.3 Media production networks

The work presented in this chapter consists of two complementary approaches for media production networks. The combination of a customized resilient AR approach with a highly dynamic event-driven runtime adaptation approach consists of several functions which are of essential importance in the considered media production networks and have not been previously studied in the context of advance reservations.

This work proposes a dual approach which partially makes use of our previous works [5] and [6]. In [5], we devised an ILP-based model and proposed heuristic approaches for an exact solution. We showed that the heuristics yielded favorable results in much less time complexity than the linear programs. In [6], we enhanced the media production reservation system and made it more reliable in case of failures by following a protection mechanism and provisioning backup reservations for each request. As redundancy imposes cost and resource waste, the main motivation of our approach is to mitigate the side-effect of using redundant reservations by employing underutilized network capacities for transferring extra data as long as those are not needed for redundancy purposes. This work in the extension of [32] in which design of the proposed approach is explained in depth and [33] where the initial evaluation of our proposed approach without considering the impact of failure rates (stable network conditions) and video stream pauses/restarts was presented. This work differs from our previous work as it provides a highly dynamic, complementary and discrete-event driven approach which improves both reliability and performance of media production reservation systems over the time when the network is in operation. In this chapter, the impact of different failure rates on the performance of the runtime adaptation approach is extensively evaluated.

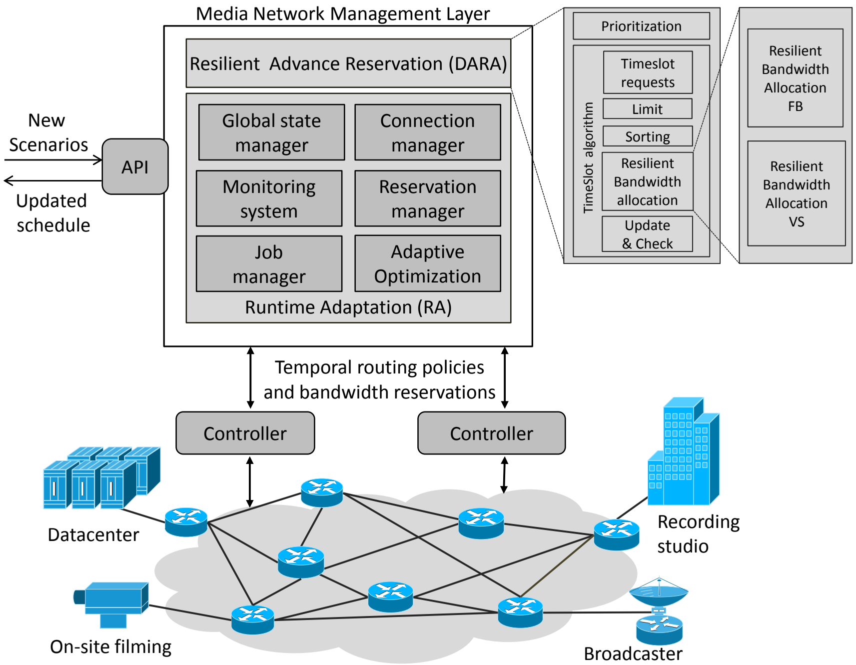


Figure 3.1: Different components of media production network.

3.3 Runtime adaptation approach in media production networks

3.3.1 Envisioned media production network

The envisioned media production network for MECaNO is depicted in Figure 3.1. The different actors and locations involved in the media production process, such as recording studios, on-site filming crews, broadcasters, and storage datacenters, are connected to a shared wide-area network. The management layer provides a reservation interface that allows the users of the network to submit their requests. The management layer contains two complementary processes of the proposed approach: the dynamic version of resilient advance reservation algorithm and the runtime adaptation approach, which we refer to as DARA and RA respectively. The DARA scheduling component is responsible for reserving the required amount of bandwidth including backup capacities for all requests and the RA component dynamically re-optimizes the request transmissions. Different components of the advance reservation scheduler are extensively explained in [5] and [6].

In the DARA approach, we deal with dynamic time-dependent reservations based on time constraints given by the user, stating the earliest start time and latest completion time. We discretize dynamic network into several snapshots and apply bandwidth allocation algorithms efficiently on every snapshot of the graphs and produce a schedule consisting of distinct reserved capacities in every time interval for each admitted request. A time interval or timeslot is a period of time in which reservations remain invariant. Multiple requests in media production networks may depend on each other, meaning that one request can only start when other requests that are dealt with, have been finished. This interdependency is explicitly incorporated in our approaches. A multi-path routing scheme is followed and the bandwidth scheduling algorithms is based on extending the classical shortest path and maximum flow problems, i.e. modified version of the Dijkstra [34] and Edmonds-Karp [35] algorithms.

An example of advance reservation schedules provided by the DARA algorithm is shown in Figure 3.2. As can be seen, in every timeslot each request has been served with different allocations, considering the request demands and network capacities. Two individual schedules are generated separately for primary and backup reservations. This scheduling is employed by the reservation system as long as no new scenario is submitted to the system. In this context, scenario refers to a set of several interdependent video transfer requests.

Our advance reservation algorithms support rescheduling in order to incorporate new requests at runtime. As shown in [36–38], advance reservations decrease network utilization if dynamic reservations are also supported. To improve network utilization, in DARA approach, we assume that whenever a new scenario enters into the reservation system, all new and advance-scheduled requests are being re-scheduled. Since a fixed timeslot-based approach has been followed, if the new scenario is admitted, the entire schedule will be updated from the next time interval.

In the DARA algorithm, the backup paths are disjoint from the primary ones. The provisioned protection method guarantees a single link failure recovery. The backups are determined to fulfill the maximum bandwidth allocated on the links of the primary paths. This means that to provide 100% backup, there is no need to allocate the exact amount of bandwidth as in the primary paths [6]. The amount of backup reservation also depends on how the primary demands are allocated. To make it clearer Figure 3.3 is depicted for a request with 30Mbps primary allocation and 100% backup demand. This figure indicates that how different ways of allocating primary paths can affect the amount of backup demand. In Figure 3.3a three paths of 10Mbps are allocated to the request. Therefore, it is sufficient for the shared backup to provide 10Mbps. In figure 3.3b one path is dedicated as primary. In this case the backup has to offer full primary capacity which is 30Mbps. In the third case, backup path offers 17Mbps, which equals to the maximum bandwidth reservation among all primary allocations.

Based on the outcome of the DARA approach, the requests are either rejected

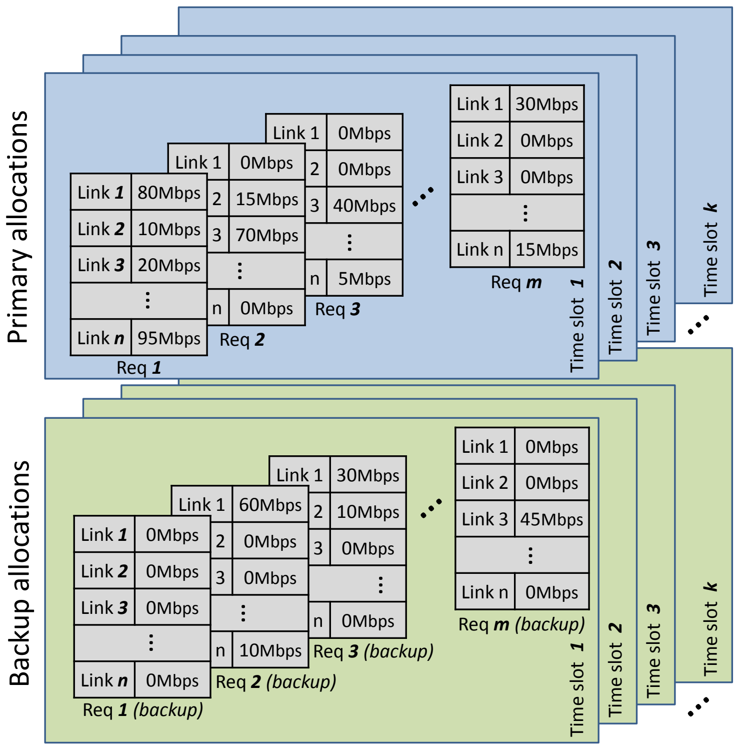


Figure 3.2: The primary and backup schedules provided by DARA.

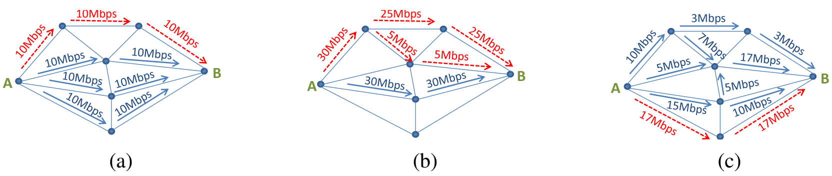
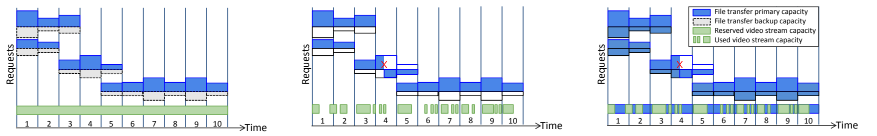


Figure 3.3: Dependency of backup demand on the reserved primary paths. (Blue: Primary reservation, Red and dashed: Backup reservation)



(a) Reservations based on the DARA

(b) Transfers based on the DARA in practice

(c) Transfers based on the RA in practice

Figure 3.4: Comparing the DARA algorithm in theory, in practice and the RA approach contribution.

or admitted. However, in presence of failures, not all the admitted requests can be completely transferred. Hence, the admitted requests can be categorized as *succeeded*, *degraded* or *failed*. Succeeded requests are those that have been fully transmitted. Deciding on the degraded or failed states depends on the users' preference. In this work, we assume that the users asked for the same value as percentage of backup demand, i.e. if a request has a demand for 60% backup, this request is considered as degraded if at least 60% (but less than 100%) of its volume has been transferred by its deadline, otherwise the request is failed. It should be noted that for 0% and 100% of backup demand, no degradation has been considered. Those requests are either fully-transferred or failed.

3.3.2 Runtime adaptation (RA) methodology

Figure 3.4a illustrates an example schedule for four file transfers and one video stream based on the DARA algorithm in a time span of 10 time intervals. For the file transfers, the parts in blue show the primary bandwidth allocations and the parts in gray refer to the backups which are provisioned to be used when a failure occurs in order to be able to transfer the video according to the agreed SLA (Service Level Agreement). Figure 3.4b shows how the network operates in practice: the backup paths are seldom in use and the reserved bandwidth for the video streams are not continuously utilized, resulting in wasted network capacity. As can be seen in Figure 3.4c, to have a higher performance and network utilization, we propose a hybrid approach that combines the DARA scheduling approach with an online adaptation system which uses wasted network capacity to increase network utilization.

The RA approach follows two sequential phases in every timeslot: 1) the periodic update and 2) the periodic adaptation. Dynamic network conditions (such as fluctuations, failures, etc.) affect the allocated capacities and network status. As such, the periodic update is repeated before the end of every timeslot to take into account the real transmitted data instead of scheduled ones and update the schedule based on recent information. The periodic update is followed by the periodic

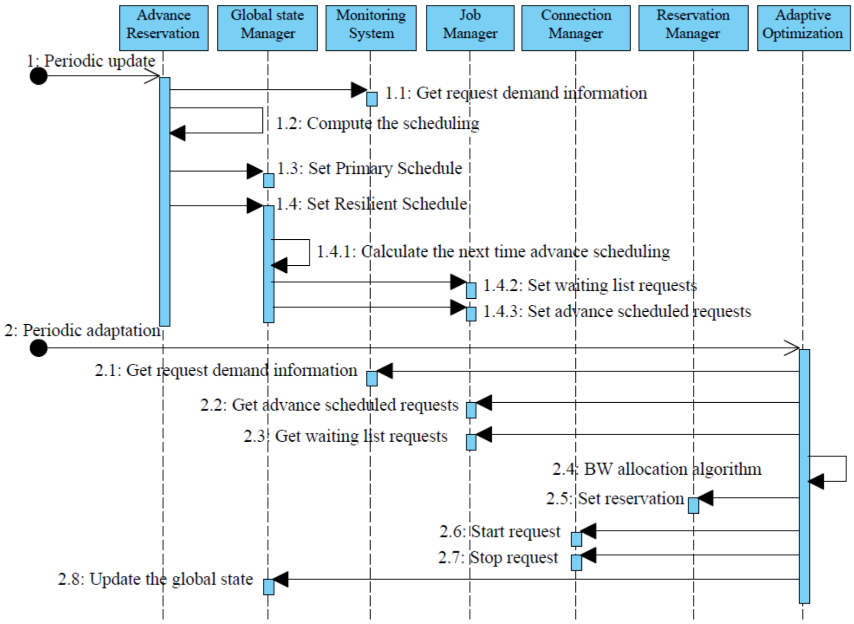


Figure 3.5: The collaboration between different components in every timeslot in the RA approach.

adaptation, which is a complementary step to continually adapt network transfers, taking into account the current state of network and transfers and making use of idle network capacity. The periodic adaptation phase is in operation throughout the next time interval.

As illustrated in Figure 3.5, the RA approach consists of seven components as follows:

- **Advance reservation**: in charge of producing a schedule using the DARA scheduling algorithm. The DARA algorithm is invoked under two circumstances. First, when new scenarios enter the reservation system, leading to an update of the entire schedule for all admitted and unfinished requests. Second, when the schedule needs to be updated during the periodic update. Since periodic adaptation algorithms make use of idle network capacities and real transfers potentially run ahead of schedule, the latter is necessary to take into account the extra transfers and do the re-scheduling for the residual demands. In both cases the schedule is modified at the start of the next timeslot.

- **Global state manager**: contains all information about scheduling, network and request reservations, connections, demands, deadlines, etc. The time when the current timeslot is started or when it finishes can be retrieved from the global state.

- **Monitoring system**: keeps track of monitored times, residual demand and current allocated bandwidth for all requests. The monitoring system also regularly checks network conditions and raises an event as soon as a failure is detected.

- **Job manager**: contains the list of current advance-scheduled requests and current waiting list requests. Advance-scheduled requests refer to the requests that have already been scheduled by the DARA algorithm to be transferred in the current timeslot. The waiting-list requests are those requests that can potentially be started in this timeslot, but are postponed due to limited network capacity.

- **Connection manager**: decides what to do when a transfer is started or stopped. As long as there are requests with active connections, this component is operational. Whenever a connection for a file transfer is terminated, the links those were in use by this connection become free. In order to improve network utilization, this capacity can be used by other active requests if shared links were in use. To achieve this, after completion of a file transfer, an event will be raised.

- **Reservation manager**: collects all the information about the reservations of each request. Primary allocations, backup reservations, extra allocations made during the periodic adaptation phase and allocated network resources can be retrieved from this component.

- **Adaptive optimization**: in charge of optimization to try and push more data than what has been guaranteed through advance reservation. The Adaptive Optimization (AO) algorithm is the main algorithm in this component which is triggered by several events: start of a timeslot, start and end of file based requests, link failures and repairs. Based on this algorithm, the current schedule is analyzed and adapted to use idle bandwidth capacities.

The rest of this section describes the way the RA phases make use of these components to contribute in performance improvement of media production networks.

3.3.3 First phase: Periodic update

During the periodic update, first the current status of the network and transfers are monitored and then the DARA algorithm is invoked. This process updates the entire schedule based on the information retrieved from the monitoring system. This new information will be set in the global state manager. Then the next timeslot reservations are derived from the advance reservation schedule and are set as advance-scheduled requests in the Job manager. The list of advance-scheduled requests contains all requests which have been scheduled to be transferred from now on. Take into account that potentially there are other requests which could be started, but have been postponed due to bandwidth constraints. These requests are kept in a waiting list and used in the periodic update phase.

3.3.4 Second phase: Periodic adaptation

The Adaptive Optimization (AO) algorithm is the main algorithm in the periodic adaptation phase. The AO algorithm is triggered several times, i.e. whenever a file-based video starts/finishes transferring , and in the case of any link failure or link repair. The first invocation of this algorithm is before the start of the next timeslot. Based on this algorithm, in this step the scheduling over the next timeslot is analyzed and modified to make use of idle bandwidth capacities. To achieve this, the advance-scheduled requests are retrieved from the job manager and then the reservations for backups and video streams are ignored (Because video streams may not always be active and can be resumed/played-back multiple times throughout the reserved period). This gives us a network in which only the primary reservations occupy the network capacities.

3.3.5 Modeling of the runtime adaptation methodology

In order to model the dynamic aspect of the proposed approach, we have designed a discrete-event based simulator in which the following events are noteworthy:

- Scheduling update: When the DARA algorithm is finished, this event is raised.
- File-based video transfer start time: When a file transfer starts, the AO algorithm is invoked for all active and new requests.
- File-based video transfer stop time: The fully completed request is removed, the other active requests' demands are updated and the AO algorithm is invoked. The previously calculated end times of other active requests are canceled.
- Video stream start time or play-back: The video stream transmission is started and allocated bandwidth for the affected file transfers are updated based on the information provided by the VS activation/deactivation handler.

- Video stream stop time or resume: The allocated bandwidth for the affected file transfers reset to the previous value provided by the AO algorithm.
- Global state update: This event is raised to update reservations and connections, etc.
- Failure: As soon as any failure is detected, the link failure/repair handler is invoked, based on which the failed link is removed from the network graph and the AO algorithm is invoked to adapt ongoing transfers.
- Repair: When the failure is resolved, the link failure/repair handler is invoked.

3.4 Runtime Adaptation (RA) algorithms

In this section the algorithms which are used in the periodic update and periodic adaptation phases of the RA approach, shown in Figure 3.6, are described.

3.4.1 Periodic update algorithms

The periodic update phase consists of two main steps: The *UpdateRequestsInfo* algorithm, shown in algorithm 6, and the DARA algorithm. We do not elaborate on the DARA algorithm in-detail as it has already been explained in-depth in [6]. In the *UpdateRequestsInfo* algorithm, the demand of submitted requests is updated. To achieve this, first finished and unadmitted requests are removed from the reservation system and then demand of all other submitted requests is updated based on the type of request. For file-based requests however, we deal with volume, so the allocated bandwidth is not fixed and may vary from one timeslot to another. For video steaming requests we deal with fixed bandwidth requirements. Therefore, for file transfers the last monitoring time, last allocated bandwidth and residual transfer volume are updated based on monitored information. If the residual demand of a file-based request is zero, the request has been finished and has to be removed. For video streams, the requests whose deadlines (t_e^n(rq)) are expired are removed. As our approach supports interdependencies among requests, for all requests, list of dependencies are adjusted in case there is any start time dependency to removed requests.

3.4.2 Periodic adaptation algorithms

The *Adaptive Optimization (AO)* algorithm, which is frequently triggered in the periodic adaptation phase of the RA approach, is shown in Algorithm 7. This algorithm also triggers the *UpdateRequestsInfo* algorithm. Therefore, demands of all requests are already updated whenever the AO algorithm is called.

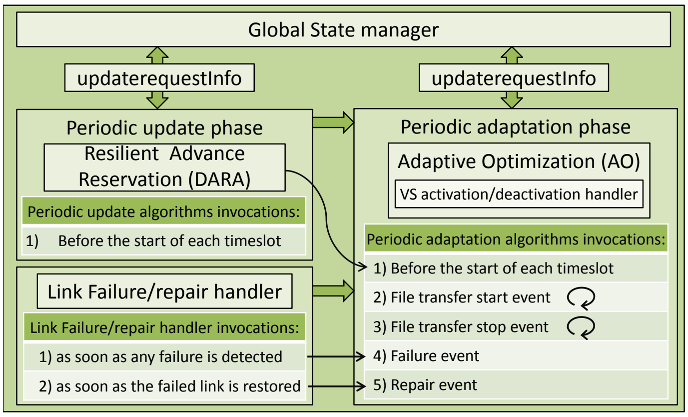


Figure 3.6: Algorithms used in periodic update and periodic adaptation phases of the RA approach. Narrow arrows show invocations.

algorithm 6: The UpdateRequestsInfo algorithm.

for *(rq $\in$ Set of submitted requests)* **do**
  if *(request not admitted)* **then**
    removedReq.add(rq);
  else if *(Type (rq) $=$ FB)* **then**
    LastMonitoringTime(rq) $\leftarrow$ current time;
    LastAllocatedBW(rq) $\leftarrow$ monitoring.getBW(rq);
    residualVol(rq) $\leftarrow$ monitoring.getVol(rq);
    if *(residualVol(rq) $=$ 0)* **then** removedReq.add(rq);
  else if *(Type(rq) $=$ VS & t_e^n(rq) $<$ current time)* **then**
    removedReq.add(rq)
  end
end
for *(rq $\in$ Set of submitted requests)* **do**
  SetDependencylist(rq).remove(removedReq);
end

Based on the AO algorithm, the advance-scheduled requests (ASReq) and the list of waiting requests (WLReq) are retrieved from the job manager. In both lists the requests are sorted. The main factor for this sorting is the estimated deadline: the sooner the deadline, the higher the priority. The second factor, volume, comes into consideration only when the deadlines are equal, the higher the demand, the higher the priority. Then, reservations made for backups and video streams (VSs) are ignored. This gives us a network in which only the primary reservations occupy network capacity. Then, for the advance-scheduled requests, new extra allocations (ExtraAlloc) for requests over this residual graph are computed. Extra reservations are aggregated with the primary reservations of requests and their assigned band-

algorithm 7: The Adaptive Optimization (AO) algorithm which is the main algorithm in the periodic adaptation phase of the RA approach.

```
UpdateRequestInfo();
ASReq ← JobManager.getASReq(current time);
WLReq ← JobManager.getWLReq(current time);
PriorityBasedSorting (ASReq);
PriorityBasedSorting (WLReq);
graph ← GlobalState.getGraph(current time);
graph.removeReservations(VSs, Backups);
ExtraAlloc ← BWallocation(ASReq, graph);
for (rq ∈ ASReq) do
    TotalAlloc(rq) ← PrimaryAlloc(rq) + ExtraAlloc(rq);
    FinishTime(rq) ← residualVol(rq)/TotalAlloc(rq);
    rq.SetReservations(TotalAlloc(rq));
    rq.StartConnection(current time);
    rq.FinishConnection(FinishTime(rq));
end
WLAlloc ← BWallocation(WLReq, graph);
for (rq ∈ WLReq) do
    if (WLAlloc(rq) ≠ 0) then
        FinishTime(rq) ← residualVol(rq)/WLAlloc(rq);
        rq.SetReservations(WLAlloc(rq));
        rq.StartConnection(current time);
        rq.FinishConnection(FinishTime(rq));
    end
end
```

width will be potentially increased. For each request, new allocations are updated in the reservation manager. Based on these new allocations, the start time and finish time of requests are configured in the connection manager. All the reservation and connection information is saved in the global manager. The same steps are repeated for the waiting-list requests. The only difference is that there is no primary reservations for those requests.

A request that finishes will raise an event which first cancels the stop time events of all other active requests. Then, the AO algorithm is triggered to calculate extra allocations and finish times. Since a request just finished, these new finish times will be earlier than the previously canceled ones. Moreover, whenever a new request is ready to be started within a time interval, this may also raise another event to trigger the AO algorithm. This cycle is repeated as long as requests trigger events.

Detecting a failure or repair throws an event whose handler is shown in Algorithm 8. Based on this algorithm, first the failed/restored network elements are removed from/restored to the network. By calculating the effect of failures on each request, the primary and backup reservations of all affected requests are *algorithm 8: Link failure/repair event handler.*

```
input: failed/repaired link, network graph
FailedLinks.add/remove(link);
if (FailedLinks ≠ empty) then
  | graph.removeFailedLinks(FailedLinks);
  | GlobalState.setGraph(current time);
end
for (rq ∈ requests affected by failure) do
  | if (PrimaryAlloc(rq).contains(FailedLinks)) then
  |   | rq.UpdatePrimaryAllocations(FailedLinks);
  | else if (BackupAlloc(rq).contains(FailedLinks)) then
  |   | rq.UpdateBackupAllocations(FailedLinks);
  | end
end
AdaptiveOptimization();
```

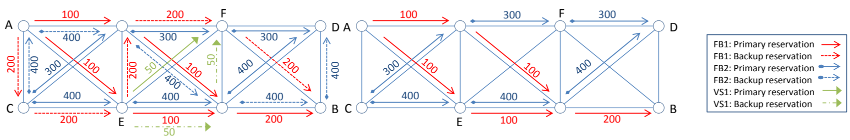
updated. The AO algorithm is invoked to re-optimize ongoing transfers, taking into account network failure status. In general, Algorithm 8 allows the reservation system to adjust in-advance reservations for affected requests and makes new additional reservations over residual network capacity.

Algorithm 9 shows how file based transfers' bandwidth allocations are varied when transmission of video streaming requests (with B^{VS} bandwidth demand over the VSlinks) are started. This algorithm is executed whenever the AO algorithm is invoked to determine which requests have to be restrained to serve the video streaming request whenever it becomes active. In order to do this, first the algorithm checks the residual network capacity to update the video stream demand if part of the advance reservations for this request is still untouched. Then, ongoing file transfers are sorted from lowest priority to highest, common links (Clinks) between video stream and extra allocation of file transfers are checked and the common allocations are removed from the extra allocations of file-based request. This is repeated until the video stream demand is fulfilled. Note that this is a provisioning algorithm to re-act immediately as soon as a video stream request starts using its reservations.

3.4.3 Clarifying examples

Figure 3.7 shows an example of the difference between the reservations made by the DARA algorithm and the reservation as input of the AO algorithm in periodic adaptation phase of the RA approach. As can be seen in Figure 3.7a, two file-based transfers and one video stream are active with primary and backup paths in one timeslot. FB1 is a file-based request from node A to node B with 200Mbps multipath allocation, FB2 is similar to FB1 but from node C to node D and 700Mbps reservation, video stream VS1 has a requirement of 50Mbps, from node E to node *algorithm 9: VS activation/deactivation handler.*

input: request VS
VSlinks $\leftarrow$ AdvanceReservedLinks(VS);
for *($l \in$ VSlinks)* **do**
    VSdemand(l) $\leftarrow B^{VS} -$ residualCapacity(l);
    if *(VSdemand(l) ≤ 0)* **then** VSlinks.remove(l);
end
for *(FB $\in$ File-Based requests)* **do**
    CLinks $\leftarrow$ CommonLinks(ExtraAlloc(FB),VSLinks);
    for *($l \in$ CLinks)* **do**
        x $\leftarrow$ ExtraAlloc(FB, l);
        if *(x $\geq$ VSdemand(l))* **then** x $\leftarrow$ VSdemand(l);
        ExtraAlloc(FB) $\leftarrow$ ExtraAlloc(FB) $-$ x;
        VSdemand(l) $\leftarrow$ VSdemand(l) $-$ x;
        if *(VSdemand(l) $= 0$)* **then** VSlinks.remove(l);
    end
end



(a) Reservations made by the DARA algorithm (b) Reservations considered by the AO algorithm

Figure 3.7: Comparing a snapshot view of bandwidth reservations for 3 requests as output of the DARA algorithm and as input of the AO algorithm (Full lines: primary reservations, dashed lines: backup reservations, Open arrows: file-based transfers).

F. As shown in this figure, for each individual request, primary and backup paths do not have any link in common. All three requests had a requirement of 100% backup and the amount of capacity reserved as backups equals the maximum allocated bandwidth along the primary paths, which is equal to 200Mbps, 400Mbps and 50Mbps for FB1, FB2 and VS1 respectively.

Figure 3.7b shows the reservations taken into account in the AO algorithm before the start of the next timeslot. The video stream and all the backups are eliminated and only file transfer primary paths are kept. Taking this network into account, extra allocations for each request are calculated. These extra reservations will be aggregated with the primary reservations of the requests and their assigned bandwidth will be increased. For each request the new allocations are updated in the reservation manager. Whenever a request is finished, the bandwidth reserved for that finished request is returned to the network resource pool and the AO algorithm is re-invoked to determine new allocations. Based on the new allocations, the

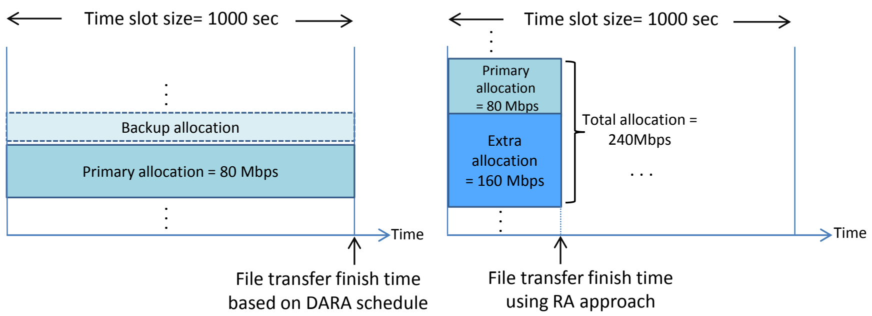


Figure 3.8: Differences between bandwidth allocation algorithms in the DARA and RA approaches.

start time and finish time of the requests are set and kept in the connection manager. This cycle is repeated until all requests are finished or the timeslot ends. In case a video stream is activated, the advance reservation for this request is prioritized and other extra allocations that have made use of this capacity, are interrupted. Applying the AO approach, the reservations will not remain fixed during each time interval. As for the actual transfers, we make use of the extra allocations in addition to the primary allocations, in stable network conditions the actual transfers are higher than what had been previously envisioned by the DARA approach.

Figure 3.8 clarifies the differences between bandwidth allocation algorithms in the DARA approach and in the AO algorithm of the RA approach. The key difference is that the advance reservation algorithm takes the size of the timeslots into account. For example for a 10 GB file, the primary allocated bandwidth is 80 Mbps which is enough for the file to be transferred in a 1000-second timeslot. Note that based on the DARA approach, the allocated bandwidth may vary from timeslot to timeslot but it is fixed within each timeslot. Using the RA approach, the allocations may vary when the network is in operation, even within timeslots. The extra allocation for the request is calculated, depending on the spare network capacity. Considering the extra allocation of e.g. 160Mbps, the sum of all allocations is three times higher than the primary allocation. Therefore, the file transfer is finished in a third of the time of the finish time which was computed in advance when no failures occur.

During the runtime, any early finish will trigger an event which indicates that the present connection can be torn down. The links which were in use by this request are now free, allowing other active requests from the advance-scheduled list, to use more bandwidth if the shared links were in use. Furthermore, there can be other future requests in the advance reservation schedule which can make use of some of these links. Since these links have already been reserved for a finished request, the future requests stored in the waiting list, could be analyzed and poten-

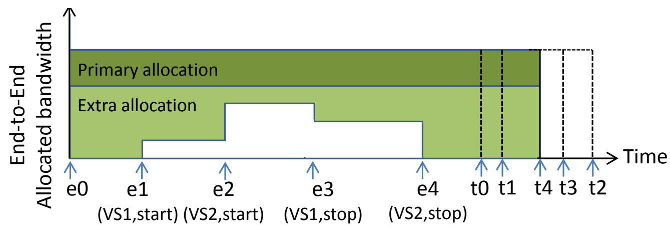


Figure 3.9: The impact of video streaming requests activation / deactivation on file transfer finish time in the periodic adaptation phase of the RA approach.

tially scheduled. To achieve this, the finished request is removed from the request list and the AO algorithm is triggered. In doing so, the corresponding file transfers begin earlier than they were scheduled by the advance reservation scheduler, improving link utilization.

As the reserved capacities for video streams and backups have a double purpose, pre-determining how to manage the conflicts before they happen is crucial. To achieve this, two important functions called failure/repair handler and VS activation/deactivation handler are proposed. The VS activation/deactivation handler determines when any video stream is active, the extra allocation of which file-based requests are affected and how the extra allocation of the affected request is adjusted to reflect this activation. The failure/repair handler determines how to handle the conflicts when the backup reservations are active for their original purpose. During the runtime, any failure/repair or any video streaming start/stop or resume/play-back will prompt an event and the information provided by these functions allows the management system to quickly handle the event. To elaborate more on this, Figure 3.9 shows how video streaming requests' activation/deactivation affect the extra allocation of a file transfer, allocated by the RA approach. As can be observed from this figure, the finish time of a file transfer is adjusted as soon as a video stream request starts/stops. Five events e0-e4 cause 5 different finish time t0-t4 for the file transfer request. e0 is the first event in the periodic adaptation phase, at the beginning of the timeslot when no VS is active, and the estimated finish time is t0. VS1 activation raises an event (e1), based on which t1 is calculated as the new finish time and then t0 is substituted by t1. Other VSs activations/deactivations have the same impact on the file transfer finish time. Eventually as e4 is the final event in this timeslot, the file transfer is finished at t4.

Several invocations of the AO algorithm during the periodic adaptation phase are shown in depth in Figure 3.10. The reservations made by the DARA algorithm for the first timeslot, starting at 0 and ending at 300s, are illustrated in Figure 3.10(a). We assume that no failures are occurred during this time interval. Three file-based transfers FB1, FB2 and FB3 are reserved with 100%, 50% and 30% backup (shown in dashed rectangles) respectively. Figures 3.10(b)-(f) show

how the AO algorithm in the periodic adaptation phase is being used to optimize the transmissions. In Figure 3.10(b), the primary reservations are adapted by the first invocation of the AO algorithm before the timeslot starts. Based on these new allocations, FB1 finishes at 140s, FB2 at 200s and FB3 at 100s. As FB3 has the earliest finish time, the AO is invoked again at 100s. This is shown in Figure 3.10(c). As the capacity used by FB3 has now been released, FB4 has the opportunity to start earlier. Based on the advance reservations, FB4 should have started in the next time interval at 300s, but thanks to the AO algorithm, it can be started earlier at 100s. The stop time of FB1 and FB2 is also updated from 200s to 150s and from 140s to 130s respectively. As can be seen in Figure 3.10(d), the next time for re-invocation of the AO algorithm is at 130s when FB2 is terminated.

It should be noted that for some requests there could be enough capacity to be transferred but due to inter-dependencies on other requests, they have to be postponed. The RA approach tries to accommodate these requests as soon as their dependencies are eliminated. To elaborate more on this, we assume that FB5 has a dependency to FB2 and can only be started when FB2 finishes. Based on the advance reservation schedule FB5 cannot start in this timeslot as FB2 is in operation. Now, using the RA approach, it can be started earlier at 130s instead of the next timeslot. As Figure 3.10(e) shows, only FB4 and FB5 are active from time 140s. This means that all primary reservations have been transferred by then which indicates that the transfer of advance-scheduled requests is 160s ahead of the resilient schedule. The last invocation takes place at 190s when FB4 finishes and from then only FB5 is being transferred. This is shown in Figure 3.10(f). This example also shows that the RA approach tries to mitigate the side-effects of fixed time intervals. In periodic adaptation step, not only the spare capacities are re-used, but also the allocated bandwidths vary within timeslots, resulting in a more flexible transfers comparing to the DARA schedule.

Figure 3.11 illustrates the impact of periodic update and periodic adaptation algorithms on the performance of bandwidth reservation system within 3 timeslots. FB1, FB2 and FB3 asked for 100%, 30% and 50% backup respectively. According to the advance reservations, FB1 has bandwidth allocations of 200Mbps in the first and 100Mbps in the second timeslots, FB2 150Mbps in the first and FB3 300Mbps in the third time interval. Applying the RA approach and by several invocations of the AO algorithm, which ignores the backup reservations, FB1 and FB2 have completely transferred in the first timeslot and transfer of FB3 has already been started. Before the start of the second timeslot, the schedule is updated during the periodic update phase. Therefore, the reminder of FB3 is shifted to the second timeslot and the bandwidth reservations for FB3 in the third timeslot are completely released. As can be seen when a new scenario is submitted to the reservation system at the end of the second timeslot, if the RA approach is not used, the management system would not have been able to serve the new scenario, but it is admitted thanks to the

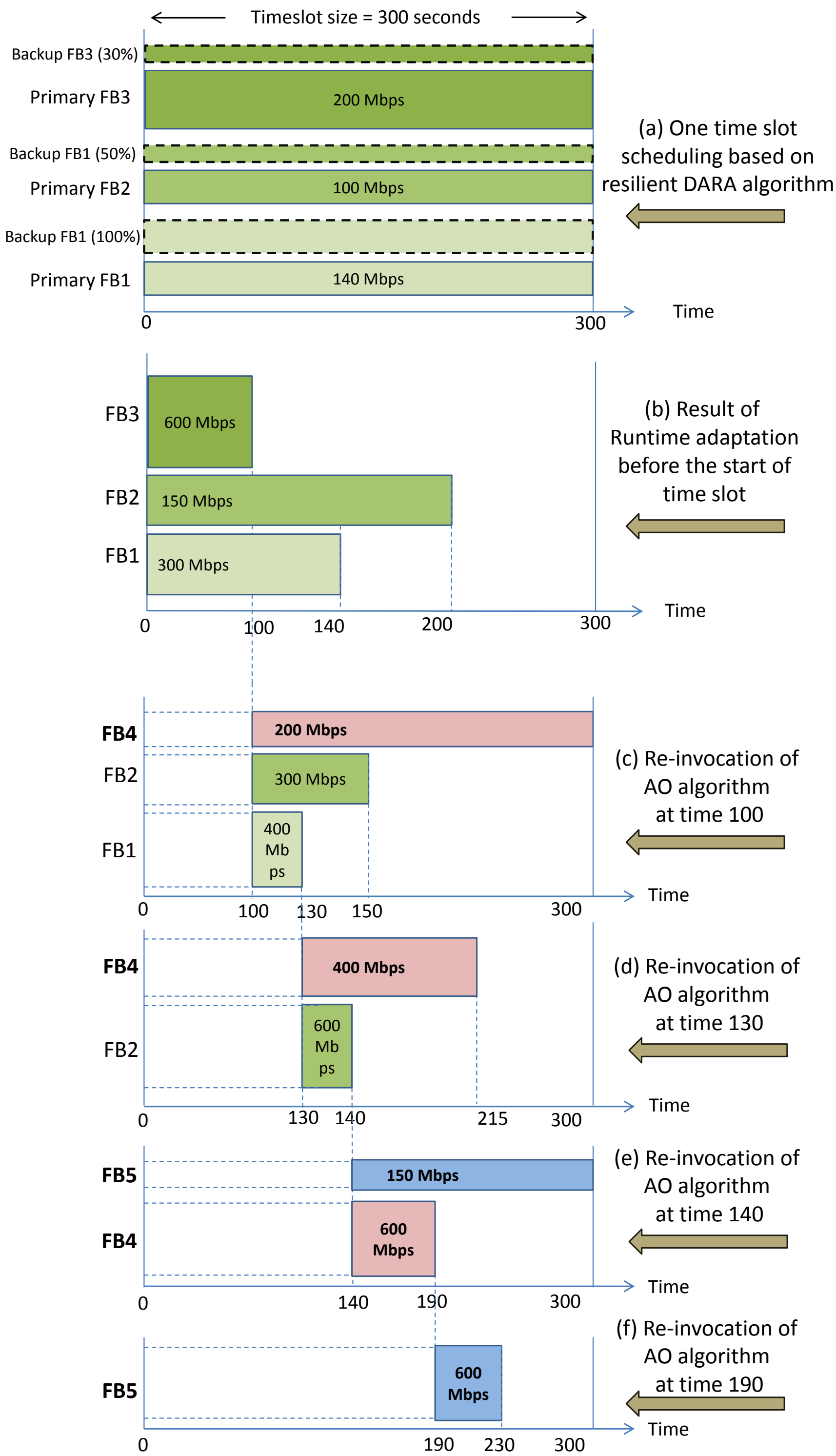


Figure 3.10: Single timeslot reservations made by the DARA algorithm and multiple re-invocations of the AO algorithm during the periodic adaptation phase of the RA approach.

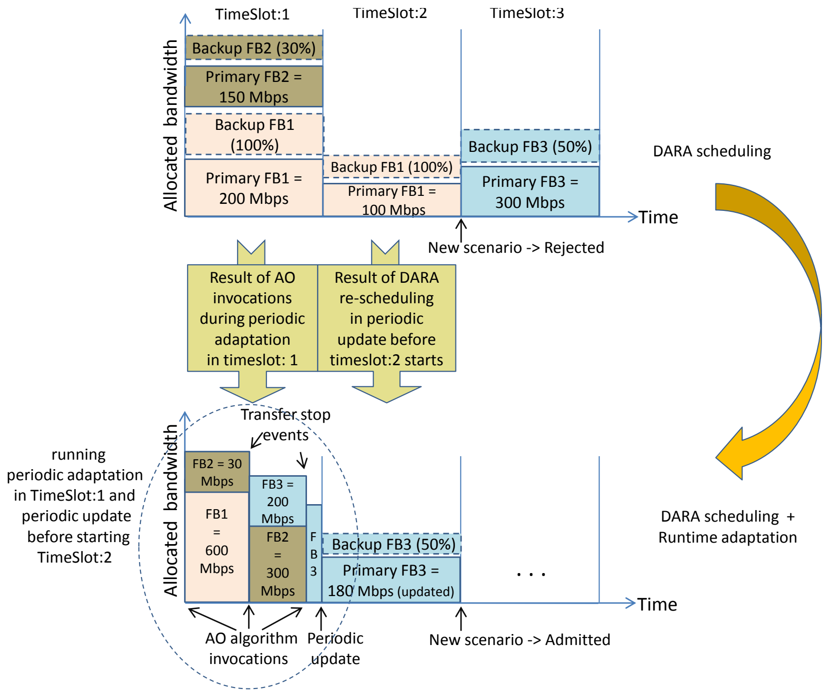


Figure 3.11: The impact of invocations of algorithms in periodic update and periodic adaptation phases of the RA approach on performance of reservation system.

RA approach.

3.5 Performance evaluation

In order to model the dynamic aspect of our model, we have designed a discrete-event-based simulator using the MASON multi-agent simulation toolkit [39]. In this section the impact of using a runtime adaptation approach is thoroughly evaluated and compared to the DARA algorithm. The DARA approach can be configured for different percentages of requests known in advance. This refers to the percentage of scenarios that are known at the start of the simulated period. In these evaluations, we assume that none of the scenarios are known in advance, which is the most realistic case. It should be noted that the DARA approach has been previously validated compared to an exact optimal ILP-based solution.

Throughout this section, *DARA*[*XX*%,*YY*]+*RA* denotes that dynamic version of resilient advance reservation approach with *XX*% of backup and failure rate of *YY* is used. The second part (*RA*), is optional and specifies if the runtime adaptation approach was used or not.

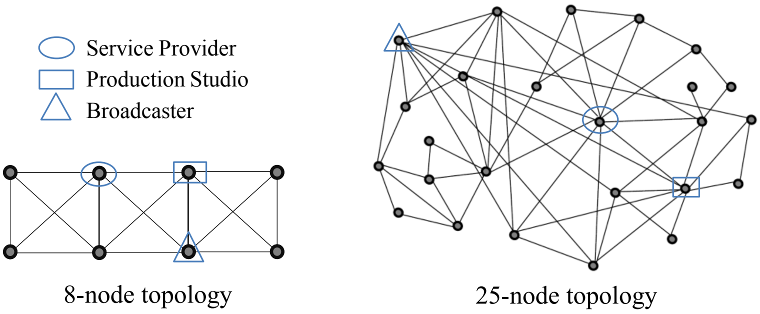


Figure 3.12: Media production network topologies used for evaluation.

3.5.1 Evaluation Setup

In this evaluation we have used 8-nodes and 25-nodes media production network topologies, depicted in Figure 3.12. The 25-node topology is the well-known ATT North America topology [40] consisting of 25 nodes and 56 bidirectional links (112 links in total) which matches to the size of realistic media production networks. The default network capacity is 300Mbps per link. We have previously defined three scenario templates based on the information gathered from several Belgian media production actors, including a broadcaster, service provider and recording facility provider [5]. Each scenario contains a collection of interdependent file and video streaming transfers with randomized parameters. Template1 is composed of 5 different file transfer requests. Template2 comprises 18 interdependent file transfers. The third template includes 4 file transfer requests and 4 video streams.

For the 8-node topology, the number of scenarios equals 20, of which 7, 7 and 6 are of template1, template2 and template3 respectively (209 requests in total). For the larger 25-node topology, the number of scenarios is 50, of which 17, 17 and 16 are of the first, second and third templates respectively (519 requests in total). A fixed time interval granularity of 1 hour is used. It should be noted that, every 1-hour timeslot in the AR approach is divided into several flexible timeslots by the RA approach. As in the considered scenario templates the requests are only known a few hours in advance, each simulation run covers a 24 hour period. All results are averaged over 50 runs with different generated scenarios, error bars denote the standard error.

In order to generate MTBF (mean time between failures), MTTR (mean time to repair) and video stream activation/deactivation events, we used a normal distribution function with equal values for both mean and standard deviation. This value equals 5 minutes for video stream activation/deactivation. It is not trivial to assign a value for MTTR, as it depends on multiple factors, e.g. type of links, type

Table 3.1: Maximum and average number of concurrent failures for different failure rates in 8-node and 25-node topologies.

	8-node topology (32 links)		25-node topology (112 links)	
Failure rate	**AVG**	**MAX**	**AVG**	**MAX**
1h	14.45	22	47.82	63
2h	8.76	17	31.97	46
4h	5.3	10	16.64	26
10h	2.17	6	8.47	20
20h	1.2	4	3.9	8

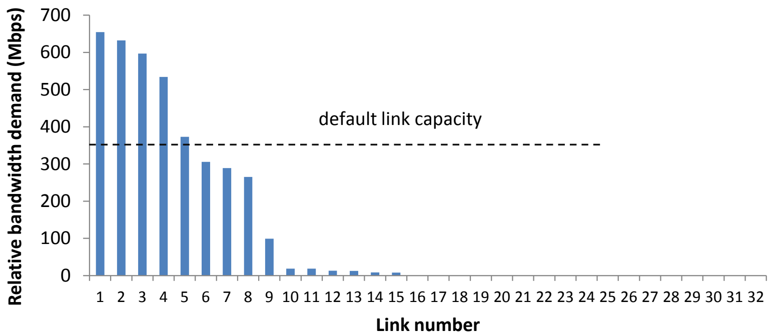
of failures, underlying technology [41]. The main focus of this section is to evaluate the performance of our approach under catastrophic failures in failure-prone networks. As such, 48 minutes is chosen as mean/standard deviation value for the MTTR to experience higher unavailability. However, in unstressed network conditions this value is reduced to 5 minutes. To give an insight in number of concurrent failed links for each topology, Table 4.1 shows the maximum and average number of failed links for different MTBF values under stressed network conditions.

The bandwidth contention per link for the 8-node and 25-node topologies is shown in Figure 3.13. When calculating bandwidth contention, we assume that all requests are admitted and use a single shortest path from source to sink. Contrary to the video streaming requests which have a fixed bandwidth demand, the bandwidth requirement of file-based transfers has to be estimated. Due to interdependencies among requests of each scenario, some requests may not have specified start or stop times. In order to have an estimation, the volume of all file transfer requests belonging to a scenario are divided by the time from when the earliest request of the scenario is ready to be transferred until its final deadline (the deadline of the latest request). Actors between scenarios move, except for some common locations. As shown in Figure 3.12, prim locations i.e. service provider, production studio and broadcaster, are located at highly connected nodes. Locations of other actors are randomly chosen. Therefore, Figures 3.13a and 3.13b show the connection to prime locations as a set of hotspots with high intensity. In the 8-node and 25-node topologies the highest contention per link is at maximum 655Mbps and 2,095Mbps respectively.

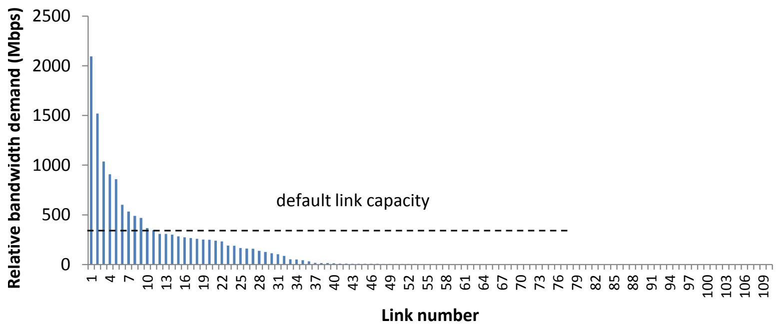
3.5.2 Impact of different failure rates, fixed backup demand

3.5.2.1 Impact of available bandwidth

First, we evaluate the impact of changing the network link capacity. Figure 3.14 and Figure 3.15 show the impact of available bandwidth and different failure rates on the performance of the RA approach for the 8-node and 25-node topology respectively. In both topologies, bandwidth capacity per link is parametrized from



(a) Bandwidth contention for the 8-node topology.



(b) Bandwidth contention for the 25-node topology.

Figure 3.13: Bandwidth contention per link for 20 scenarios in 8-node topology and 50 scenarios in 25-node topology.

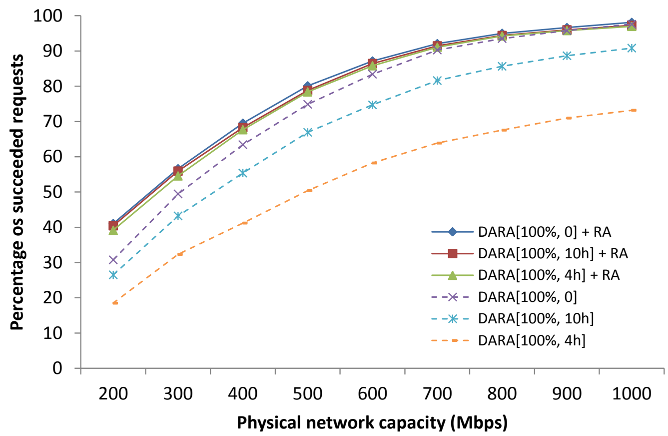


Figure 3.14: Impact of network capacity and failure rates on the performance of using the RA approach for the 8-node topology.

$200Mbps$ to $1Gbps$ and all scenarios' requests demand 100% of backup. These figures show that the RA outperforms the DARA up to 27.6% and 36.12% with 4-hour failure rates for 8-node and 25-node networks respectively. The standard error at maximum reaches to 1.16% in smaller and 1.5% in larger topologies (not shown for greater legibility). Our results show that regardless of the failure rate, the RA approach almost always outperforms the DARA approach.

3.5.2.2 Impact of network load

Figure 3.16 and Figure 3.17 show the impact of network load and different failure rates on the performance of the RA approach using the 8-node and 25-node topology respectively with a backup demand of 100%. Since the network capacity remains fixed, adding more requests leads to an increase in rejection rate. The results show this for both smaller and larger topologies, the RA approach improves the percentage of admitted requests up to 23% and 30% on average respectively.

3.5.3 Impact of different backup demands, fixed failure rate

3.5.3.1 Impact of available bandwidth

Figures 3.18a, 3.18b, 3.19a and 3.19b analyze the impact of network capacity and percentage of backup demand on the performance of our approaches. In these evaluations, backup demand of 0% and 100% and failure rate of 10 hours are taken into account. Figure 3.18a and Figure 3.19a show the average percentage of succeeded requests (out of all submitted requests) in 8-node and 25-node networks

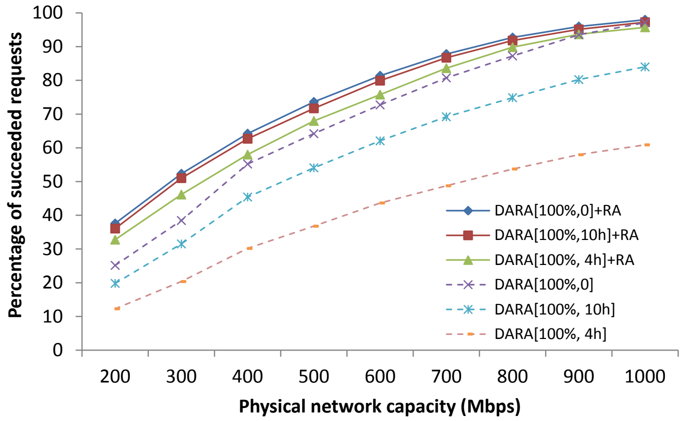


Figure 3.15: Impact of network capacity and failure rates on the performance of using the RA approach for the 25-node topology.

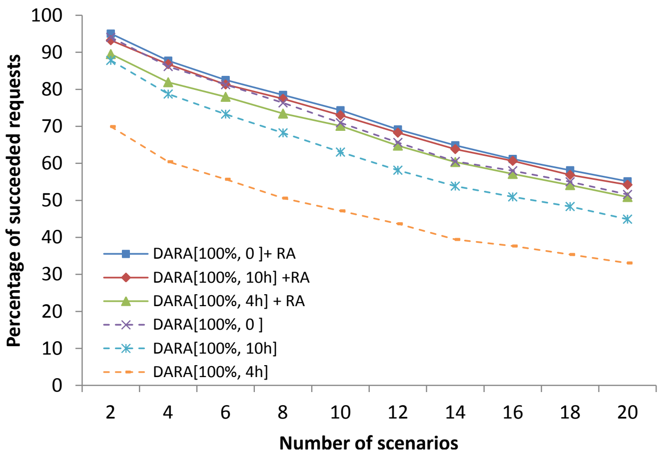


Figure 3.16: Impact of network load and failure rates on the performance of using the RA approach for the 8-node topology.

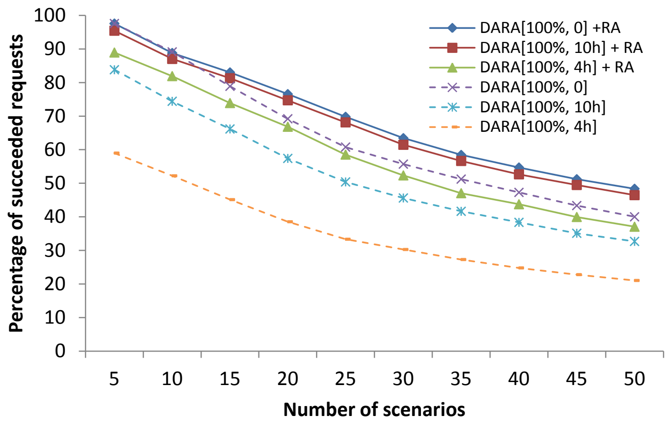
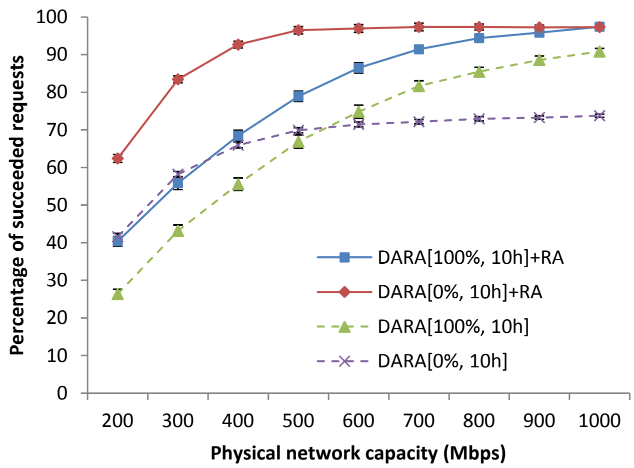


Figure 3.17: Impact of network load and failure rates on the performance of using the RA approach for the 25-node topology.

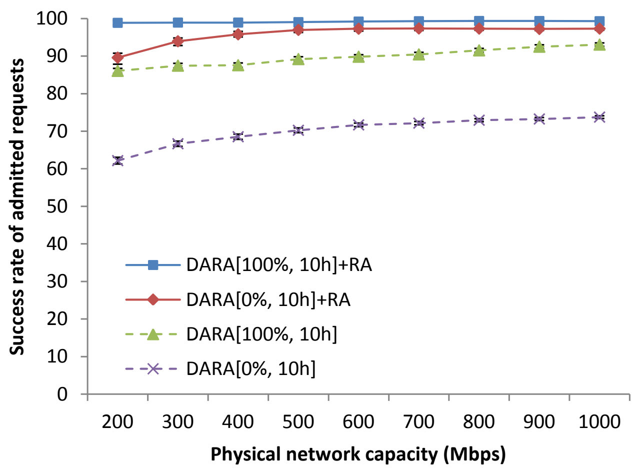
respectively. Figure 3.18b and Figure 3.19b compare the same experiments for the success rate of admitted requests. As can be seen in these figures, the RA approach has noticeably improved the request success rate. In Figure 3.18a, the highest performance in terms of number of succeeded requests is achieved when dropping backup requirements and using the RA approach. Nevertheless, as can be observed from Figure 3.18b with a backup setting of 100% the highest QoS (success of admitted requests) can be achieved. These evaluations also reveal that when there is sufficient capacity in the network (1Gbps), the RA approach is able to achieve the same quality when dropping backup requirements, for 100% of backup demand in terms of succeeded requests. The same trend can be observed in 3.19a and 3.19b for the 25-node topology. For both 0% and 100% of backup demand, the RA approach outperforms the DARA approach.

3.5.3.2 Impact of network load

Figures 3.20a and 3.21a show the percentage of succeeded requests (out of submitted ones) and compare the impact of scenario load and different percentages of backup demands on the performance of proposed approaches. Figures 3.20b and 3.21b show the success rate of admitted requests for the same experiments, using a backup setting of 0% and 100% and 10-hour failure rate. For the smaller topology the number of scenarios increases 2 by 2, up to 20 and for the larger topology, this number increases 5 by 5, up to 50. These figures show that the RA approach achieves the best performance in terms of number of succeeded requests and success rate of admitted requests. Figures 3.20a and 3.21a show that when

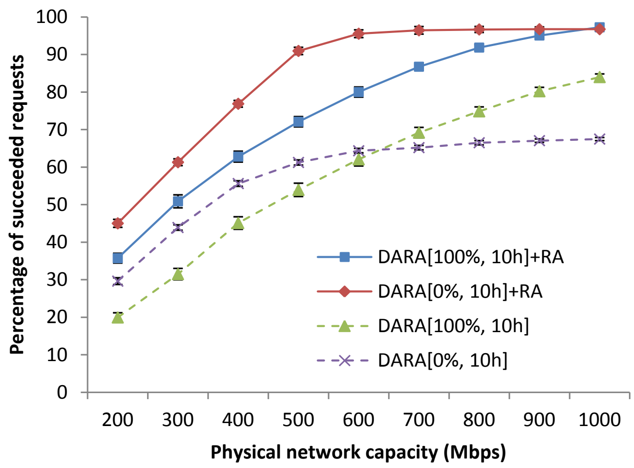


(a) Percentage of succeeded requests.

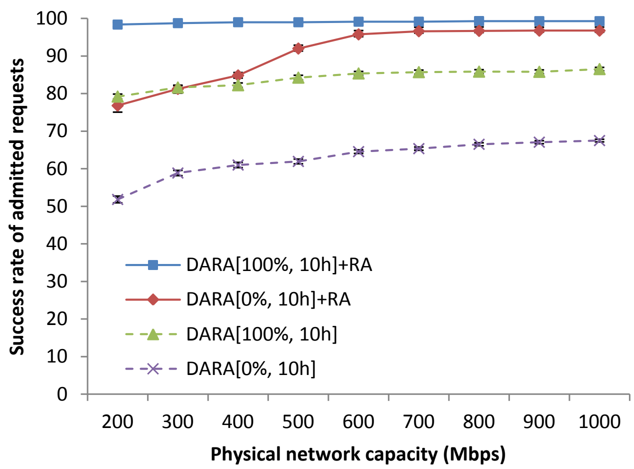


(b) Success rate of admitted requests.

Figure 3.18: Impact of available bandwidth and backup demands on the performance of using the RA approach for the 8-node topology.

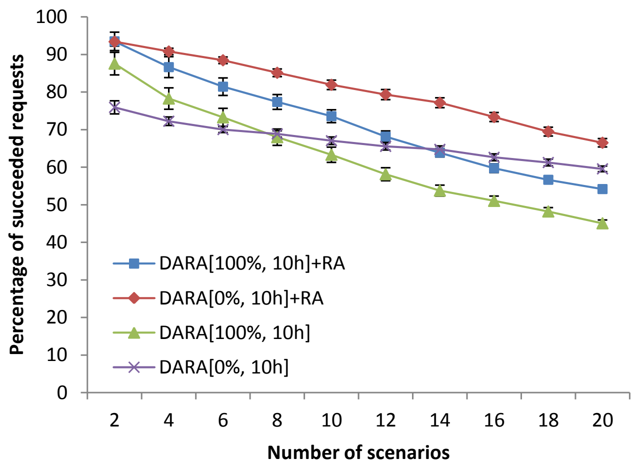


(a) Percentage of succeeded requests.

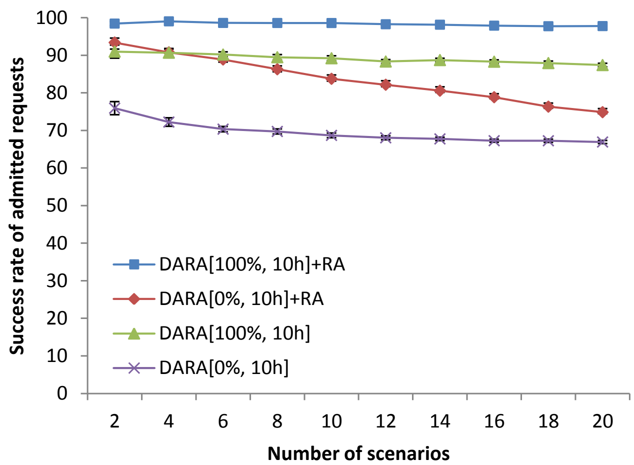


(b) Success rate of admitted requests.

Figure 3.19: Impact of available bandwidth and backup demands on the performance of using the RA approach for the 25-node topology.

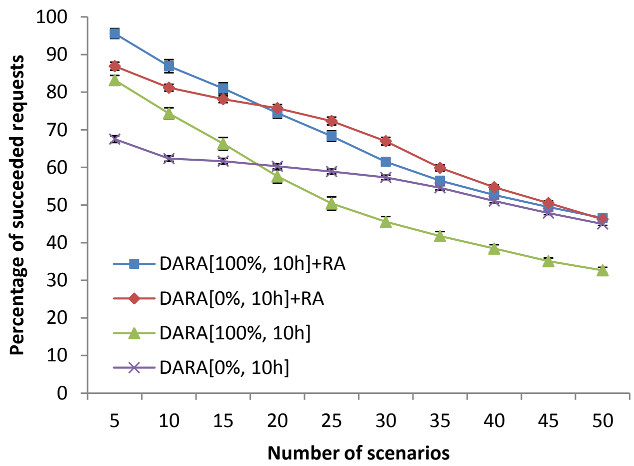


(a) Percentage of succeeded requests.

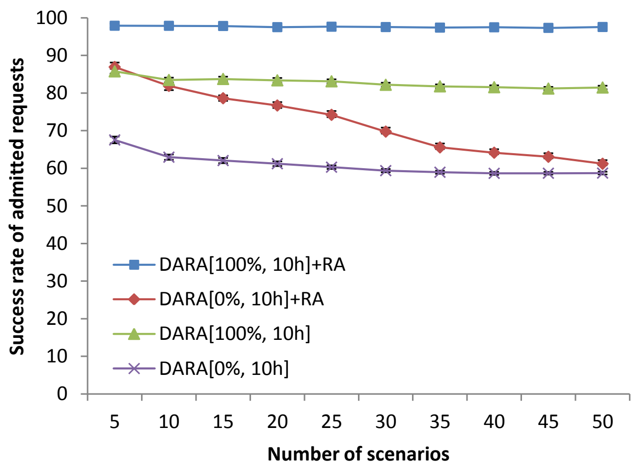


(b) Success rate of admitted requests.

Figure 3.20: Impact of network load and backup demands on the performance of using the RA approach for the 8-node topology.



(a) Percentage of succeeded requests.



(b) Success rate of admitted requests.

Figure 3.21: Impact of network load and backup demands on the performance of using the RA approach for the 25-node topology.

there is sufficient network capacity, no backup requirements outperform the 100% backup demand in the DARA approach. However, Figures 3.20b and 3.21b reveal that using DARA with no protection leads to the worst performance in terms of success of admitted requests. Interestingly, it can also be observed that using the RA approach without protection provides almost always higher success rate than the DARA approach even with 100% of protection.

3.5.3.3 Stressed versus non-stressed network conditions

Figure 3.22 and Figure 3.23 elaborate on the final state of requests in stressed and non-stressed network conditions. The stressed network condition is defined as having 300Mbps bandwidth per link, 2-hour failure rate and default 48-minute repair rate. In non-stressed conditions, the available bandwidth is increased to 800Mbps and the MTTR is reduced to 5 minutes. Figure 3.22a and Figure 3.23a show that in general having more protection in the DARA approach leads to more succeeded requests. In both figures, the percentage of succeeded requests with 100% backup demand is almost 2 times higher than when no backup is provisioned. Figure 3.22b and Figure 3.23b show the impact of deploying the RA approach with the same experiments (note that in these figures the y-axis starts from 90%). These figures reveal crucial advantages of the RA approach. First, comparing to Figures 3.22a and 3.23a, we can see that the RA approach leads to a lower percentage of rejection, up to 1.65% and 4.87% in Figure 3.22b and Figure 3.23b respectively. Second, the percentage of failed requests significantly improves both from lower to higher protection and also compared to the DARA approach. To be precise, with 100% of backup demand in the 8-node topology, the percentage of failed requests is reduced from 1.47% to 0.08% when compared to 0% backup demand in the RA approach, and more importantly, when compared to the DARA approach, this percentage is reduced from 15% to only 0.08%. Third, dropping backup requirements and under non-stressed network conditions, runtime adaptation significantly improves the success rate of reservation system. It can be seen that the number of succeeded requests is highest when dropping backup requirements. However, compared to 100% backup a noticeable number of admitted requests fail.

3.5.4 Impact of different backup demands, varying failure rates

Figure 3.24a and Figure 3.24b compare the success rate of admitted requests in the RA approach to the DARA approach for the 25-node topology. In these experiments, failure rates vary from 1h to 20h and backup demands of 0% up to 100% are assessed. In both figures, regardless of the failure rate, the highest success rate comes with 100% backup. It can also be observed that by employing the RA approach, the number of succeeded requests increases significantly up to 6.77 times with 1-hour failure rate and 100% of backup demand. The same trend has been

Table 3.2: Execution time per algorithm invocation (ms) of main algorithms of the RA approach. Failure rate is 2h.

	8-node topology			25-node topology		
execution time (ms)	**AVG**	**std error**	**#invocation/24h**	**AVG**	**std error**	**#invocation/24h**
UpdateRequestsInfo	0.005	0.001	498.38	0.025	0.006	1578.4
AO	0.28	0.01	475.38	2.61	0.07	1555.4
DARA	51.96	2.05	23	551.4	15.9	23

observed for the smaller topology in which the RA approach can provide up to 5.3 times higher success rates.

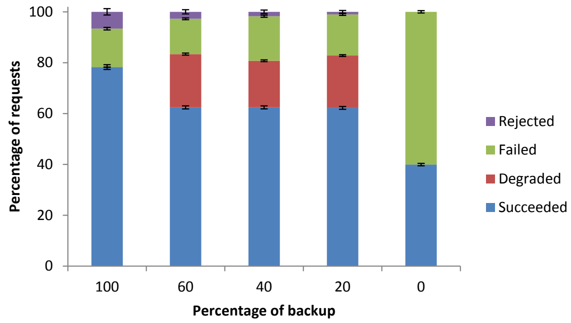
3.5.5 Evaluation of execution times

Figure 3.25 and Figure 3.26 compare the computational time of the DARA algorithm and the proposed RA approach, using 100% backup capacity. The execution time of the RA approach is the sum of all invocations of the periodic update and the periodic adaptation algorithms and the execution time of the DARA approach is the sum of all re-scheduling invocations whenever a new scenario enters to the reservation system. Our results indicate that deploying the RA approach increases the execution time by 2.75 and 2.12 times on average in the 8-node and 25-node topologies respectively.

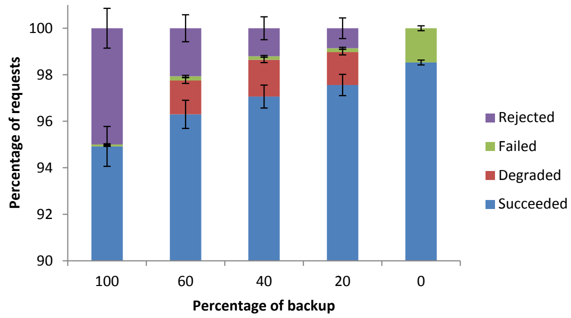
The number of invocations and the average execution time of a single invocation of main algorithms in the RA approach is shown in Table 3.2. The number of scenario is 20 in 8-node and 50 in 25-node topology and 2h failure rate is used. This evaluation shows that the AO algorithm, with 0.28 ms in the smaller and 2.61 ms in the larger topology, is fast enough to immediately reconfigure the network and react to sudden changes.

3.6 Conclusions

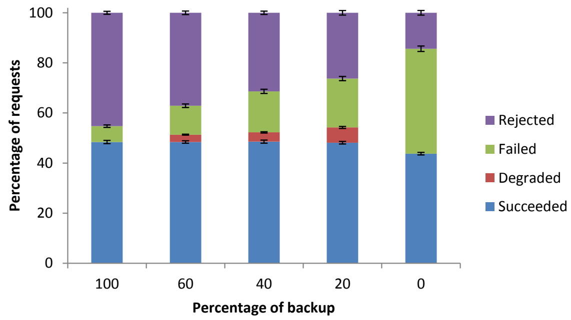
In order to deliver reliable data transfers, we have previously proposed a resilient advance reservation approach. Offering guaranteed video delivery in media production networks is of prime importance, however, using redundancy imposes significant performance overheads and extra costs. In this chapter, we proposed a dual optimization approach for exploiting underutilized network capacities to transfer more data than what has been scheduled as long as no failures are detected. This chapter deals with the design, development and evaluation of the proposed approach in which a constant monitoring, adaptation and re-optimization is being applied during runtime, taking into account potential failures. The main objective is to mitigate the side-effect of redundant allocations and dynamically reconfigure transmissions in response to sudden changes in network conditions.



(a) The DARA approach in non-stressed network conditions.

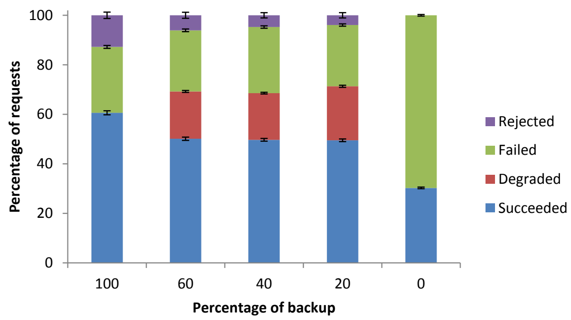


(b) Using the RA approach in non-stressed network conditions.

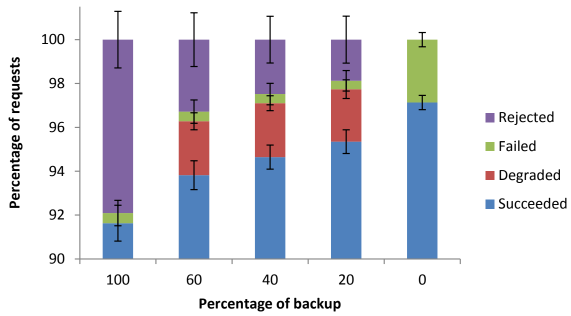


(c) Using the RA approach in stressed network conditions.

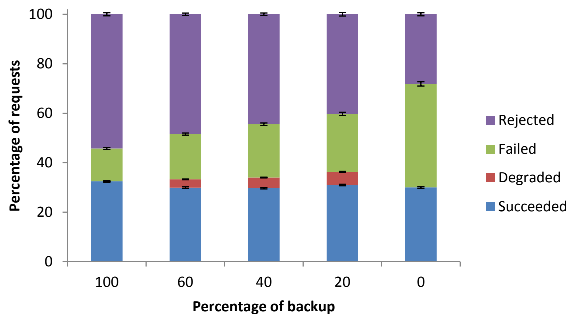
Figure 3.22: Final state of requests in stressed and non-stressed 8-node topology with a failure rate of 2h.



(a) The DARA approach in non-stressed network conditions.

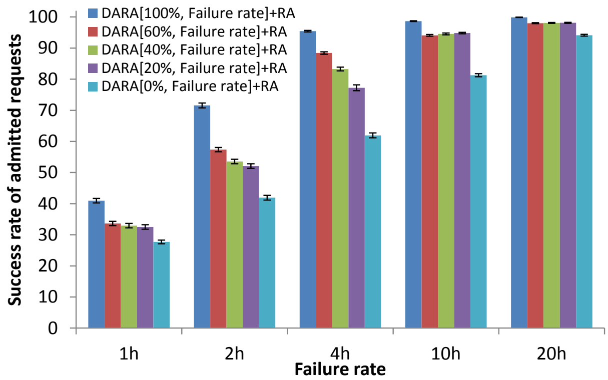


(b) Using the RA approach in non-stressed network conditions.

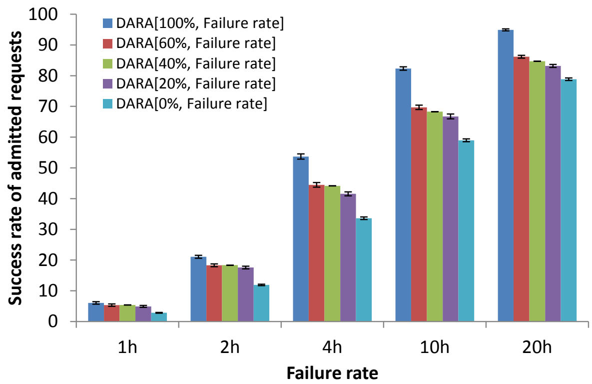


(c) Using the RA approach in stressed network conditions.

Figure 3.23: Final state of requests in stressed and non-stressed 25-node topology with a failure rate of 2h.



(a) Success rate of using the RA approach.



(b) Success rate of the DARA approach.

Figure 3.24: Comparing the success rate of the DARA and the RA approaches with different backup demands for the 25-node topology.

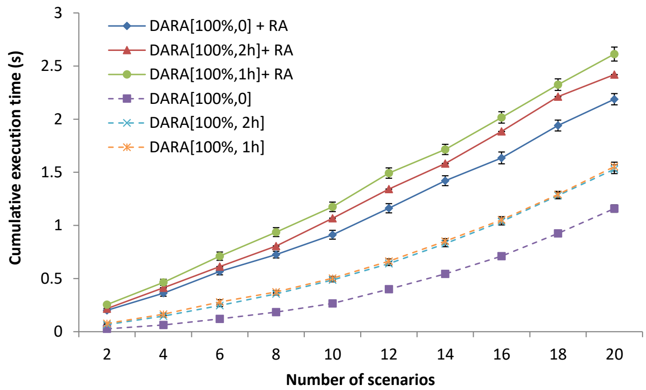


Figure 3.25: Evaluation of execution times for the 8-node topology.

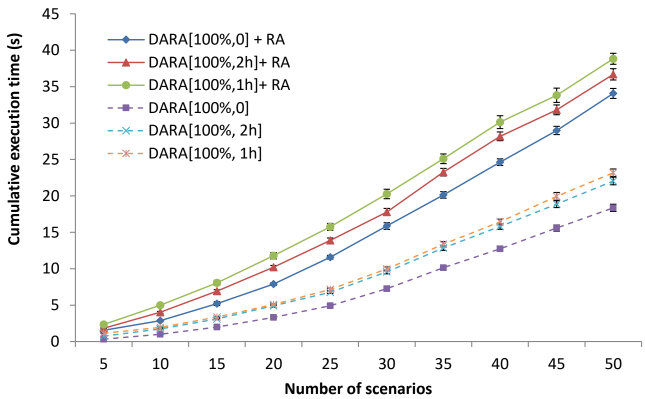


Figure 3.26: Evaluation of execution times for the 25-node topology.

The experimental results showed that our approach works efficiently both in stable and failure-prone networks. Deploying this approach will noticeably increase the performance of the advance reservation systems by increasing the number of succeeded requests and with computational time of less than 3ms for all evaluated cases, our solution is fast enough to react immediately and re-configure the network in response to sudden changes.

Acknowledgment

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government - department EWI. The research leading to these results has been performed within the context of ICON MECaNO, co-funded by iMinds, a digital research institute founded by the Flemish Government under grant agreement no. 130646.

References

[1] N. Charbonneau and V. M. Vokkarane. *A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks*. IEEE Communications Surveys & Tutorials, 14(4):1037–1064, 2012.

[2] K. Rajah, S. Ranka, and Y. Xia. *Advance Reservations and Scheduling for Bulk Transfers in Research Networks*. IEEE Trans. Parallel Distrib. Syst., 20(11):1682–1697, November 2009. doi:10.1109/TPDS.2008.250.

[3] *ICON MECaNO project*. http://www.iminds.be/en/projects/mecano/, 2014 - 2016. Accessed: 2017-02-10.

[4] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Algorithms for advance bandwidth reservation in media production networks*. In IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 183–190, May 2015. doi:10.1109/INM.2015.7140291.

[5] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Deadline-aware advance reservation scheduling algorithms for media production networks*. Computer Communications, 77:26–40, 2016. doi:10.1016/j.comcom.2015.10.016.

[6] S. Sahhaf, M. Barshan, W. Tavernier, H. Moens, D. Colle, and M. Pickavet. *Resilient algorithms for advance bandwidth reservation in media production networks*. In International Conference on the Design of Reliable Communication Networks (DRCN), pages 130–137. IEEE, 2016.

[7] B. G. Józsa and D. Orincsay. *Shared backup path optimization in telecommunication networks*. In International Conference on the Design of Reliable Communication Networks (DRCN), pages 251–257, 2001.

[8] J. T. Haahr, T. Stidsen, and M. Zachariasen. *Heuristic methods for shared backup path protection planning*. In 4th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pages 712–718. IEEE, 2012.

[9] W. Depoorter, K. Vanmechelen, and J. Broeckhove. *Advance reservation, co-allocation and pricing of network and computational resources in grids*. Future Generation Computer Systems, 41:1–15, 2014.

[10] K. Bochenina, N. Butakov, and A. Boukhanovsky. *Static scheduling of multiple workflows with soft deadlines in non-dedicated heterogeneous environments*. Future Generation Computer Systems, 55:51–61, 2016.

[11] H. Bai, K. Shaban, M. Khodeir, F. Gu, J. Crichigno, S. Khan, and N. Ghani. *Overlay network scheduling design*. Computer Communications, 82:28–38, 2016.

[12] C. Guok, E. N. Engineer, and D. Robertson. *ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)*. In Internet2 Joint Techs Workshop, Salt Lake City, Utah, 2005.

[13] B. Gibbard, D. Katramatos, and D. Yu. *TeraPaths: end-to-end network path QoS configuration using cross-domain reservation negotiation*. In 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS), pages 1–9. IEEE, 2006.

[14] L. Chunlin, Z. J. Xiu, and L. Layuan. *Resource scheduling with conflicting objectives in grid environments: Model and evaluation*. Journal of Network and Computer Applications., 32(3):760–769, 2009.

[15] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. *StorNet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers*. In IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 121–126. IEEE, 2011.

[16] J. Kuri, N. Puech, M. Gagnaire, and E. Dotaro. *Routing foreseeable lightpath demands using a tabu search meta-heuristic*. In IEEE Global Telecommunications Conference (GLOBECOM'02), volume 3, pages 2803–2807. IEEE, 2002.

[17] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, and R. Douville. *Routing and wavelength assignment of scheduled lightpath demands*. IEEE Journal on Selected Areas in Communications, 21(8):1231–1240, 2003.

[18] J. Zheng and H. T. Mouftah. *Supporting advance reservations in wavelength-routed WDM networks*. In Tenth International Conference on Computer Communications and Networks, pages 594–597. IEEE, 2001.

[19] J. Zheng and H. T. Mouftah. *Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks*. In IEEE International Conference on Communications (ICC), volume 5, pages 2722–2726. IEEE, 2002.

[20] C. Xie, H. Alazemi, and N. Ghani. *Rerouting in advance reservation networks*. Computer Communications, 35(12):1411–1421, 2012.

[21] M. Balman, E. Chaniotakis, A. Shoshani, and A. Sim. *A flexible reservation algorithm for advance network provisioning*. In 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pages 1–11. IEEE, 2010.

[22] M. Balman. *Advance resource provisioning in bulk data scheduling*. In IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pages 984–992. IEEE, 2013.

[23] T. Watanabe, T. Omizo, T. Akiyama, and K. Iida. *ResilientFlow: Deployments of distributed control channel maintenance modules to recover SDN from unexpected failures*. In 11th International Conference on the Design of Reliable Communication Networks (DRCN), pages 211–218. IEEE, 2015.

[24] S. Tanwir, L. Battestilli, H. Perros, and G. Karmous-Edwards. *Dynamic scheduling of network resources with advance reservations in optical grids*. International Journal of Network Management, 18(2):79–106, 2008.

[25] L.-O. Burchard, H.-U. Heiss, B. Linnert, J. Schneider, and C. A. De Rose. *VRM: a failure-aware grid resource management system*. International journal of high performance computing and networking, 5(4):215–226, 2008.

[26] P. Latchoumy, P. Khader, and S. Abdul. *Job Scheduling with Failure Prevention Strategies in Grid Computing Environment*. International Journal Of Research In Advance Technology In Engineering, 1, 2013.

[27] B. Nazir, K. Qureshi, and P. Manuel. *Replication based fault tolerant job scheduling strategy for economy driven grid*. The Journal of Supercomputing, 62(2):855–873, 2012.

[28] P. Chołda and P. Jaglarz. *Optimization/simulation-based risk mitigation in resilient green communication networks*. Journal of Network and Computer Applications, 59:134–157, 2016.

[29] C. Cavdar, M. Tornatore, F. Buzluca, and B. Mukherjee. *Dynamic Scheduling of Survivable Connections with Delay Tolerance in WDM Networks*. In IEEE INFOCOM Workshops, pages 1–6, 2009. doi:10.1109/INFCOMW.2009.5072134.

[30] B. Wang and T. Li. *Survivable scheduled service provisioning in WDM optical networks with iterative routing*. Optical Switching and Networking, 7(1):28–38, 2010.

[31] A. Jaekel, Y. Chen, and A. Bari. *Stable logical topologies for survivable traffic grooming of scheduled demands*. Journal of Optical Communications and Networking, 2(10):793–802, 2010.

[32] M. Barshan, H. Moens, B. Volckaert, and F. De Turck. *Design of a dynamic adaptive reservation system in media production networks*. In IEEE/IFIP Network Operations and Management Symposium (NOMS), pages 1149–1152. IEEE, 2016.

[33] M. Barshan, H. Moens, and B. Volckaert. *Dynamic adaptive advance bandwidth reservation in media production networks*. In IEEE NetSoft Conference and Workshops (NetSoft), pages 58–62. IEEE, 2016.

[34] T. Cormen. *Introduction to Algorithms*. MIT Press, 2009.

[35] J. Edmonds and R. M. Karp. *Theoretical improvements in algorithmic efficiency for network flow problems*. Journal of the ACM (JACM), 19(2):248–264, 1972.

[36] W. Smith, I. Foster, and V. Taylor. *Scheduling with advanced reservations*. In 14th International Parallel and Distributed Processing Symposium (IPDPS), pages 127–132. IEEE, 2000.

[37] C. Ernemann, V. Hamscher, and R. Yahyapour. *Economic scheduling in grid computing*. In Job Scheduling Strategies for Parallel Processing, pages 128–152. Springer, 2002.

[38] J. Cao and F. Zimmermann. *Queue scheduling and advance reservations with COSY*. In 18th International Parallel and Distributed Processing Symposium, page 63. IEEE, 2004.

[39] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. *Mason: A multiagent simulation environment*. Simulation, 81(7):517–527, 2005.

[40] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. *The internet topology zoo*. IEEE Journal on Selected Areas in Communications, 29(9):1765–1775, 2011.

[41] M. M. Alam Khan. *Multi-Path Link Embedding for Survivability in Virtual Networks*. Master's thesis, University of Waterloo, 2015.

4

A Flexible, Reliable and Adaptive Timeslot-based Advance Bandwidth Reservation Mechanism for Media-centric Networks

Advance reservation systems can be implemented either based on flexible or fixed timeslot sizes. In Chapter 2, we proposed optimal and near-optimal timeslot-based advance bandwidth reservation solutions based on fixed size timeslots, taking into account the specific characteristics of media transfers. The resilient version of this solution and the complementary Runtime Adaptation (RA) approach were presented in Chapter 3. In this chapter, we have first designed, implemented and evaluated the flexible timeslot-based advance bandwidth reservation system. We have then optimized the resilient advance bandwidth reservation approach, proposed in Chapter 3. This optimization results in greater network utilization and higher request admittance ratio, up to 9.2%. In addition, the optimized resilient solution has been combined with the RA approach. Quality and complexity of the proposed approach has been extensively compared with that of a fixed timeslot algorithm. Our simulation studies reveal that the request admittance ratio is up to 3.6% higher using flexible timeslots in combination with the RA approach and in a failure-free network, the execution time of this approach is up to 17.5 times lower, compared to the approaches with fixed timeslot sizes.

⋆ ⋆ ⋆

M. Barshan, H. Moens, B. Volckaert and F. De Turck

Submitted to International Journal of Network Management, Jun. 2017.

4.1 Introduction

Nowadays, various media-centric industries are faced with a rising need for decentralized collaborations over shared substrate networks. As multimedia services have high bandwidth requirements, it becomes necessary for such networks to provide a quality-of-service (QoS) guarantees. In order to manage the available bandwidth and to provide a guaranteed QoS, bandwidth reservation and scheduling techniques must be applied. Bandwidth scheduling refers to bandwidth allocations with flexible user options on timing and bandwidth requirements in both Immediate Reservation (IR) and Advance Reservation (AR) disciplines. While the start time of the data transmission is assumed to be immediate in IR, the principle behind AR relies on reserving the resource before they are required for a specific transfer. An AR demand typically specifies information about the start of the data transmission and / or a deadline, as well as the bandwidth requirement of the transmission.

In AR, the knowledge of future network traffic and demand allows the network to make better decisions compared to immediate reservation requests, especially for large demands which are difficult to allocate. Since AR requests have been previously reserved, they have higher priority over immediate transfers. This results in an efficient resource management within the network and better quality of service for users. AR has various important applications for wide-area networks [1] and there exists a number of applications where AR is more appropriate than IR. For example, offsite backups can be scheduled overnight using advance reservation. Various real-time streaming applications that require large amounts of bandwidth, such as IPTV, video conferencing, and video on demand, are also well-suited for advance reservations. These applications can benefit from advance reservation as real-time online conferences are typically being scheduled for specific times in advance and require guaranteed bandwidth.

The problem of advance reservation for multimedia applications with different characteristics and requirements, has not gained enough attention in literature. As such, this chapter focuses on timeslot-based advance bandwidth reservation for media delivery services with 3 main features: flexibility, reliability and adaptability.

Flexibility: To manage the time domain of advance reservations, fixed or flexible timeslot-based solutions are introduced. The computational complexity of fixed size approaches highly depends on timeslot granularity, whereas with flexible time intervals this rather depends on the number of requests. The impact of this flexibility is investigated in this work and quality and complexity of flexible timeslots is compared to equivalent fixed size solutions.

Reliability: In media delivery services, reliability of the data transport is of great importance, e.g. network failures should be tackled to enable reliable transmission without any loss in QoS. To enable a quick response to sudden changes such as failures in the network, we rely on a protection mechanism, which reserves backup paths in advance, before any failure happens in the network. The proposed scheme aims at minimizing the resource usage of backups, while guaranteeing 100% recovery against any single link failure.

Adaptability: Here the main idea is to make use of *wasted* network capacity, mainly caused by the redundant failure protection reservations. Adaptability results in an increase in network utilization and admittance ratio, while the reliability of reservations remains guaranteed. To tackle unforeseen failures in the network, the proposed approach has to immediately react to any failure / repair detection while re-utilizing unused reservations. In order to provide an adaptive resilient advance bandwidth reservation system, the generated resilient schedule is continually updated over time in order to be capable of dynamically adapting the reservations to changing conditions.

To offer predictable complexity, easier hardware implementation and systematic reconfiguration of network devices, we have previously designed and implemented several timeslot-based algorithms using fixed time slots and taking into account the characteristics of requests in media-centric networks [2–4]. The resilient version of this solution has been proposed in [5] and adaptability has been added to the above-mentioned fixed size timeslot-based approaches in [6], by introducing the runtime adaptation (RA) approach.

This work is an extension of our previous work [7] and [8]. In [7], we have theoretically analyzed the benefits and drawbacks of using flexible timeslots, finding that a flexible approach is highly beneficial when dealing with bursty traffic conditions in a low-demand network with long-term downtimes, as it has the potential to significantly reduce the number of timeslots needed, resulting in execution speed improvements. In [8], our previously designed static advance reservation algorithm has been extended to add support for flexible timeslot sizes.

In this work, we however focus on dynamic version on the advance reservation algorithm, referred to as DARA, and optimize the resilient dynamic advance reservation algorithm in combination with flexible timeslots and the RA approach to offer a flexible, reliable and adaptable advance bandwidth reservation system. We discuss the impact of the RA approach in resilient and flexible timeslot-based advance bandwidth reservation systems.

The remainder of this chapter is structured as follows. In Section 4.2, we discuss related work. Section 4.3, provides brief information about advance reservation for media delivery services. In Section 4.4, the proposed solution is explained in detail. A general overview of our proposed advance bandwidth reservation architecture is discussed in Section 4.5. The heuristic-based advance reservation scheduling algorithms are described in Section 4.6. Sections 4.7 and 4.8 provide the simulation outline and simulation results respectively and finally, Section 4.9 concludes the chapter.

4.2 Related work

The advance reservation scheduling problem has been well studied in literature. While some have focused on rescheduling [9–11] and multi domain reservation [12], others specifically take into account real-world deployments [13–16], and WDM optical networks [1]. Nevertheless, only two advance reservation algorithms [17, 18] support elastic reservations, and both considered fixed start time for the requests [1], while we consider flexible or unspecified start times. Advance bandwidth reservation for on-demand and flexible data transfer in scientific applications is investigated in [19]. However, they purely focus on data transfers, not streaming requests, the routing mechanism is based on single-path in contrary to our multi-path approach and dependency among different transfers is ignored.

There exist a few works in the literature focusing on the flexible advance reservation approaches [1]. In [20], a flexible single-path resource reservation is proposed for a set of files between multiple end-sites. However, streaming requests and interdependencies among requests are ignored. The authors in [21], concluded that for a guaranteed network resource availability and to increase network utilization and user satisfaction, bandwidth has to be reserved in advance and using flexible time windows. Nonetheless, interdependence and resilience against failures are not taken into account. Flexible advance reservation for cloud-based resources has been investigated by [22–24]. A more recent work, [25], focuses on flexible bandwidth reservation between VMs of cloud applications, for which it is hard to estimate the exact needed capacity. They propose a solution to elastically determine the bandwidth between VMs in order to reduce job execution times.

Resilient reservations can be achieved either through restoration or protection techniques [26, 27]. In protection approaches, backup resources are reserved in advance before any failure happens in the network, while in case of restoration backup resources are selected upon failure detection. The former results in larger resource consumption but the recovery time is generally quicker. In [28], the authors propose a restoration technique to deal with link failures. In their work, the active requests and the scheduled future requests which are affected by a failure are restored. In [29, 30], optimal ILP-based solutions were proposed to provide shared and dedicated path protection. Authors in [31, 32] also provide resiliency through Shared Backup Path Protection (SBPP).

This chapter is in line with our previous works on media-centric network bandwidth reservation approaches, conducted within the context of MECaNO project [33]. We have first proposed optimal [2] and near-optimal advance bandwidth reservation algorithms [3], focusing on a media production use case. These proposed approaches were based on a fixed size timeslot-based approach which is reported to be inefficient when the number of requests is limited [1]. This was the motivation of our recent work [7, 8] in which a theoretical and analytical comparison between fixed size and flexible reservations is drawn respectively, to investigate which approach would be more appropriate for media related environments.

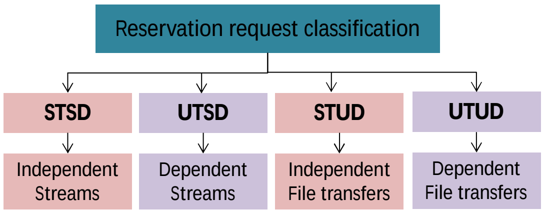
The work presented in this chapter differs from our recent work. In this chapter, first, the impact of flexible timeslots is investigated by designing a dynamic flexible timeslot-based advance reservation algorithm. Second, the resilient approach is optimized to enhance network utilization. Third, the resilient flexible solution is combined with the RA approach and quality and complexity of this combination is compared to an equivalent fixed size solutions. In addition, in order to make a comprehensive comparison a baseline algorithm is presented. This chapter is the final work on the MECaNO project, which outlines a comprehensive architecture for the advance bandwidth reservation of collaborative media-centric networks and presents an integrated evaluation to thoroughly compare different aspects of our proposed approaches.

4.3 Advance Reservation for media delivery services

The advance bandwidth reservation framework is responsible for admission control, scheduling of submitted requests and reserving the required amount of bandwidth resources for all admitted requests, according to the agreed SLA. The output of the scheduling algorithms takes the form of a set of bandwidth reservations associated with each flow over time. This information can be transferred to the network controllers, used to configure the switches in the network. The controllers keep track of the temporal aspects of the policies, adjusting configurations whenever needed.

4.3.1 Type of reservation requests

In collaborative media delivery scenarios, reservation requests can either be streams or file transfers and multiple reservation requests may depend on each other, meaning that one request can only start when other requests that are dealt with, have been finished. This interdependency is an important characteristic of collaboration



Type	Characteristics		
	Streaming request		
	Start time	**End time**	**Duration**
STSD	Specified	Specified	Fixed
UTSD	Unspecified (Co-dependent)	Unspecified (Co-dependent)	Fixed
	File transfer request		
	Ready for transfer	**Deadline**	**Duration**
STUD	Specified	Specified	Flexible
UTUD	Unspecified (Co-dependent)	Unspecified (Co-dependent)	Flexible
	Specified	Unspecified (Co-dependent)	Flexible
	Unspecified (Co-dependent)	Specified	Flexible

Figure 4.1: Types of advance reservation requests.

scenarios, which has not gained enough attention in other advance reservation approaches. As stated by [1, 34, 35] advance reservation requests are classified into four individual categories, all are taken into account in this chapter and shown in Figure 4.1. We assume that for file transfers, volume and for streaming requests duration is always known.

4.3.1.1 STSD (Specified Time, Specified Duration) and UTSD (Unspecified Time, Specified Duration) requests

STSD advance reservation requests specify both the start time of the request and its duration. For UTSD requests, start time is unspecified, but duration is known. STSD and UTSD requests are related to streams, as we assume that duration of streaming requests are always known. Dependent streaming requests that the start times have codependency on the other requests are in the UTSD category and independent streams belongs to STSD.

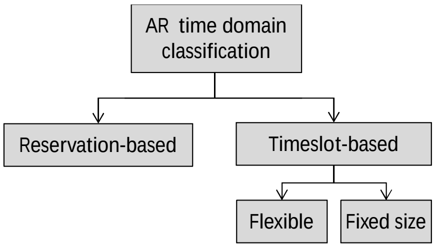


Figure 4.2: Time domain classification of AR approaches.

4.3.1.2 STUD (Specified Time, Unspecified Duration) and UTUD (Unspecified Time, Unspecified Duration) requests

These classes are related to file transfers. The reservation timing of a file-based request can be fluid, as long as this timing fits between the time when data is ready to be transferred and the delivery deadline. Independent and dependent file based requests are classified as flexible UTUD and flexible STUD respectively. The file-based STUD advance reservation requests specify the times when data is ready and the deadlines, but durations are unspecified. However, for UTUD requests, the time when data is ready and/or the deadline depends on other requests (co-dependency) and duration of these requests are unspecified.

4.3.2 Time domain classification in AR approaches

As has been shown in Figure 4.2, two mechanisms have been introduced for time management of the AR systems, the reservation-based and timeslot-based approaches [36]. The reservation-based approaches show high computational complexity when processing a large number of reservation requests. Alternatively, timeslot-based solutions are introduced as an efficient mechanism to cope with this complexity [1, 36]. Timeslot-based approaches discretize the time span into a set of timeslots. Each timeslot maintains aggregated information about resource usage and network residual capacity. Timeslot-based solutions can be static (i.e. fixed) or dynamic (i.e. flexible). In a static timeslot-based classification, the timespan is broken into a fixed number of predetermined-length timeslots which makes it easy to implement. The complexity of the approach highly depends on the granularity of timeslot sizes and the amount of network state information is to some extent independent of the number of requests. Although the majority of timeslot-based approaches in literature have followed a static approach, it is inefficient for advance reservation systems with a small number of reservation requests [1].

However, in the dynamic timeslot approach, use of flexible timeslots increases the complexity of timeslot-based advance reservation approaches in two ways. first, the flexible timeslots are typically started with any request start time and end with either the start of a new request or the earliest end time of current requests. As such, the complexity of the scheduling highly depends on the number of requests and in the worst case the number of time slots is twice the number of requests. The number of timeslots is a factor which directly increases the computational overhead of timeslot-based advance reservation systems. Second, in the dynamic timeslot solution, upon entrance of a new request, duration and number of timeslots are repeatedly being changed during the re-scheduling process, making it unpredictable and harder to manage with highly frequent request arrival rate.

4.3.3 The resilient AR scheduling approach

In addition to efficient bandwidth scheduling management, reliability of the data transport is important. Reliable AR systems can be deployed either through restoration or protection failure recovery mechanisms.

In order to have a quick response to sudden changes such as failures in the network, we have proposed the resilient advance bandwidth reservation algorithm in [5]. The proposed solution uses a protection mechanism which finds backup paths for connections in advance, before the occurrence of any failure and exploits Shared Backup Path Protection (SBPP), capable of covering single link failures. The objective is to minimize resource usage by the protection paths while full recovery is guaranteed against any single link failure in the network.

In the resilient approach, we have made a tradeoff between reliability and resource usage in 2 ways: first, deploying SBPP significantly reduces the bandwidth requirements for backup purposes, since the backups have to fulfill the maximum bandwidth allocated on the links of the primary paths. This means that to provide full protection (100% backup), there is no need to allocate the exact amount of bandwidth as the primary paths. Second, the percentage of redundancy is defined for each individual request based on an input parameter provided by the customer, to influence the importance of reliability for each individual connection. This way, the higher priority requests can be protected while the ones with lower priority can remain unprotected.

4.3.4 Runtime adaptation (RA) approach

As a first provisional stage to offering a robust reservation system, deploying protection mechanisms ensures that the reservations remain valid when the system is in operation. To maximally utilize the network, and to ensure a quick response in a dynamic network environment, constant monitoring and optimization is needed. We have previously proposed an efficient dual optimization approach, referred to as the Runtime Adaptation (RA) approach, consisting of two consecutive processes [6]. In the RA approach, first, a schedule is produced by a resilient advance reservation algorithm. Then, the generated schedule is continually updated over time in order to be capable of dynamically adapting the schedule to changing conditions. Moreover, to mitigate the side effects of provisioned reliability, this approach uses the inter-connecting network links' leftover capacity, resulting in increased performance. In unreliable networks, as soon as any failure is detected, the ongoing network transfers are adapted according to the current state of the network. This leads to better utilization of substrate network resources, higher success rate and rapid reaction to sudden changes when the network is in operation. Concretely, regardless of the fixed or flexible advance reservation scheduling, the RA approach follows two sequential phases in every timeslot: 1) the periodic update and 2) the periodic adaptation.

4.3.4.1 First phase: Periodic update

During the periodic update, first the current status of the network and transfers are monitored. The monitoring system keeps track of demand, time and last allocated bandwidth for all requests including the last monitored demand, the last monitored time and the last allocated bandwidth. Then the DARA algorithm is invoked. Since periodic adaptation algorithms make use of idle network capacities and real transfers potentially run ahead of schedule, this invocation is necessary to take into account the extra transfers and perform re-scheduling for the residual demands. DARA updates the entire schedule and this new information is set in the global state manager. The global state manager contains all the information about the schedule, the network and request reservations, connections, demands, deadlines, etc. Then the next timeslot reservations are derived from the advance reservation schedule and are set as advance-scheduled requests. The list of advance-scheduled requests contains all requests which have been scheduled to be transferred from now on. Take into account that potentially there are other requests which could be started, but have been postponed due to bandwidth constraints. These requests are kept in a waiting list and used in the periodic adaptation phase.

4.3.4.2 Second phase: Periodic adaptation

The Adaptive Optimization (AO) algorithm is the main algorithm in the periodic adaptation phase. The AO algorithm is triggered several times, i.e. whenever a file-based request starts / finishes transferring and in case of any link failure or link repair. The first invocation of this algorithm is before the start of the next timeslot. Based on this algorithm, the scheduling of the next timeslot is analyzed and modified to make use of idle bandwidth capacities. To achieve this, the advance-scheduled requests are retrieved and the reservations for backups and streams are ignored (because streams may not always be active and can be resumed / played-back multiple times throughout the reserved period). This gives us a network in which only the primary reservations occupy the network capacities and the remainder of network capacity is continuously re-utilized. In case of failures, the AO is triggered to restore the backup reservations. In case of streaming request activations, the streaming reservations are prioritized and transferred.

4.4 Problem description

4.4.1 Flexible approach

In literature, the static solution with fixed size timeslots is followed by the majority of timeslot-based approaches. In this section, we distinguish 5 factors which restrict the capabilities of fixed size timeslot-based advance bandwidth reservation approaches. We discuss how the flexible timeslots contribute to eliminating these factors. The restrictions are as follows:

Reservation request characteristics: The reservation request types can either be streams or file transfers. The file can be transferred whenever possible from the time when the file is ready to be transferred until its deadline. The start of a reservation for a file has to be restricted to the beginning of the next timeslot and the start of the timeslot in which the request deadline fits. For file-based requests, the file transfer volume is the determining factor. The allocated bandwidth can vary from one timeslot to another. There is no restriction for the amount of bandwidth allocation as long as its deadline is met and enough reservations have been made to fully transfer the file. In the fixed timeslot-based approach, these restrictions for file-based requests imply that the file has a tighter time opportunity for transmission and therefore the probability of timely transfer is decreased.

In contrast to file transfers, the allocated bandwidth for streams must equal to their required bandwidth demand, from the start time to the end time, as their demand is constant for the entire reservation period. As such, for streams the reservation has to be made from the start of the timeslot in which the start time of request fits, until the end of the timeslot to which the request's end time belongs. It should be noted that reservations based on a fixed size approach for streaming requests lead to a waste of resources due to making unused reservations.

Another point is that the fixed size of timeslots is more restrictive when there are dependencies among different transfers, meaning that one request can only start when other requests on which this request depends, have been finished. This implies that even a small portion of two interdependent requests can not be accommodated in one timeslot. In high-bandwidth networks with plenty of unused capacity, a chain of interdependent request may remain longer in the schedule compared to the flexible approach, which can be problematic for future requests. This impacts the request admittance ratio.

Contrary to the fixed size approach, the use of flexible time windows can mitigate these issues for both streaming and file-based requests. Regardless of the type of request, the start and the end of time windows can be tuned up to the start and end time of each request. The interdependent requests can also be scheduled as soon as the dependencies have been eliminated without having to wait for the start of the next timeslot.

Delay prior to request processing: Predefined timeslot sizes imply that each new request arrival has to wait until the start of the next timeslot to be processed. This waiting time equals the time gap between the request start time and the start of the next timeslot. Although more fine-grained timeslot sizes can shorten the delay, it can be completely eliminated by deploying flexible time intervals.

Optimized timeslot size: In the fixed size advance reservation approaches, timeslot size is of great importance, because it has a high impact on the complexity and quality of the advance bandwidth reservation system. In the fixed size approaches, it is not trivial to find a good value for this. Nevertheless, this is a non-issue with flexible timeslots.

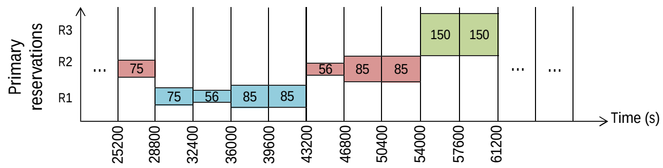
Unnecessary periodic computations for long transfers: The periodic nature of the fixed timeslot-based solutions leads to unnecessary periodic computations for long-term streaming requests and large files. In the fixed size advance bandwidth reservation approaches, the residual demand of ongoing requests is periodically updated at each timeslot, and new and updated requests are periodically reallocated together. This issue causes unneeded computational overhead, which becomes worse with very fine-grained timeslots. Again, this is not an issue in the flexible approach.

High computational complexity for long-term schedules: Another problem with the fixed timeslot-based approaches, specially with fine-grained timeslots, is that the computational complexity of these approaches mostly depends on the number of timeslots, making it impractical or at least unrecommended for long-term schedules.

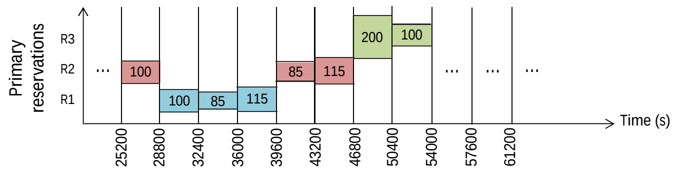
4.4.2 Optimized resilient approach

In the resilient approach, first the primary paths for a given request are determined and then disjoint backup paths are found corresponding to these primary paths [5]. For file-based requests, if this backup limit can not be found, it backtracks to the initial state and retries the bandwidth allocation with 50% of the primary bandwidth demand. This is repeated until both primary and backup demands are fulfilled. If both primary and backup demands are not accommodated by the file's deadline, the entire scenario to which the file belongs, is rejected.

We have found that halving the request demand does not always lead to an optimal solution because we may miss the opportunity to transfer a higher volume of the file and the network capacity may not be fully utilized if other concurrent requests can not make use of it. As such, to make better use of leftover capacities



(a) Original fixed-size timeslot-based approach.



(b) Optimized fixed-size timeslot-based approach.

Figure 4.3: Comparing the original and modified versions of resilient fixed size timeslot-based advance bandwidth reservation.

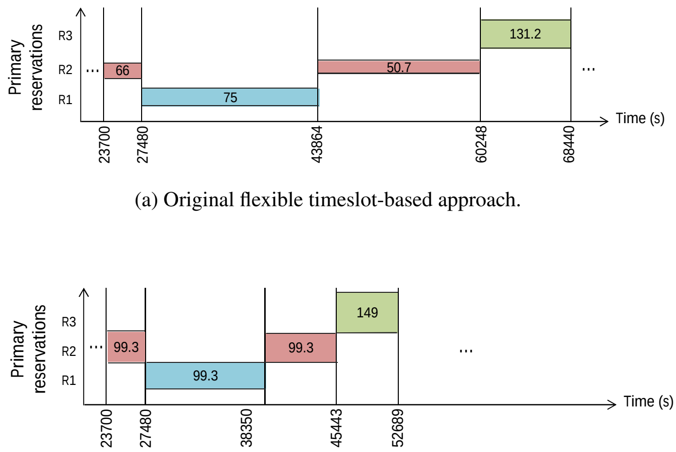
the binary search mechanism [37] is used to find the maximum value within a margin, which we refer to as ϵ. This given margin can be altered to make a tradeoff between achieving a precise value and the complexity of the solution.

Figures 4.3 and 4.4 compare the impact of the optimized resilient algorithm for 3 file-based requests, R1, R2 and R3, for fixed size and flexible approaches respectively. In these figures only primary reservations are shown. These figures reveal how the optimized resilient approach can improve network utilization and increase the probability of admittance for future requests. As can be seen, by allocating a higher volume of a given file, this file can potentially be transferred earlier compared to the original approach. This way, higher capacity is available for requests in the future and the request admittance ratio will potentially be increased.

4.4.3 Combining dynamic, flexible, resilient and RA approaches

A combination of dynamic, flexible, resilient and runtime adaptation approaches results in a highly dynamic environment, including online entrance of reservation requests, flexible timeslot with variable number and duration of timeslots, continually updating the schedule and re-utilizing network resources during runtime, consideration of unforeseen network conditions (failures), distinguishing streaming requests' reservations and usages, reacting to the sudden changes such as failures / repairs and streaming requests' resumes / playbacks.

More specifically, the combination of the RA and flexible approaches is more



(a) Original flexible timeslot-based approach.

(b) Optimized flexible timeslot-based approach.

Figure 4.4: Comparing the original and modified versions of resilient flexible timeslot-based advance bandwidth reservation.

noticeable and challenging, compared to the combination with fixed size timeslots. In fact, the advance reservation schedules based on fixed size and flexible timeslots have been dissimilarly impacted by deploying the RA approach.

4.4.3.1 Impact of RA on advance reservation approaches

Since the schedule is being repeatedly updated by each individual re-scheduling, and due to frequent re-scheduling during the periodic adaptation phase, the use of RA has different impacts on the fixed-size and flexible timeslot-based advance reservation systems. This rescheduling is being done in 2 cases: 1) upon a new reservation submission or 2) in the periodic update phase of the RA approach.

In the fixed size approach, although it's not trivial to find an optimal size, as soon as the size is determined, it won't be changed afterwards. The timeslot sizes are predefined, meaning that the number and duration of timeslots are not affected either during the re-scheduling process invoked upon new scenario arrival or by updating the schedule (i.e. during the periodic update phase of the RA approach). This means that in the fixed size approach, this update only affect the reservations.

By contrast, in the flexible approach, the schedule is entirely impacted during each individual rescheduling process, meaning that in addition to the reservations, the number and duration of timeslots are also affected. Moreover, the frequency of the periodic update and periodic adaptation phases in the RA approach, depends on the number of timeslots. As such, in combination of flexible timeslots, the number of repetition of these phases is also affected by each individual invocation of re-scheduling process, making it much harder to manage compared to the fixed size approach.

4.5 Advance bandwidth reservation architecture

Figure 4.5 shows the various components of our proposed advance bandwidth reservation system. The management layer, provides a reservation interface that allows users to submit their requests, and contains two complementary processes: the dynamic version of resilient advance reservation algorithm (DARA) and the runtime adaptation (RA) approach. The DARA scheduling component is responsible for reserving the required amount of bandwidth including backup capacities for all requests and the RA component dynamically re-optimizes the request transmissions. DARA can be implemented using flexible or fixed size timeslots.

4.5.1 FixedTimeSlot module

The *FixedTimeSlot* module consists of the following components for each time interval.

TimeSlotRequests: determines which unserved requests can be served in the current timeslot. Independent requests are added to the list of current requests if the current interval is greater or equal to the request start time. Requests with start time dependencies can be added provided that all requests on which the request depends are fulfilled.

Limit: This is where the size of the timeslot impacts the bandwidth allocation for file-based requests. The limit component determines the maximum amount of bandwidth reservations for each request in each timeslot. The limit for file-based requests is calculated as follows: the residual volume of this file, which is modified whenever a part of a file is transferred, is divided by the size of timeslot, in order to avoid the extra reservation for the requests. The limit for the streams is their required demand, because their demand is fixed and non-variable.

PrioritySorting: sorts the requests based on their priorities, calculated by the prioritization component.

BWallocation: Bandwidth reservations can be based on single-path or multi-path routing mechanisms. Regardless of the type of routing and protection mechanisms, two different bandwidth allocation algorithms are designed for streams and files because their requirements are dissimilar. Details of these algorithms can be found in [3, 4].

UpdateAndCheckFeasibility: based on the result of the BWallocation component and by calculating the residual demands, the requests requirements are

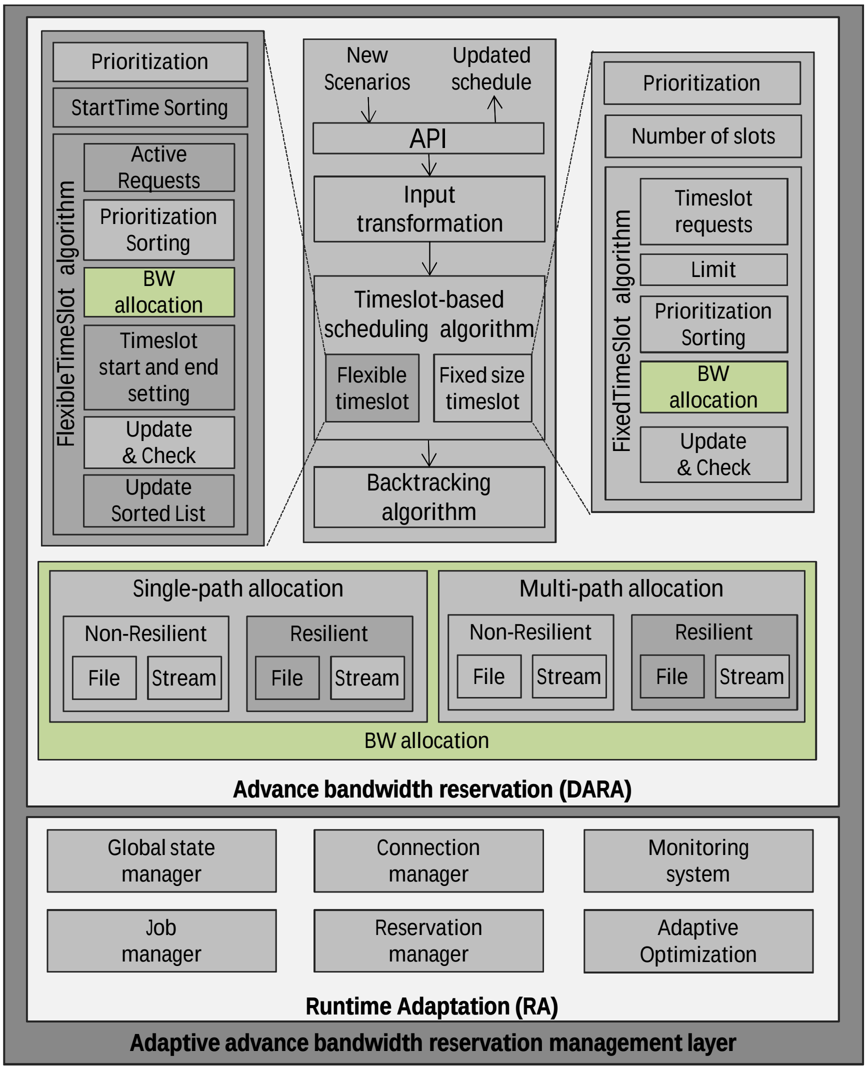


Figure 4.5: A comprehensive overview of the control plane of the adaptive advance bandwidth reservation system.

updated and the feasibility of the results is checked. If the hard deadline of a request is reached, but part of the request has not been transferred yet, rescheduling is infeasible.

4.5.2 FlexibleTimeSlot module

The *FlexibleTimeSlot* module consists of the following components.

StartTimeSorting: All requests submitted to the reservation system are chronologically sorted based on their start time and stored in *sortedSysReqList*. This list does not contain the request with unfulfilled dependencies, because their start time is not specified. Then, the *FlexibleTimeSlot* algorithm jumps to the start time of the earliest request in the *sortedSysReqList*, sets the end of the current timeslot and the start of the next timeslot to the start time of the earliest request.

ActiveRequests: This component sequentially looks for any other requests in the sortedSysReqList which can be started simultaneously in the current timeslot and keeps these requests in the *currentReq* list. For the requests with start time dependencies, the algorithm checks if the requests' dependencies have been eliminated. This implies that all other requests on which this request depends, have already been scheduled.

BWallocation: This component is similar to the component in *FixedTimeSlot* approach but does not take any limitations into account during the bandwidth allocation for file-based requests.

MinDuration: The duration of the current timeslot is calculated by this component. The size of timeslot is determined as the earliest time either an active request is finished or a new request is started. As soon as the timeslot duration is determined, the end of the current timeslot can be set.

UpdateAndCheckFeasibility: As duration of timeslots is not predefined, this component has been modified (compared to the same component in the *FixedTimeSlot* module) to take into account the calculated size of the timeslot when updating the requests' demands and checking the feasibility of the schedule.

Update the sortedSysReqList: All admitted and scheduled requests have to be removed from the sortedSysReqList.

4.5.3 Runtime adaptation module

As illustrated in Figure 4.5, the RA module consists of the following components:

Global state manager: contains all information about scheduling, network and request reservations, connections, demands, deadlines, etc. The time when the current timeslot is started or when it finishes can be retrieved from the global state. The global state manager is currently centralized and in future we intend to deploy this in a decentralized way.

Monitoring system: keeps track of monitored times, residual demand and current allocated bandwidth for all requests. The monitoring system also regularly checks network conditions and raises an event as soon as a failure is detected.

Job manager: contains the list of current advance-scheduled requests and current waiting list requests. Advance-scheduled requests refer to the requests that have already been scheduled by the DARA algorithm to be transferred in the current timeslot. The waiting-list requests are those requests that can potentially be started in this timeslot, but are postponed due to limited network capacity.

Connection manager: decides what to do when a transfer is started or stopped. As long as there are requests with active connections, this component is operational. Whenever a connection for a file transfer is terminated, the links that were in use by this connection become free. In order to improve network utilization, this capacity can be used by other active requests if shared links were in use. To achieve this, after completion of a file transfer, an event will be raised.

Reservation manager: collects all information about the reservations of each request. Primary allocations, backup reservations, extra allocations made during the periodic adaptation phase and allocated network resources can be retrieved from this component.

Adaptive optimization: in charge of optimization to try and push more data than what has been guaranteed through advance reservation. The Adaptive Optimization (AO) algorithm is the main component in this module which is triggered by several events: start of a timeslot, start and end of file based requests, link failures and repairs. Based on this algorithm, the current schedule is analyzed and adapted to use idle bandwidth capacities.

4.6 Advance bandwidth reservation algorithms

We briefly explain the DARA approach and how this approach is extended to support variable timeslot sizes. DARA is invoked several times whenever new scenarios are submitted to the reservation system. The dynamic approach first updates the previously admitted requests' demands based on whether the request is scheduled, is finished or is in progress. Then, new scenarios are sorted. This sorting is based on the earliest average start time of the scenario's requests. If two scenarios have the same value, the one requiring more resources is chosen. Then each scenario in the sorted list is sequentially processed as follows. The prioritization algorithm assigns priorities to the scenario's requests, taking two parameters into account: the estimated hard deadline and the volume. Since the deadline may not be specified for all requests, the hard deadline (i.e., the latest possible deadline) for those with no specific deadline is estimated. This time is calculated by assuming that all requests on which the request depends, use the entire network at once. This gives the latest possible deadline for the requests. In the prioritization algorithm, the sooner *algorithm 10: DARA (Dynamic Advance Reservation Algorithm), updated to support the capability of offering flexible timeslot sizes.*

```
input: scenarios' requests, network infrastructure, approach
Update previously admitted requests;
sortedQueue ← AverageStartSort(new scenarios);
for (scenario ∈ sortedQueue) do
    Set scenario status as Pending;
    currentstate ← Save the current system state;
    Prioritization(scenario's requests);
    sysReqList.Add(scenario's requests);
    if (approach = Fixed) then
        feasible ← FixedTimeSlot(sysReqList, timeslotSize);
    else
        feasible ← FlexibleTimeSlot(sysReqList);
    end
    if (feasible) then
        Update the schedule;
        Set scenario status as Admitted;
    else
        Set current system state to CurrentState;
        Set scenario status as Rejected;
    end
end
```

deadline has the higher priority and volume comes into consideration only when the hard deadlines are equal, where a higher priority is assigned to larger demands. The scenario's requests are added to the list of system requests (*sysReqList*). Then, according to the desired solution, the flexible or fixed timeslot-based approach is chosen and based on the result of either the *FixedTimeSlot* or the *FlexibleTimeSlot* algorithm, DARA decides to admit or reject the scenario. If a feasible schedule has been achieved, the previous schedule is updated, otherwise the algorithm has to backtrack to the previous feasible state. As the *FixedTimeSlot* algorithm has been previously explained in [3], we only discuss the flexible timeslot-based and the optimized version of the resilient algorithms.

4.6.1 FlexibleTimeSlot algorithm

The *FlexibleTimeSlot* algorithm is shown in Algorithm 11. Based on this algorithm, the chronologically sorted requests, stored in the *sortedSysReqList*, are sequentially processed. To do this, the *currentReq* list is determined which contains all requests that could be simultaneously started. Then the *BWallocation* algorithm is invoked to determine the amount of bandwidth reservation for each request. According to the output of the *BWallocation* algorithm, the duration of the current *algorithm 11: The FlexibleTimeSlot algorithm.*

Data: sysReqList
sortedSysReqList ← StartTimeSorting(sysReqList);
New Timeslot.setStart(FirstReqStartTime);
while *(sortedSysReqList* $\neq \varnothing$*)* **do**
    currentReq ← ActiveRequests(sortedSysReqList);
    sortedList ← PrioritySorting(currentReq);
    reservations ← BWallocation(sortedList);
    duration ← MinDuration(reservations);
    timeSlotEnd ← duration + Timeslot.getStart();
    Timeslot.setEnd(timeSlotEnd);
    if *(!UpdateAndCheckFeasibility(reservations, timeSlotEnd))* **then**
        return false;
    else
        New Timeslot.setStart(timeSlotEnd);
        Update the sortedSysReqList;
    end
end
return true

timeslot is determined and the feasibility of the schedule is checked. The *sortedSysReqList* is updated by removing all the admitted and scheduled requests.

The *BWallocation* algorithm is capable of allocating network bandwidth using single-path or multi-path routing mechanisms and resilience can be taken into account by providing redundant reservations during the resource reservation process. To increase network utilization and to offer reliable reservations, we assume that the resilient multi-path approach is followed by using the BWallocationResilient algorithm, shown in Algorithm 12. This algorithm first assigns a cost to each network link using a cost allocation method. We have designed two algorithms for resilient bandwidth allocation depending on the type of request, shown in Algorithms 13 and 14.

algorithm 12: BWallocationResilient

Data: sortedReqList
costAllocation(Links);
for $req \in sortedReqList$ **do**
    if $req\ is\ FB$ **then**
        reservation← BWallocationFBResilient(req);
    else
        reservation← BWallocationVSResilient(req);
    end
end
return reservation;

BWallocationVSResilient is meant to serve the streams and BWallocationFBResilient is defined to serve the files. The main components in these algorithms are BWallocationFB and BWallocationVS, previously proposed in [3]. These algorithms are based on least-cost paths which are calculated based on a modified version of the Dijkstra algorithm [38], in which the calculated costs are used instead of path length. The amount of reservations for each type has been previously set by the Limit function. The common part for both algorithms is repeatedly finding the least-cost paths between source and destination of a given request until the limit of that request is fulfilled. However, provided that the limit of the request is not available, a different trend is followed by each approach. For file-based requests in the BWallocationFB algorithm, the maximum available capacity is reserved as the remainder of the file can be processed during the next timeslots. In the BWallocationVS algorithm, if there is not enough capacity to allocate the request, it can not be served and thus the feasibility is set to false. The next step in the resilient algorithms is to find the backup paths.

algorithm 13: BWallocationVSResilient for streams

Data: a VS request
reservation ← BWallocationVS(req, graph);
maxBW ← max Bandwidth(reservation);
graphReduced ← remove the links in reservation from the network graph;
backupLimit ← Min(maxBW,requestedBackup(req));
backupReservation ← BWallocationVS(req, backupLimit, graphReduced);
if *backupResevation* **then**
| return reservation , backupReservation;
else
| feasibility ← false;
end

Depending on the backup demands and primary allocations, the amount of backup demand is first calculated. Both algorithms check if the backup can be fulfilled. In order to cover single failures, the backups have to be disjointed from the primary paths. As such, the links used in the primary paths are removed from the network and the bandwidth allocation algorithms are reused on the residual network to find the backup paths for that request. If the backups can be found, the primary and backup paths can be successfully allocated for the request. Otherwise, the primary paths have to be removed and sent back to the resource pool. Any QoS violation for the stream is not tolerable, implying that the scheduling is not successful if the backups for the streaming request are not fulfilled.

algorithm 14: BWallocationFBResilient for file transfers

```
Data: an FB request
currentState ← Save the current network state;
upperBound ← Limit(req);
optimalLimit ← Limit(req);
lowerBound ← 0;
while (optimalLimit > 0.1Mbps) do
    reservation ← BWallocationFB(req, currentLimit, graph);
    maxBW ← max Bandwidth(reservation);
    graphReduced ← remove the links in reservation from the network graph;
    backupLimit ← Min(maxBW, requestedBackup(req));
    backupReservation ← BWallocationFB(req, backupLimit, graphReduced);
    if (!backupReservation) then
        set current network state to currentState;
        upperBound ← optimalLimit;
        optimalLimit ← lowerBound+upperBound/2;
    else
        if (upperBound − lowerBound > ϵ) then
            set current network state to currentState;
            lowerBound ← optimalLimit;
            optimalLimit ← lowerBound+upperBound/2;
        else
            return reservation, backupReservation;
        end
    end
end
```

4.6.2 BWallocationFBResilient algorithm

In the original BWallocationFBResilient algorithm [5], if the backup for a given request can not be provided, the limit of primary allocations is repeatedly halved and the possibility of reservation is checked with this lower limit. We argue that this can be improved by deploying a binary search algorithm. That is, given a file-based request, we seek for maximum available bandwidth which satisfies both primary and backup demands. Therefore, if the algorithm finds that value X can satisfy both primary and backup demands, instead of returning this value, which was the case in the original algorithm, a higher value based on the binary search approach is investigated and this is repeated until a near-optimal value (within an ϵ margin) is calculated and returned. This process is shown in Algorithm 14.

4.7 Evaluation setup

For the evaluation of our proposed approach, we have considered a media production network as case study. In such a network, many geographically distributed actors, e.g. production houses, broadcasters, advertisers, collaborate using a shared substrate network to exchange huge amounts of raw video and audio data and streaming transfers. Based on interviews with several Belgian media production actors, including a broadcaster, service provider, and recording facility provider, a set of use case scenarios have been previously defined that serve as a basis for the evaluation. Each scenario contains a collection of interdependent file and streaming transfers with randomized parameters. Each transfer is represented with a source node, a destination node, the start time for streams or the time when the data is ready to be transferred for file-based request, the deadline for file-based request or fixed end-time for streams, the volume for file-based request or the duration and bandwidth requirement for streams. Scenario type 1 represents a soccer after-game discussion program and comprises 5 different file transfer requests. Scenario type 2 is a 30 minute infotainment show and comprises of 18 file transfer requests. Finally, the third type is a news broadcast, consisting 4 file transfer and 4 video streaming requests. The interactions between actors in the three defined scenarios is explained in detail in [3]. In all evaluations, the number of scenarios is 20 (consisting of 209 requests), out of which 7, 7 and 6 scenarios are of type 1, type 2 and type 3 respectively.

In this evaluation we have used 8-node and 25-node network topologies. The 25-node topology is the well-known ATT North America topology [39] consisting of 25 nodes and 56 bidirectional links (112 links in total) which matches to the size of realistic media production networks. A detailed overview of network topologies can be observed from [40].

In order to generate MTBF (mean time between failures), MTTR (mean time to repair) and video stream activation / deactivation events, we used a normal distribution function with equal values for both mean and standard deviation. This value equals 5 minutes for video stream activation/deactivation. It is not trivial to assign a value for MTTR, as it depends on multiple factors, e.g. type of links, type of failures, underlying technology [41]. The main focus of this section is to evaluate the performance of our approach under catastrophic failures in failure-prone networks. As such, 48 minutes is chosen as mean/standard deviation value for the MTTR to experience higher unavailability. To give an insight in number of concurrent failed links for each topology, Table 4.1 shows the maximum and average number of failed links for different MTBF values.

Based on the outcome of the DARA scheduling algorithm, in steady network conditions the requests are either rejected or admitted. However, in presence of failures, not all admitted requests can be completely transferred. Hence, the ad-

Table 4.1: Maximum and average number of concurrent failures for different failure rates.

	8-node topology (32 links)		25-node topology (112 links)	
Failure rate	**AVG**	**MAX**	**AVG**	**MAX**
2h	8.76	17	31.97	46
10h	2.17	6	8.47	20

mitted requests can be categorized as *succeeded*, *degraded* or *failed*. In failure-free networks, succeeded requests equals to the admitted requests. In case of failures, succeeded requests are those that have been fully transmitted. Deciding on the degraded or failed states depends on the users' preference. In this evaluation we assume that the users asked for the same value as percentage of backup demand, i.e. if a request has a demand for 60% backup, this request is considered as degraded if at least 60% (but less than 100%) of its volume has been transferred by its deadline, otherwise the request is failed. It should be noted that for 0% and 100% of backup demand, no degradation has been considered. Those requests are either fully-transferred or failed.

In order to model the dynamic aspect of our model (i.e. the RA approach), we have designed a discrete-event-based simulator using the MASON multi-agent simulation toolkit [42]. For this evaluation, we have compared our proposed solution with a baseline approach. In the baseline approach, whenever a new scenario enters the reservation system, the previous reservations remain untouched and only new requests are being re-scheduled. Flexibility of the timeslots and fault tolerance properties are ignored and the timeslot size of 1 hour is used. The resilient reservations are provided using 2Mbps for the parameter ϵ.

In our evaluations, DARA[XX, YY, ZZ] denotes that timeslot size of *XX*, *YY* percentage of backup demand, and failure rate of *ZZ* hours are used in the dynamic timeslot-based advance reservation algorithm. Each simulation run covers a 24-hour period. All results are averaged over 50 runs with different randomized inputs, error bars denote the standard error.

4.8 Simulation results and discussion

This section evaluates the proposed dynamic flexible timeslot-based scheduling algorithms and compares the quality and execution time of this approach to the fixed-size algorithm. In the fixed size timeslot-based solution, timeslot granularities of 5 minutes to 60 minutes are used. The influence of the available bandwidth, network load, failure rates, backup demands and computational times are assessed.

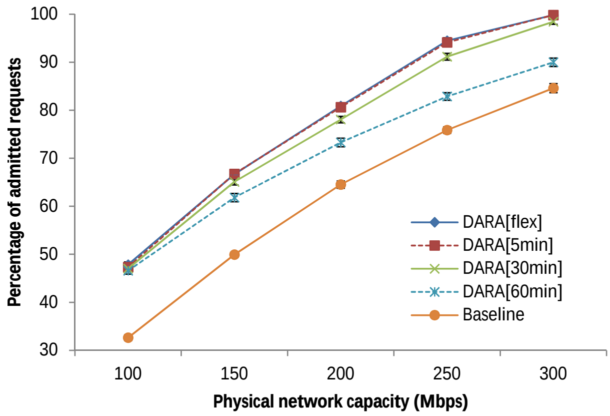


Figure 4.6: Comparing the impact of network capacity in the flexible and fixed size DARA approaches in the 8-node topology.

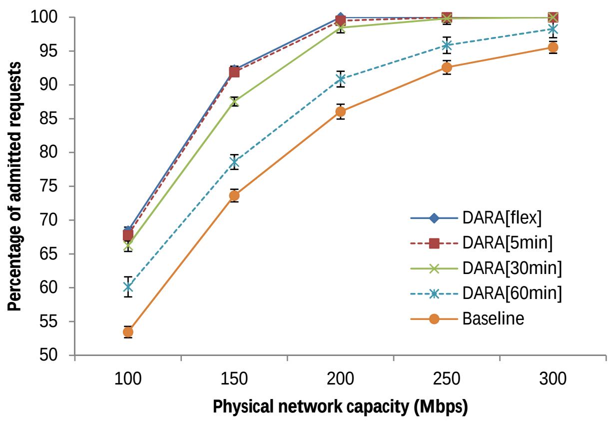


Figure 4.7: Comparing the impact of network capacity in the flexible and fixed size DARA approaches in the 25-node topology.

4.8.1 Comparing DARA fixed and DARA flex

In Figures 4.6 and 4.7, the network infrastructure has been configured for different available bandwidths, to investigate the impact of network capacity on the performance of our algorithms. The number of admitted requests in the DARA approach for flexible timeslots and fixed-size timeslots with different granularities, varying from 5-minute to 1-hour sizes, are evaluated. As can be seen in these figures, the highest percentage of admitted requests is achieved by the flexible advance scheduling algorithm. In the fixed-size approaches the longest timeslot size of 1 hour shows the worst performance in terms of number of admitted requests and this quality is improved when the time interval size is more fine-grained. Compared to the best results obtained by near-optimal fixed size timeslot-based approach (i.e. 5 minutes), the flexible DARA approach shows slightly higher percentage of admitted requests, up to 0.53% and 0.66% respectively and compared to the baseline approach, the flexible timeslots on average improve the admittance ratio 16.43% and 11.87% for the smaller and larger topologies respectively.

Figures 4.8 and 4.9 compare the request admittance ratio for the fixed size and flexible approaches when the network load increases, from 2 to 20 scenarios. In all experiments, flexible timeslots on average show the most desirable performance, resulting in an increase of up to 0.6% and 1.2% percentage of admitted requests, compared to the fixed 5-minute timeslot size and compared to the baseline approach, the flexible timeslots improves the admittance ratio on average 12.56% and 16.93% for 8-node and 25-node topologies respectively.

Figures 4.10 and 4.11 compare the execution times of fixed size, flexible and baseline approaches when the network load increases. These figures show that in the fixed size approach, in addition to the number of scenarios, the size of the timeslot has a large impact on the execution time. However, in the flexible approach an increase in the number of scenarios leads to an increase in the number of timeslots, resulting in a steeper increase when the number of scenarios grows.

4.8.2 Resilient DARA fixed vs. resilient DARA flex

In this evaluation, the impact of different backup demands, including 0%, 50% and 100% are analyzed. Due to similar trends in both topologies, this evaluation is only shown for 25-node topology. Figure 4.12 analyzes the impact of network capacity and percentage of backup demand on the performance of our approaches. As can be seen, regardless of the backup demand, the flexible approach shows almost similar performance as the 5-minute fixed size timeslots.

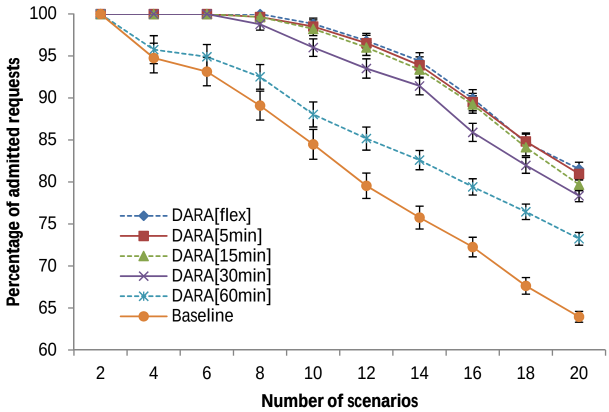


Figure 4.8: Comparing the performance of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 8-node topology. Network capacity is 200Mbps.

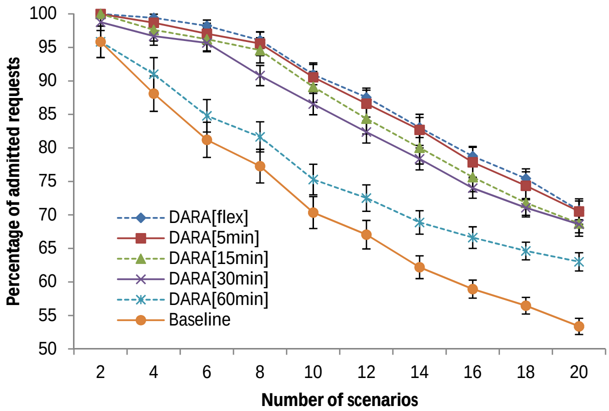


Figure 4.9: Comparing the performance of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 25-node topology. Network capacity is 100Mbps.

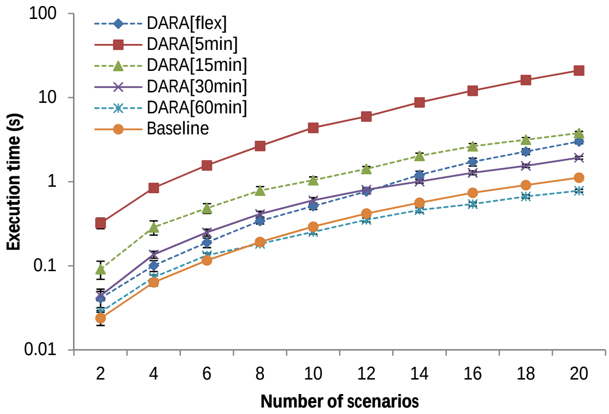


Figure 4.10: Comparing the execution time of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 8-node topology. Network capacity is 200Mbps.

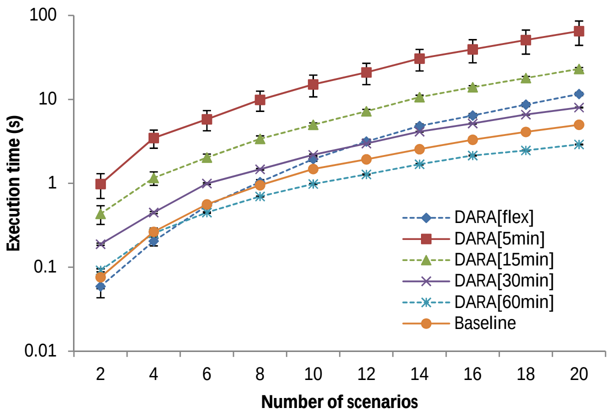


Figure 4.11: Comparing the execution time of flexible and fixed timeslot-based DARA approach with different number of scenarios in the 25-node topology. Network capacity is 100Mbps.

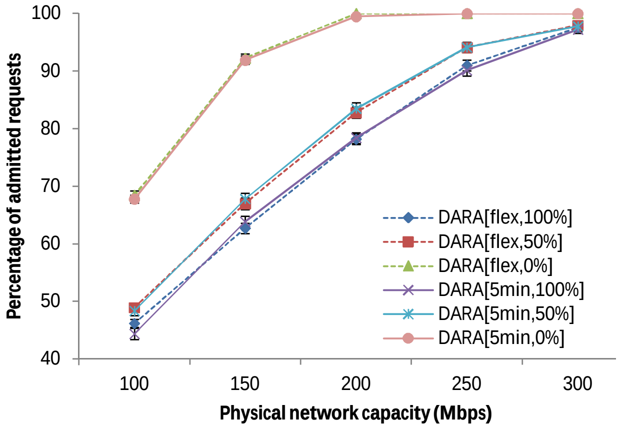


Figure 4.12: Comparing the impact of backup demand in the flexible and fixed size resilient DARA approaches in the 25-node topology.

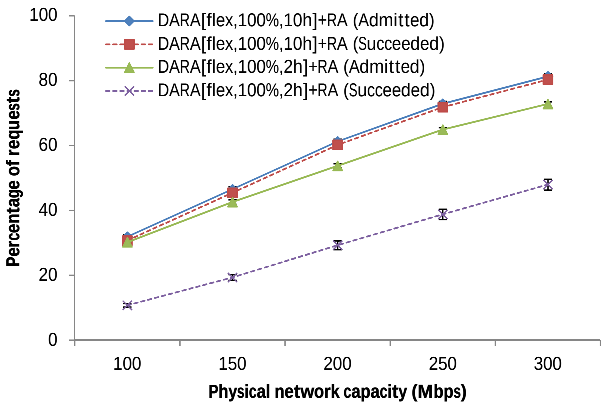


Figure 4.13: Comparing the impact of failure rates in the flexible and fixed size resilient DARA approaches in the 8-node topology.

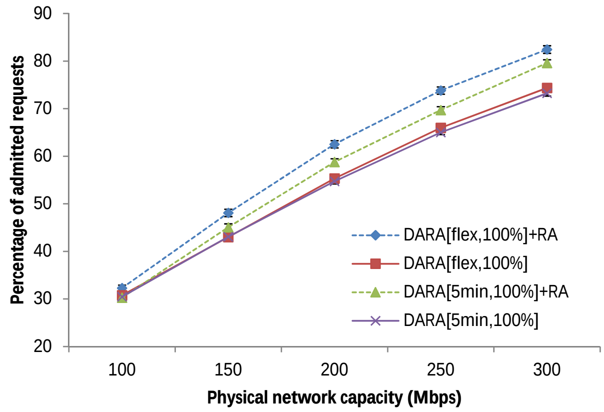


Figure 4.14: Comparing the impact of runtime adaptation (RA) in the flexible and fixed size resilient DARA approaches in the 8-node topology.

4.8.3 Resilient DARA fixed+RA vs. resilient DARA flex+RA

Figure 4.13 compares the impact of different failure rates on the percentage of admitted and succeeded requests in the 8-node topology. This evaluation reveals that on average, the catastrophic failure rate of 2h results in a success rate of only 65.8% of already admitted requests, while this is up to 98.8% for 10h failure rate.

In Figures 4.14 and 4.15, the impact of the runtime adaptation approach on fixed size and flexible approaches are evaluated for 100% backup demand. These figures first reveal that, in case of no failures, the percentage of admitted requests in the flexible approach in combination with the runtime adaptation approach is impacted to a greater extent, compared to the fixed-size approach. While the runtime adaptation is able to achieve up to 8.1% and 8.4% in combination with the flexible approach for 8-node and 25-node topologies, this impact is up to 6.36% and 3.7% with fixed-size timeslots. Second, the combination of RA with the flexible approach shows the highest performance, on average 3.15% and 3.6%, compared to the combination of RA and 5-minute timeslots, in the 8-node and 25-node topologies respectively.

Figures 4.16 and 4.18 show the average percentage of succeeded requests (out of all admitted requests) in 8-node and 25-node networks respectively. Figures 4.17 and 4.19 compare the same experiments for the success rate of admitted requests. In these figures, the failure rate of 10h is considered. As can be seen in these figures, the highest percentage of succeeded requests is almost always achieved by the flexible timeslot size with full protection. More importantly, according to these evaluations, we can conclude that in case of failures, the flexible

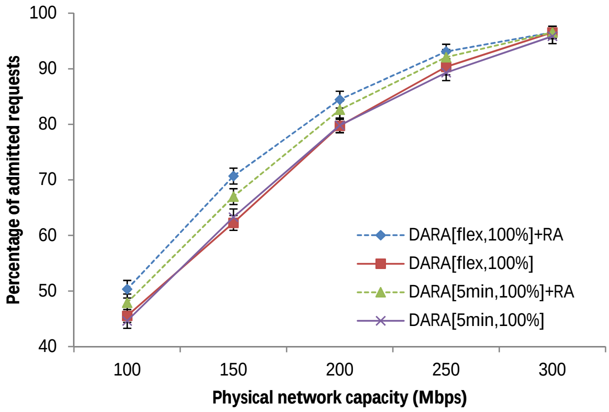


Figure 4.15: Comparing the impact of runtime adaptation (RA) in the flexible and fixed size resilient DARA approaches in the 25-node topology.

approach with 100% backup provisioning always achieves a higher percentage of succeeded requests by up to 3%, compared to 50% backup demand with 5-minute timeslots.

However, when it comes to the success rate of admitted requests, shown in Figure 4.17 and 4.19, 5-minute timeslots shows the best performance. The success rate of 5-minute timeslots is up to 0.7% and 1% higher for 100% and 50% backup provisioning respectively in the smaller topology. In the 25-node topology, as can be seen in Figure 4.19, the 5-minute timeslots can achieve on average 0.88% and 2.5% higher success rate for 100% and 50% backup demand. We argue that in these evaluations the success rate of the flexible algorithm is slightly lower because the flexible solution accepts more requests. When failures occur, the flexible approach has more requests that need to be kept successful. The 5-minute timeslot-based algorithm rejects more requests from the beginning, leaving it with more breathing room in terms of network capacity, making it easier to keep those requests successful.

Figures 4.20 and 4.21 evaluate the impact of 10h failure rate on the final state of reserved requests. As can be seen in Figure 4.20, with 10h failure rate, the highest admittance and success rate is achieved by the flexible timeslots, on average 1.77% and 1.34% respectively compared to the 5-minute timeslot size. In the 25-node topology, while a higher percentage of requests are admitted by flexible approach, the percentage of succeeded requests for flexible and 5-minute timeslots are almost equal. However, with a catastrophic failure rate of 2h in Figure 4.22, while the flexible timeslots results in the highest request admittance ratio, the success rate is slightly lower, on average 0.38%, compared to the 5-minute fixed size approach.

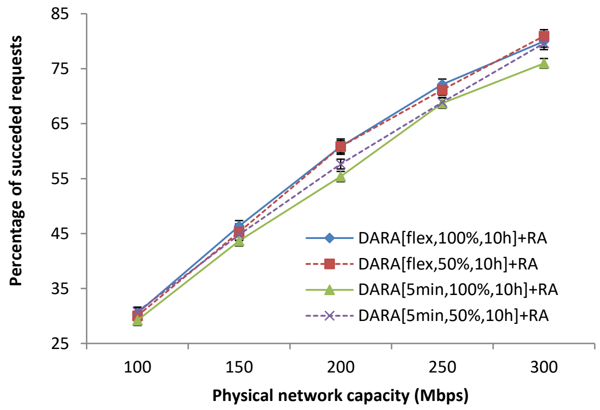


Figure 4.16: Comparing the percentage of succeeded requests in the flexible and fixed size DARA approaches in the 8-node topology.

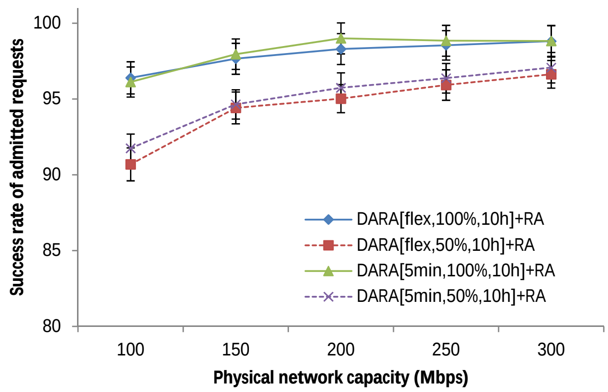


Figure 4.17: Comparing the success rate of admitted requests in the flexible and fixed size DARA approaches in the 8-node topology.

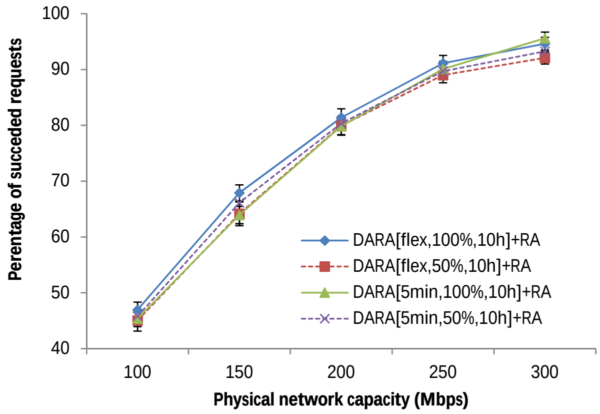


Figure 4.18: Comparing the percentage of succeeded requests in the flexible and fixed size DARA approaches in the 25-node topology.

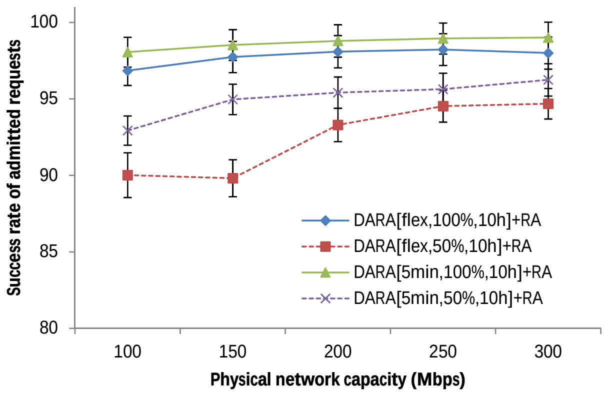


Figure 4.19: Comparing the success rate of admitted requests in the flexible and fixed size DARA approaches in the 25-node topology.

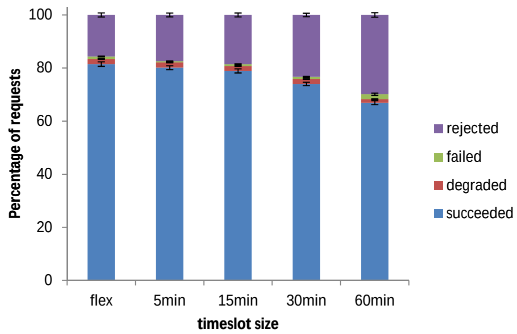


Figure 4.20: Final state of requests with a failure rate of 10h in the 8-node topology. Backup demand is 50% and network capacity is 300Mbps.

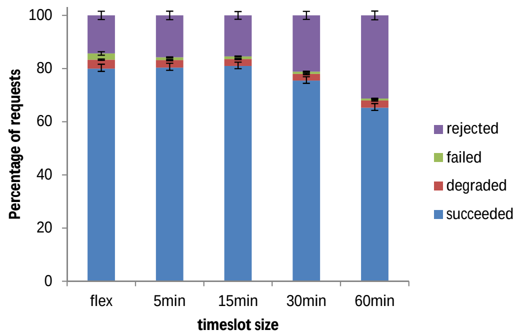


Figure 4.21: Final state of requests with a failure rate of 10h in the 25-node topology. Backup demand is 50% and network capacity is 200Mbps.

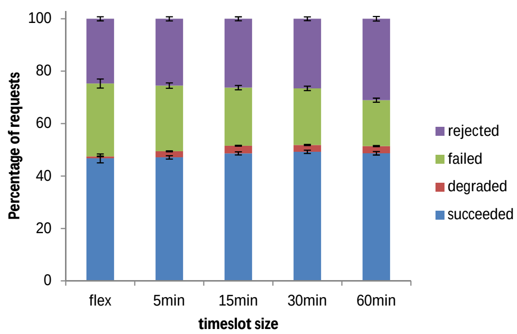


Figure 4.22: Final state of requests with a failure rate of 2h in the 8-node topology. Backup demand is 50% and network capacity is 300Mbps.

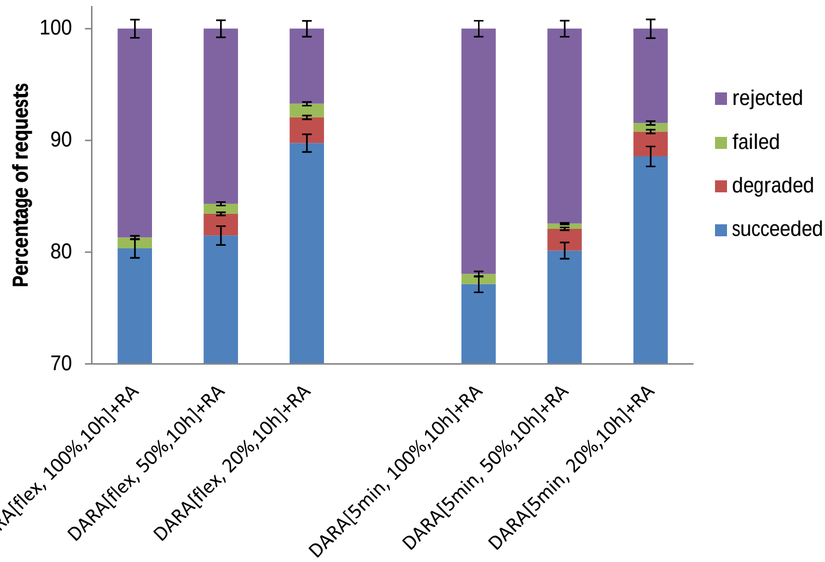


Figure 4.23: Comparing the impact of backup demand on the final state of requests with a failure rate of 10h in the 8-node topology. Network capacity is 300Mbps.

The impact of different backup demands of 20%, 50% and 100% on the final state of admitted requests is evaluated in Figures 4.23 for smaller and larger topologies respectively. A network capacity of 300Mbps and failure rate of 10h is used. Our simulation results for the 8-node topology reveal that the flexible approach is able to fully transfer of admitted requests up to 3.2%, 1.34% and 1.2% with 100%, 50% and 20% backup demands respectively, compared to the equivalent evaluations for 5-minute timeslots.

4.8.4 Discussion

Based on these evaluations, we can conclude that the flexible approach is a better fit for advance bandwidth reservation of media-centric network as in most evaluation cases the flexible approach has on average and quality-wise outperformed the fixed size approach and shows a lower execution time. In individual evaluations, however, even 15-minute timeslot can outperform 5-minute or flexible timeslots. There exist 3 reasons for this. 1) In our evaluations, we admit all or none of scenario requests. 2) We compare number of admitted requests, instead of the number of admitted scenarios. 3) Admitted scenarios are not allowed to be rejected during the next rescheduling. We elaborate on this with an example to point out in which situations the flexible timeslot shows worse performance. Considering 3 priority-based sorted scenarios, S1 (18 requests), S2 (5 requests, high demand) and S3 (8 requests, low demand). While the flexible approach is able to admit S1 and S2, resulting in 23 admitted requests, the fixed size approach may admit S1 and S3, resulting in 26 admitted requests.

4.9 Conclusions

This chapter dealt with the design, development and evaluation of a flexible, reliable and adaptable advance bandwidth reservation system for media-centric networks. This architecture provides a highly dynamic framework capable of improving network utilization and request admittance ratio as well as reacting to sudden network changes. We distinguished the restrictions of fixed size timeslots and showed that while the flexible approach, in most cases, slightly outperforms the fixed size solution, the execution duration is considerably reduced. Adding resilience features increases the reliability of our approach and guarantees single link failure recovery. However, using redundancy imposes significant performance overheads and additional costs. To mitigate the side-effect of redundant allocations and dynamically reconfigure transmissions in response to sudden changes in network conditions, the runtime adaptation approach is introduced which applies a constant monitoring, adaptation and re-optimization is being applied at runtime. This improves the performance of transfers and request admittance ratio and percentage of fully transferred requests in case of failures. Our results showed that the flexible approach with 100% backup demand is able to achieve up to 3% higher percentage of succeeded requests compared to the 5 minute fixed timeslots with 50% backup provisioning.

Acknowledgment

The research leading to these results has been performed within the context of ICON MECaNO, a project co-funded by iMinds, a digital research institute founded by the Flemish Government. Project partners are SDNsquare, Limecraft, Video-House, Alcatel-Lucent, and VRT, with project support from IWT under grant agreement no. 130646.

References

[1] N. Charbonneau and V. M. Vokkarane. *A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks*. IEEE Communications Surveys & Tutorials, 14(4):1037–1064, 2012.

[2] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Algorithms for advance bandwidth reservation in media production networks*. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 183–190. IEEE, 2015.

[3] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Deadline-aware advance reservation scheduling algorithms for media production networks*. Computer Communications, 2015.

[4] M. Barshan, H. Moens, B. Volckaert, and F. De Turck. *Single-path versus multi-path advance reservation in media production networks*. In Network of the Future (NOF), 2015 6th International Conference on the, pages 1–6. IEEE, 2015.

[5] S. Sahhaf, M. Barshan, W. Tavernier, H. Moens, D. Colle, and M. Pickavet. *Resilient algorithms for advance bandwidth reservation in media production networks*. In 2016 12th International Conference on the Design of Reliable Communication Networks (DRCN), pages 130–137. IEEE, 2016.

[6] M. Barshan, H. Moens, B. Volckaert, and F. De Turck. *Design and evaluation of a dual dynamic adaptive reservation approach in media production networks*. Journal of Network and Computer Applications, 80:109–122, 2017.

[7] M. Barshan, H. Moens, B. Volckaert, and F. De Turck. *A Comparative Analysis of Flexible and Fixed Size Timeslots for Advance Bandwidth Reservations in Media Production Networks*. In NOF 2016-7th International Conference on Network of the Future. IEEE, 2016. submitted.

[8] M. Barshan, H. Moens, B. Volckaert, and F. De Turck. *Design and Evaluation of a Flexible Advance Bandwidth Reservation Algorithm for Media Production Networks*. In IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2017. To appear.

[9] K. Rajah, S. Ranka, and Y. Xia. *Advance Reservations and Scheduling for Bulk Transfers in Research Networks*. IEEE Trans. Parallel Distrib. Syst., 20(11):1682–1697, November 2009. Available from: http://dx.doi.org/10.1109/TPDS.2008.250, doi:10.1109/TPDS.2008.250.

[10] C. Xie, H. Alazemi, and N. Ghani. *Rerouting in advance reservation networks*. Computer Communications, 35(12):1411–1421, 2012.

[11] L. Zuo, M. M. Zhu, and C. Q. Wu. *Fast and Efficient Bandwidth Reservation Algorithms for Dynamic Network Provisioning*. Journal of Network and Systems Management, 2013.

[12] H. Alazemi, F. Xu, C. Xie, and N. Ghani. *Advance reservation in distributed multi-domain networks*. IEEE Systems Journal, 2013.

[13] C. Guok, E. N. Engineer, and D. Robertson. *ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)*. Internet2 Joint, 2006.

[14] B. Gibbard, D. Katramatos, and D. Yu. *TeraPaths: end-to-end network path QoS configuration using cross-domain reservation negotiation*. In Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on, pages 1–9. IEEE, 2006.

[15] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. *StorNet: Integrated Dynamic Storage and Network Resource Provisioning and Management for Automated Data Transfers*. In Journal of Physics: Conference Series, volume 331, page 012002. IOP Publishing, 2011.

[16] S. Sharma, D. Katramatos, D. Yu, and L. Shi. *Design and Implementation of an Intelligent End-to-end Network QoS System*. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pages 68:1–68:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press. Available from: http://dl.acm.org/citation.cfm?id=2388996.2389089.

[17] S. Naiksatam and S. Figueira. *Elastic reservations for efficient bandwidth utilization in LambdaGrids*. Future Generation Computer Systems, 23(1):1–22, 2007.

[18] L.-O. Burchard, H.-U. Heiss, and C. A. De Rose. *Performance issues of bandwidth reservations for grid computing*. In Computer Architecture and High Performance Computing, 2003. Proceedings. 15th Symposium on, pages 82–90. IEEE, 2003.

[19] M. Balman, E. Chaniotakisy, A. Shoshani, and A. Sim. *A flexible reservation algorithm for advance network provisioning*. In 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–11. IEEE, 2010.

[20] L. Shi, S. Sharma, D. Katramatos, and D. Yu. *Scheduling end-to-end flexible resource reservation requests for multiple end sites*. In Computing, Networking and Communications (ICNC), 2015 International Conference on, pages 810–816. IEEE, 2015.

[21] N. R. Kaushik, S. M. Figueira, and S. A. Chiappari. *Flexible time-windows for advance reservation scheduling*. In 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pages 218–225. IEEE, 2006.

[22] S. J. Nirmala, N. Tajunnisha, and S. M. S. Bhanu. *Service provisioning of flexible advance reservation leases in IaaS clouds*. International Journal of Big Data Intelligence, 3(3):154–162, 2016.

[23] H. Bai, F. Gu, K. Shaban, J. Crichigno, S. Khan, and N. Ghani. *Flexible advance reservation models for virtual network scheduling*. In Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th, pages 651–656. IEEE, 2015.

[24] M. A. Netto, K. Bubendorfer, and R. Buyya. *SLA-based advance reservations with flexible and adaptive time QoS parameters*. In International Conference on Service-Oriented Computing, pages 119–131. Springer, 2007.

[25] H. Shen, L. Yu, L. Chen, and Z. Li. *Goodbye to Fixed Bandwidth. Reservation: Job Scheduling with Elastic. Bandwidth Reservation in Clouds*. In International Conference on Cloud Computing Technology and Science. IEEE, 2016.

[26] T. Watanabe, T. Omizo, T. Akiyama, and K. Iida. *ResilientFlow: Deployments of distributed control channel maintenance modules to recover SDN from unexpected failures*. In 11th International Conference on the Design of Reliable Communication Networks (DRCN), pages 211–218. IEEE, 2015.

[27] D. S. Yadav, A. Chakraborty, and B. Manoj. *A Multi-Backup Path Protection scheme for survivability in Elastic Optical Networks*. Optical Fiber Technology, 30:167–175, 2016.

[28] S. Tanwir, L. Battestilli, H. Perros, and G. Karmous-Edwards. *Dynamic scheduling of network resources with advance reservations in optical grids*. International Journal of Network Management, 18(2):79–106, 2008.

[29] T. Li, B. Wang, C. Xin, and X. Zhang. *On survivable service provisioning in WDM optical networks under a scheduled traffic model*. In Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, volume 4, pages 5–pp. IEEE, 2005.

[30] T. Li and B. Wang. *On optimal survivability design in WDM optical networks under a scheduled traffic model*. In Design of Reliable Communication Networks, 2005.(DRCN 2005). Proceedings. 5th International Workshop on, pages 8–pp. IEEE, 2005.

[31] C. Cavdar, M. Tornatore, F. Buzluca, and B. Mukherjee. *Dynamic scheduling of survivable connections with delay tolerance in WDM networks*. In INFOCOM Workshops 2009, IEEE, pages 1–6. IEEE, 2009.

[32] C. Cavdar, M. Tornatore, F. Buzluca, and B. Mukherjee. *Shared-path protection with delay tolerance (SDT) in optical WDM mesh networks*. Journal of Lightwave Technology, 28(14):2068–2076, 2010.

[33] *ICON MECaNO project*. http://www.iminds.be/en/projects/mecano/, 2014 - 2016. Accessed: 2017-02-10.

[34] J. Zheng and H. T. Mouftah. *Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks*. In IEEE International Conference on Communications (ICC), volume 5, pages 2722–2726. IEEE, 2002.

[35] E. He, X. Wang, V. Vishwanath, and J. Leigh. *CAM03-6: AR-PIN/PDC: Flexible Advance Reservation of Intradomain and Interdomain Lightpaths*. In IEEE Global Telecommunications Conference (GLOBECOM'06), pages 1–6. IEEE, 2006.

[36] C. Barz, U. Bornhauser, P. Martini, and M. Pilz. *Timeslot-based resource management in grid environments*. In IASTED Conference on Parallel and Distributed Computing and Networks, PDCN, 2008.

[37] D. H. Lehmer. *Teaching combinatorial tricks to a computer*. In Proc. Sympos. Appl. Math. Combinatorial Analysis, volume 10, pages 179–193, 1960.

[38] T. Cormen. *Introduction to Algorithms*. MIT Press, 2009. Available from: http://books.google.be/books?id=Jwr8jwEACAAJ.

[39] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. *The internet topology zoo*. Selected Areas in Communications, IEEE Journal on, 29(9):1765–1775, 2011.

[40] M. Barshan, H. Moens, and B. Volckaert. *Dynamic adaptive advance bandwidth reservation in media production networks*. In 2016 IEEE NetSoft Conference and Workshops (NetSoft), pages 58–62. IEEE, 2016.

[41] M. M. Alam Khan. *Multi-Path Link Embedding for Survivability in Virtual Networks*. Master's thesis, University of Waterloo, 2015.

[42] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. *Mason: A multiagent simulation environment*. Simulation, 81(7):517–527, 2005.

5

Algorithms for Network-Aware Application Component Placement for Cloud Resource Allocation

In previous chapters we proposed several advance bandwidth reservation and bandwidth management approaches for media-centric networks. In such networks, data centers are mainly used to store large amount of data and run the applications. Data centers can offer cloud-based infrastructures as a service. Due to the soaring popularity of cloud-based services over the last years, the size and the complexity of cloud environments has been growing quickly. In the context of cloud systems, mapping a number of application components to a set of physical servers and assigning cloud resources to those components is challenging. In this Chapter, we present optimal and heuristic solutions for network-aware placement of multi-component applications with differing component characteristics. The optimal ILP-based solution minimizes the application rejection rate and the cost of mapping while respecting application component requirements and physical network limitations. As the execution time of the optimal model scales exponentially, we also offer scalable heuristic solutions for centralized and hierarchical application placement, which are thoroughly explained and evaluated and compared to the optimal solution. Our evaluations show that while the proposed centralized heuristic is near-optimal, the hierarchical approach is much faster and offers higher scalability compared to a centralized approach, e.g. mapping 2.7 million application components onto 512k servers. Moreover, the percentage of servers used and fully *placed applications remain close to that of the centralized and optimal solutions.*

⋆ ⋆ ⋆

M. Barshan, H. Moens, Steven Latré, B. Volckaert and F. De Turck

Accepted to be published on the Journal of Communications and Networks.

5.1 Introduction

Cloud computing has emerged as a powerful paradigm which has revolutionized the way in which computing infrastructures are used. Elasticity and on demand services are the main characteristics which make these computing infrastructures appealing. Nowadays many companies make use of cloud technologies to reduce costs, increase flexibility and to respond faster to customer needs. Although the benefits of cloud systems are considerable, numerous challenges remain, among them, effective supervision of resource usage, scalability and in particular resource allocation to the applications. The application placement refers to the act of deciding where on the clusters of servers, applications are placed [1].

The initial placement policy used to map applications onto physical servers has important effects in terms of application performance and resource efficiency, and making a suitable initial decision is essential to reduce the future need for migrations. In literature, most efforts have been directed towards optimizing the usage of CPU, memory and disk resources, and reducing the energy consumption of physical servers. According to [2], however, there has been a drastic increase in the amount of data generated and consumed by each application. Thus, resource allocation methods have to expand and take into account this growing focus on data. Inappropriate placement of application components with heavy communication requirements could lead to the saturation of certain network links, with subsequent negative impact on applications, e.g. slow response or execution times.

Cloud-based applications often consist of multiple interacting components with differing requirements. While some components may consist of high CPU intensive tasks, requiring powerful computational servers, others may deal with large volumes of data, making servers tailored to data-throughput more appropriate for such components. In order to offer an efficient placement service, different requirements of application components should be taken into account in deciding on where to deploy application components.

In order to address this problem of network-aware application placement in cloud environments, in this chapter, we first introduce a centralized ILP-based op-

Xi	Component i of application X
Xi (R)	Redundant version of Xi

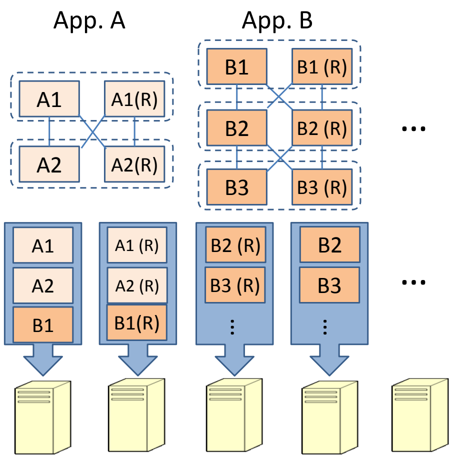


Figure 5.1: The process of application component placement with the anti-collection placement requirement. Redundant components are not allowed to be allocated on the same host as non-redundant components.

timal model. The main objective is maximizing the percentage of mapped applications while taking the cost of application mapping into account. In our proposed approach, we deal with multi-component applications with multiple component types. The interdependence among application components implies that either the entire application or none of the application components are mapped. This is known as the full deployment placement constraint [3]. Therefore, a mapped application is an application for which all components are successfully allocated. Moreover, due to the characteristics of our applications, in order to have deterministic performance and for security reasons we have made a distinction between different types of application components. Each pair of application components may either be allowed to share a hardware resource or not. This isolation of component types can be modeled as an anti-collocation placement constraint [3]. In our approach, the anti-collocation constraint implies that different component types are not allowed to be placed onto the same servers. We achieve this by ensuring that each server is only allowed to place one type of application component. It should be noted that these constraints can be applied to a wider range of generic problems, such as multi-tenant applications with a strong focus on data security (banking, insurance, etc.) or anti-collocation of redundant components for increased reliability and fault tolerance purposes. Figure 5.1 shows an example of a reliable application component placement onto a small cluster of cloud servers. As illustrated in this figure, each application component and its redundant version must be mapped onto two different physical servers.

Due to the NP-hardness of the problem [4–6] and limited scalability of the optimal model, we also propose an approximate centralized heuristic as well as a hierarchical approach for large-scale cloud environments with the following design goals: scalability and performance. While centralized approaches are omniscient in nature and can make better placement decisions, our hierarchical solution has been designed in such a way that component placement optimality is nearing that of the centralized heuristic approach. Also, the proposed hierarchical algorithm executes faster as each management cluster maintains a partial view of the network. In this chapter, we will prove that the resource allocation process is scalable both in number of cloud servers (e.g. 512k servers) and the number of application components (2.7 million application components) needing to be placed onto the cloud servers. Furthermore, as part of our presented approach an application placement policy, prioritizing local deployment is taken into account for each administrative domain. This partial solution also tries to minimize the number of servers used within each administrative domain. This feature, known as server consolidation, is mostly effective in reducing the power consumption of large-scale datacenters [7].

In the context of modern cloud platforms, the application placement process consists of placing the application's components in a set of VMs (Virtual Machines) and deploying these to the physical infrastructure [8–11]. In this chapter, we assume that the components are already encapsulated in VMs or (micro-service) containers, and the application component placement decides on where to place these VMs on the available physical servers, taking network demands between the VMs into account.

The rest of the chapter is organized as follows. In Section 5.2, the related work is discussed. Section 5.3 describes the architecture for our distributed approach to network-aware application placement for large-scale cloud datacenters. In Section 5.4, the formulation of the ILP-based model is presented in detail. In Section 5.5 the proposed algorithms are extensively discussed, followed by an evaluation of the proposed algorithms in Section 5.6. Finally in Section 5.7, we sum up our contribution and conclude the chapter.

5.2 Related Work

Recently many different approaches for application placement and cloud resource allocation have been proposed [12], each focusing on different aspects of the problem. While many approaches such as [10, 13–15] and [16] rely on centralized approaches which suffer in term of scalability, [17] offers a distributed protocol in order to design a resource management middleware. However, their solution is different as interaction between application components is ignored and there is no global overview of system states for the network administrators. Authors in [18] also focus on resource allocation in IaaS clouds. Nevertheless, the main contribution of this paper is maximizing resource utilization and request acceptance rate.

Another work, [19], clarifies the definition of distributed cloud and the challenges of resource allocation on distributed clouds.

The authors in [20] have made a discussion and categorized the VM placement schemes into resource-aware, power-aware, cost-aware and network-aware. Network-aware approaches, such as [21–25], try to reduce traffic related issues or avoid congestion and in general, VMs make use of network either to communicate with each other or to access the required data from storage components. However, none of these approaches take into consideration the anti-collocation requirement of VMs. On the other hand, deployment of VMs under placement constraints have been investigated in [3, 26] and [27]. Shi et. al. [3, 26] focus on different VM placement constraints, e.g. full deployment, anti-collocation, security, etc. and Breitgand et. al. [27] study SLA compliant placement of multi-VM elastic services under the anti-collocation placement constraints. However, these approaches do not take network demands between the VMs into account.

The model defined in this chapter has similarity with [4] that describes a linear application placement model and [28] which offers a cost-aware algorithm. Nevertheless, these approaches work at the application level, contrary to our component-level application modeling policy where we make a distinction between different component types. In addition, based on the definition of "the best-fit placement" in [4], our centralized heuristic solution follows the same rule, which is finding a feasible server whose residual capacity is minimal. Nevertheless, it differs from our approach as it is centralized and not network-aware. Many other application placement approaches [13, 29–31] also focus on placing a set of independent applications, and do not take the underlying network into account. These approaches do not provide a guaranteed quality of communication between individual interacting components due to potential contention of network resources [12]. In order to mitigate the effect of this risk, network-aware solutions have recently been proposed. Authors in [32] pay special attention to time varying nature of traffic demand and dynamic routing capabilities for medium size data centers. Jiang et al. [33] focus on a multi-path routing scheme and live migration. Incorporation of various network functions has been studied in [34]. While these network-aware approaches only focus on network congestion or minimizing network traffic, we explicitly take both network capacity and delay requirements of applications into account in our formulation as QoS constraints. Among the other network-aware approaches which also take both requirements into account, we refer to [10, 35–38] that are centralized and [39] which specifically focus on geo-distributed clouds.

According to [12], while most of the existing application mapping solutions focus on centralized systems, only 11.5% of approaches, including [40–43] and [44], use hierarchical control schemes and one of those approaches [45] takes network constraints into account. However, their hierarchical approach is not related to their management system but instead it is related to their placement process.

This chapter is related to our previous work on hierarchical cloud resource management [46, 47]. In [46] the underlying network was however not taken into account, while this work specifically focuses on network-aware management of multi-tier interactive applications. [47] does take the underlying network into account, but does not make a distinction between different component types. In this chapter we however make a distinction between multiple component types, taking their requirements and characteristics into account during the placement decision. In addition, this chapter presents much larger scale evaluations.

This work is the extension of our previous work [48] and [49]. In [48], we presented an optimal ILP-based solution which offers limited scalability, making it only suitable for small datacenters. This optimal solution is also useful for benchmarking real-time heuristic algorithms. In [49], a decentralized algorithm was designed and evaluated which can be applied to large-scale cloud environments. In this chapter, we have updated the ILP-based model in 3 ways. First, the model has been generalized to support arbitrary numbers of component types as it was previously limited to two different component types. Secondly, in the previous version, a feasible solution was reached only when all the applications have been mapped. The model has now been extended to incorporate an application mapping failure rate, which penalizes failures of mapping applications, but allows finding solutions when otherwise no feasible solution would be possible. Finally, the main objective has been altered to minimize the application mapping failure rate in addition to the cost of application mapping. Our evaluations have also been extended to be more comprehensive.

Generally, what distinguishes our method from other approaches is the combination of the following: 1) our solution is network-aware, decentralized and scalable; 2) application modeling is component-based with different types, and interaction between those components affects the placement process; 3) SLA agreements and the properties of underlying network, bandwidth and delay, are respected; 4) anti-collocation placement constraints are defined, based on which multiple but same-type components can be placed onto a single physical node; and 5) our method minimizes the number of servers used while respecting application requirements.

5.3 Modeling of a large-scale cloud environment

The model for the proposed large-scale algorithm can be divided into three parts: the physical cloud system, the management plane model and the application model.

Physical cloud system model: In literature, several new topologies have been proposed for future cloud-based environments, such as Jellyfish [50], Dcell [51], etc. Nevertheless, tree-based topologies are still dominant in the existing operational cloud datacenters [52, 53]. Although our proposed approach is not limited to the type of network topologies, in this work the physical cloud system is considered a hierarchical tree topology, which is common in modern data centers. As application components of different types can not be mapped onto the same server in our proposed approach, each server has different responsibility and provides specific functionality.

Management plane model: The management plane relies on multi-layered hierarchical architecture in which three types of managers are defined: LLM (Low Level Manager), MLM (Mid Level Manager) and RLM (Root Level Manager). The LLMs are located in the lowest level of the management hierarchy, the RLM in the top level and MLMs in middle ones. LLMs directly deal with physical servers and are responsible for mapping application components onto physical servers. MLMs manage several LLMs and have the authority to choose the current active LLM, which has to take the responsibility of mapping new application components. RLMs have the general overview of the cloud management systems. In multi-domain cloud environments, RLMs can communicate with other domains if the need arises. The management plane specifications are shown in Table 5.1. The number of management levels (|ML|) and the number of supported servers (|SS|) for each LLM are taken and the branch factor of each tier (μ) is calculated for each management domain. In addition, the number of supported servers and the number of levels determine the number of LLMs. By calculating the level branch factor the number of MLMs can be achieved as follows.

$$\mu = \sqrt[|ML|-1]{|LLM|} \tag{5.1}$$

$$|LLM| = \lceil |S|/|SS| \rceil \tag{5.2}$$

$$|MLM| = \sum_{level=1}^{|ML|-2} \mu^{((|ML|-1)-level)} \tag{5.3}$$

An example of a physical cloud system and how this maps to its management plane is illustrated in Figure 5.2. In each administrative domain different servers are chosen as default servers for different component types.

Application model: In this chapter, the architecture of the applications is service oriented, meaning applications can be represented as a service graph and the application topology is a graph. Although an arbitrary number of application component types can be supported by our approach, we focus on two component types in our evaluations: database (e.g. data sources, data stores) and computational (e.g. application business logic or user interfaces) components. Database components store and manage data and are more storage intensive, whereas computational components are more CPU intensive. The application database and computational components are the nodes and connections between these components form the directed links of the application graph. Each application component requires a

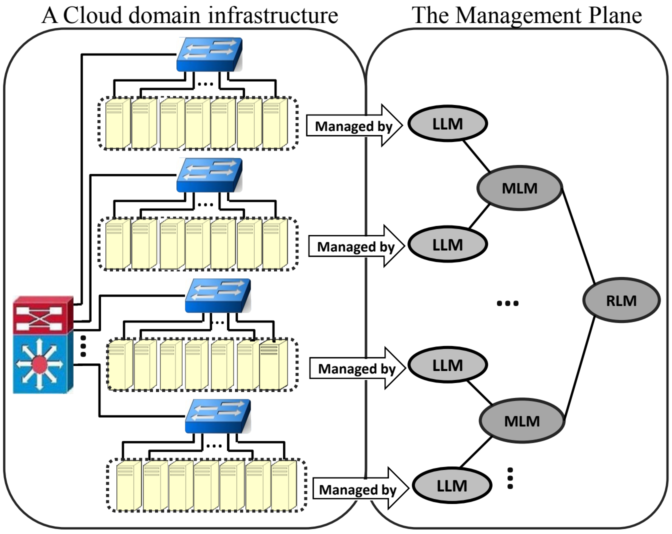


Figure 5.2: The architecture of physical infrastructure and the management plane (LLM: Low Level Manager, MLM: Mid Level Manager, RLM: Root Level Manager).

Table 5.1: The Management plane parameters.

Parameter	**Description**
$\|LLM\| \in \mathbb{N}^+$	Number of LLMs.
$\|MLM\| \in \mathbb{N}^0$	Number of MLMs.
$\|RLM\| \in \mathbb{N}^0$	Number of RLMs.
$\|SS\| \in \mathbb{N}^+$	Number of supported servers for each LLM.
$\|ML\| \in \mathbb{N}^+$	Number of management levels.
$\mu \in \mathbb{N}^+$	Management level branch factor.

specific number of data sources, CPU power, memory and storage, etc. and the storage demand of database components is much higher than their CPU/memory demand, whereas for logic components, power of CPU is of the highest demand. In our model,maximum allowed delay and bandwidth requirements are defined for application links, which need to be satisfied as well.

5.4 Formal ILP-based problem formulation

5.4.1 Introduction to the model

We first present a formal model for application component placement for cloud resource allocation. In this model the substrate network is considered as an undirected graph and the application network as a directed graph due to interdependencies between different components. The parameters of the physical network graph and application network and their descriptions are listed in Table 5.2. Both infrastructures consist of nodes (N) and links (L). In this context, application links refer to the connections between application components with certain demands that need to be met. Nodes in substrate graph ($u \in N_{ph}$) have specific properties such as data storage capacity (S_u), CPU power capacity (C_u) and memory capacity (M_u). Physical links ($e_{uv} \in L_{ph}$) can be either LAN or WAN link and this is determined by a binary variable, $type_{e_{uv}}$ in which 0 refers to the LAN and 1 refers to the WAN links. We made this distinction because WAN links cost more than LAN links. Each link has delay ($D_{e_{uv}}$) and bandwidth capacity ($BW_{e_{uv}}$) properties. It has to be noted that the physical network resource capacities are residual capacities, considering the previous mappings. As we aim to minimize the cost of mapping applications onto cloud resources, the general cost of physical nodes and links as well as cost of using each unit of CPU, memory, storage and link capacity has been taken into account.

Similarly, each application has been considered as a workflow, consisting of multiple components and links between those components form a directed weighted graph. In the application network, *ai* refers to the component *i* of application a with specific computational (c_{ai}), storage (s_{ai}) and memory (m_{ai}) requirements and e_{aij} refers to the link between component *i* and *j* of application a with specified bandwidth ($bw_{e_{aij}}$) and maximum allowed delay ($d_{e_{aij}}$) demands. Different component types are collected in S_{type} and γ_{ai}^{t} is a binary input variable which indicates whether or not ai is of type t.

5.4.2 Decision variables

Seven decision variables have been defined in this ILP model and all variables are binary. First, x_u^{ai} shows the accomplished mapping between component *i* of application *a* and physical node *u*, regardless of the type of component. It has to

Table 5.2: Symbols and notations used in the formal model.

Physical cloud-based infrastructure parameters	
Variable	**Description**
G_{ph}	Physical Graph, $G_{ph} = (N_{ph}, L_{ph})$
N_{ph}	Physical nodes set in G_{ph}
L_{ph}	Physical links set in G_{ph}
$S_u \in N^+$	Available storage capacity of physical node *u*.
$M_u \in N^+$	Available memory capacity of physical node *u*.
$C_u \in N^+$	Available CPU capacity of physical node *u*
$D_{e_{uv}} \in N^+$	Delay of physical link *euv*.
$BW_{e_{uv}} \in N^+$	Bandwidth capacity of physical link *euv*.
$type_{e_{uv}} \in \{0, 1\}$	whether Phy. link *euv* is a LAN or WAN link
$Ccost_u \in N^+$	Cost of each CPU unit of physical node *u*.
$Mcost_u \in N^+$	Cost of each memory unit of physical node *u*.
$Scost_u \in N^+$	Cost of each storage unit of physical node *u*.
$BWcost_{e_{uv}} \in N^+$	Cost of each BW unit of physical link *euv*.
$fcost_u \in N^+$	The fixed cost of using physical node *u*.
$fcost_{e_{uv}} \in N^+$	The fixed cost of using physical link *euv*.
Component-based application parameters	
Variable	**Description**
G_{app}	Application graph, $G_{app} = \{a \| a = (N_a, L_a)\}$
$AppNo$	Number of applications.
$CompNo_a$	Number of components of application *a*.
S_{type}	Set of types of application components.
$ai \in N_a$	Component *i* of application *a*.
$\gamma^t_{ai} \in [0, 1]$	has value 1 if *ai* is of type *t*.
$c_{ai} \in N^+$	Computation demand of application *a*, comp. *i*.
$s_{ai} \in N^+$	Storage demand of application *a*, comp. *i*.
$m_{ai} \in N^+$	Memory demand of application *a*, comp. *i*.
$e_{aij} \in L_a$	Link between comp. *i* and *j* of application *a*.
$bw_{e_{aij}} \in N^+$	Bandwidth demand of link e_{aij}.
$d_{e_{aij}} \in N^+$	Max. allowed delay of link e_{aij}.

be noted that this variable is equal to 0 in two states, either when due to limitations there is no possibility to have a mapping between nodes or when physical node *x* is not chosen for the mapping although it was possible. Next, $f_{e_{uv}}^{e_{aij}}$ indicates success of mapping between physical link e_{uv} and the link between components *i* and *j* of application *a* (e_{aij}).

As we assume that each physical node is exclusively used for components of the same type, variable T_u^t is defined for the purpose of determining whether node *u* is used to host components of type *t*. Multiple components of the same type can be mapped onto the same physical server.

Furthermore, two other variables are defined: B_u is a binary variable to show whether physical node u is used, either as a routing node or a used server in the entire mapping. $B_{e_{uv}}$ is another binary variable to indicate whether physical link e_{uv} is used in the mapping scheme or not. Finally, M_a has been defined to indicate whether the application *a* is fully mapped or not.

$$
\begin{array}{ll}
x_u^{a,i} \in [0,1] & \forall u \in N_{ph}, \forall a \in G_{app}, \forall i \in N_a \\
f_{e_{uv}}^{e_{aij}} \in [0,1] & \forall e_{uv} \in L_{ph}, \forall a \in G_{app}, \forall e_{aij} \in L_a \\
T_u^t \in [0,1] & \forall u \in N_{ph}, \forall t \in S_{type} \\
B_u \in [0,1] & \forall u \in N_{ph} \\
B_{e_{uv}} \in [0,1] & \forall e_{uv} \in L_{ph} \\
M_a \in [0,1] & \forall a \in G_{app}
\end{array}
$$

5.4.3 Objective function

Guaranteeing the quality of service and taking physical constraints into account, application placement services have to be performed with minimum rejection of application placement requests. To achieve this, the sum of M_a variables should be maximized. Minimizing cost of mapping should always be considered the second optimization objective. The cost of physical servers can be determined by combining the individual costs of using each unit of CPU, memory and storage and the fixed cost of using each server. Moreover, since in multi-domain cloud networks the cost of LAN links are almost zero, for estimating the link cost, only the WAN links are taken into account.

The optimization objective function minimizes both the application rejection rate and the cost of mapping with lower and higher priorities respectively. Given α and β as higher (e.g. 10^6) and lower (e.g. 1) priority parameters respectively, *FailureRate* as application mapping failure rate and *NodeMapCost* as cost of physical node usage and *LinkMapCost* as cost of physical links usage, the objective function is defined as follows:

Minimize:

$$\alpha \times FailureRate + \beta \times (NodeMapCost + LinkMapCost)$$

where:

$$FailureRate = \left(AppNo - \sum_{\forall a \in G_{app}} M_a \right) / AppNo$$

$$NodeMapCost = \sum_{\forall u \in N_{ph}} \Bigg(fcost_u \times B_u + \sum_{\forall a \in G_{app}} \sum_{\forall i \in N_a}$$
$$(c_{ai} \times Ccost_u + m_{ai} \times Mcost_u + s_{ai} \times Scost_u) \times x_u^{ai} \Bigg)$$

$$LinkMapCost = \sum_{\forall e_{uv} \in L_{ph}} \Bigg(fcost_{e_{uv}} \times B_{e_{uv}} \times type_{e_{uv}} +$$
$$\sum_{\forall a \in G_{app}} \sum_{\forall e_{aij} \in L_a} \left(bw_{e_{aij}} \times BWcost_{e_{uv}} \times type_{e_{uv}} \right) \times f_{e_{uv}}^{e_{aij}} \Bigg)$$

As a result of minimizing the cost of mapping, the objective of the presented model also minimizes the number of nodes on which the applications can be hosted while satisfying the following constraints and requirements.

5.4.4 Constraints

The defined constraints for application component mapping in cloud system have been organized into 7 sub-Sections as follows:

5.4.4.1 Physical node limitations

Constraints (1), (2) and (3) are considered for physical network nodes and are related to computational, memory and storage limitations respectively. For all physical nodes, the common idea is that sum of all mapped requests' demand must not exceed their maximum available capacities.

$$\sum_{\forall a \in G_{app}} \sum_{\forall i \in N_a} c_{ai} \times x_u^{ai} \leq C_u \qquad \forall u \in N_{ph} \tag{5.1}$$

$$\sum_{\forall a \in G_{app}} \sum_{\forall i \in N_a} m_{ai} \times x_u^{ai} \leq M_u \qquad \forall u \in N_{ph} \tag{5.2}$$

$$\sum_{\forall a \in G_{app}} \sum_{\forall i \in N_a} s_{ai} \times x_u^{ai} \leq S_u \qquad \forall u \in N_{ph} \tag{5.3}$$

5.4.4.2 Physical link limitations

A bandwidth constraint has to be considered for each physical link, regardless of being either WAN or LAN link. Constraint (4) represents that for each physical link, the sum of bandwidth demands of all applications must not exceed maximum available bandwidth.

$$\sum_{\forall a \in G_{app}} \sum_{\forall e_{aij} \in L_a} bw_{e_{aij}} \times f_{e_{uv}}^{e_{aij}} \leq BW_{e_{uv}} \quad \forall e_{uv} \in L_{ph} \tag{5.4}$$

5.4.4.3 Quality of service requirements

For delay and bandwidth, Constraints (5) and (6) are defined for each application link e_{aij}. It has to be noted that the bandwidth constraint can be ignored as it will be satisfied with the physical link Constraint (4).

$$\sum_{\forall e_{uv} \in L_{ph}} D_{e_{uv}} \times f_{e_{uv}}^{e_{aij}} \leq d_{e_{aij}} \quad \forall a \in G_{app}, \forall e_{aij} \in L_a \tag{5.5}$$

$$BW_{e_{uv}} \times f_{e_{uv}}^{e_{aij}} \geq bw_{e_{aij}} \quad \forall e_{uv} \in L_{ph}, \forall a \in G_{app}, \forall e_{aij} \in L_a \tag{5.6}$$

5.4.4.4 Well-connected mapping Constraints

Constraint (7) makes sure that when 2 adjacent application components cannot be physically mapped next to each other, a chain of continuous physical links is used to map each application link. This assures that a closed path is considered to map an application link. As can be observed from this equation, for each physical node *u*, the subtraction of the sum of all incoming and outgoing *f* values should be equal to the subtracts of *X* values between target and source of each application link e_{aij}.

$$\sum_{\forall u \in N_{ph}} f_{e_{uv}}^{e_{aij}} - \sum_{\forall u \in N_{ph}} f_{e_{uv}}^{e_{aij}} = x_u^{aj} - x_u^{ai} \tag{5.7}$$
$$\forall a \in G_{app}, \forall e_{aij} \in L_a, \forall e_{uv} \in L_{ph}, \forall u \in N_{ph}$$

5.4.4.5 Full deployment constraints

The statements below, Constraints (8) and (9), ensure that if an application is mapped each individual component of application *a* has to reside in exactly one server in order to have a successful mapping. Constraint (10) indicates that either all or none of application components have to be mapped.

$$\sum_{\forall u \in N_{ph}} x_u^{ai} = M_a \qquad \forall a \in G_{app}, \forall i \in N_a \tag{5.8}$$

$$\sum_{\forall u \in N_{ph}} x_u^{ai} \leq 1 \qquad \forall a \in G_{app}, \forall i \in N_a \tag{5.9}$$

$$\sum_{\forall i \in N_a} \sum_{\forall u \in N_{ph}} x_u^{ai} = CompNo_a \times M_a \qquad \forall a \in G_{app} \tag{5.10}$$

5.4.4.6 Anti-collocation constraints

We also need other constraints between *X* and *T* values to ensure that each physical node is only used for components of the same type.Constraint (11) and (12) are defined to ensure that the type of the application component and the physical node on which this component is mapped are identical. Since mapping of components of different types is not feasible in the proposed approach, Constraint (13) ensures that for each physical node sum of all T_u^t values for all component types should be less than or equal to 1.

$$\gamma_{ai}^t \times x_u^{ai} \leq T_u^t \qquad \forall u \in N_{ph}, \forall a \in G_{app}, \forall i \in N_a, \forall t \in S_{type} \tag{5.11}$$

$$x_u^{ai} \leq \sum_{\forall t \in S_{type}} T_u^t \qquad \forall u \in N_{ph}, \forall a \in G_{app}, \forall i \in N_a \tag{5.12}$$

$$\sum_{\forall t \in S_{type}} T_u^t \leq 1 \qquad \forall u \in N_{ph} \tag{5.13}$$

5.4.4.7 Additional constraints

Constraints (14) and (15) are needed to make logical correlations between physical resources and their usage. *K* in both constraints is a large number. In Constraint (14) its value has to be larger than the sum of all possible *X* values and in a same way larger than all possible *f* values in Constraint (15). In Constraint (16), B_u for each physical node shows that whether node *u* is used to host any type of component or not.

$$\sum_{\forall a \in G_{app}} \sum_{\forall i \in N_a} x_u^{ai} \leq K \times B_u \qquad \forall u \in N_{ph} \tag{5.14}$$

$$\sum_{\forall a \in G_{app}} \sum_{\forall e_{aij} \in l_a} f_{e_{uv}}^{e_{aij}} \leq K \times B_{e_{uv}} \qquad \forall e_{uv} \in L_{ph} \tag{5.15}$$

$$B_u = \sum_{\forall t \in S_{type}} T_u^t \qquad \forall u \in N_{ph} \tag{5.16}$$

5.5 Algorithm descriptions

5.5.1 ILP-based algorithm

This algorithm implements the optimal ILP-based model which was extensively explained in Section III. This ILP-based algorithm is solved using IBM ILOG CPLEX Optimization Studio [54] which is a tool to build efficient optimization models. The objective function minimizes both the application mapping failure rate and the cost of mapping, taking the constraints into account.

5.5.2 Heuristic algorithm

As has been shown in [4, 5] and [6], the problem of application placement onto a network with bandwidth constrained links is NP-hard. Based on computational complexity theory, in large scale environments, an optimal solution for an NP-hard problem is too expensive to be used in practice; instead a near-optimal solution is desired.

In this section a centralized and a hierarchical approach is discussed which we refer to as the Centralized Cloud Mapping Algorithm (CCMA) and the Hierarchical Cloud Mapping Algorithm (HCMA) respectively. These algorithms are executed within the management plane.

The centralized CCMA algorithm is proposed as a near-optimal alternative to the centralized ILP based approach. This centralized approach can be deployed independently and efficiently up to the scale of medium-size networks. We will show that the centralized approach always outperforms the hierarchical solution in terms of number of fully mapped applications. Comparing the quality and complexity, the use of the hierarchical approach is only recommended for large scale environments, where the CCMA can not be practically used due to high complexity. The HCMA has made use of the CCMA algorithm, in combination to the GCMA (Global Cloud Mapping Algorithm). The GCMA is introduced to have interactions between different managers within the hierarchical management plane.

5.5.2.1 Centralized Cloud Mapping Algorithm (CCMA)

This algorithm first arbitrarily chooses different nodes as the default servers for different component types. For each application the algorithm, shown in Algorithm 15, goes through all the components and tries to allocate resources to each component. An illustrative example of this placement for two types of components is shown in Figure 5.3. In order to have minimal bandwidth overhead, the algorithm uses the Dijkstra shortest path algorithm [55] for mapping the application links. However, there are two situations in which the application component cannot be placed onto the default server, either because of physical node limitations or due to physical link limitations. Node limitation occurs when there is not enough residual CPU, memory or storage capacity in one of default servers. In the latter case, again, there are two situations in which the link limitation leads to unsuccessful placement. First, the application components cannot be connected because there are no physical links to connect application components located on different servers. Second, placement can be unsuccessful if bandwidth or delay requirements cannot be resolved.

No matter what is causing unsuccessful application component placements and what the type of component is, the Next Server Selection (NSS) process should be followed to choose another server as a default server. In the NSS process, a Breadth First Search (BFS) algorithm [55] is run with the current default server as the start vertex to initialize the next server selection. We use the BFS because this algorithm finds the nearest server with minimal path length which ensures there is a minimal communication overhead between the new and the previous servers. However, when link limitation occurs, first another server is chosen temporarily by the NSS process and then the algorithm checks the path availability and SLA fulfillment, and sets it as a default server provided that choosing this server satisfies both conditions. Otherwise, the placement is not successful. In this case the algorithm must remove all placed components of the application and backtrack to the state before the placement. This state is saved before placement of each application in order to backtrack when the need arises. After backtracking, the placement starts again with new default servers. This process will be continued until the NSS process is unable to find a new server. If this occurs, placement of the application is not possible.

5.5.2.2 Hierarchical Cloud Mapping Algorithm (HCMA)

The HCMA algorithm is shown in Algorithm 16. Based on this algorithm, all placement requests are sent to the current *active LLM*, using the GCMA algorithm. The GCMA is designed in order to have interactions between different managers within the management plane. The GCMA is run on every manager. In the management hierarchy, each LLM is in charge of its own administrative

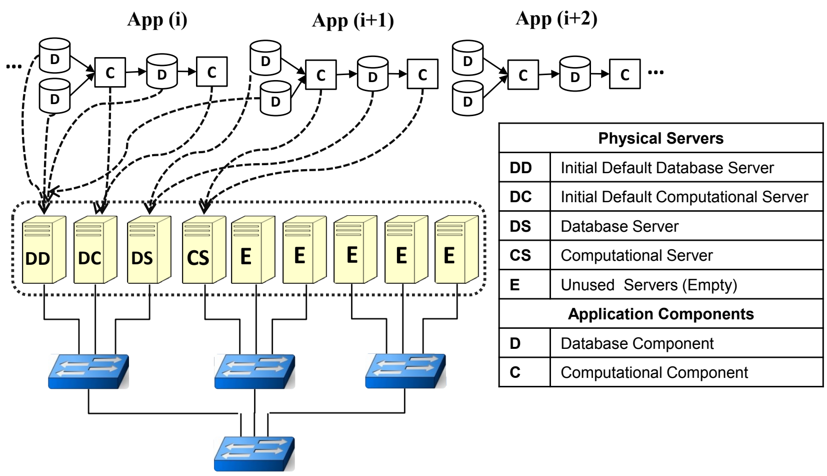


Figure 5.3: The process of application component placement onto a cluster of cloud servers for two types of components (database server and computational server).

domain and the current *active LLM* is the one which is active in mapping the application components. The current active LLM is determined arbitrarily when the algorithm starts. Each manager has two states: “full” and “not full”. A manager is “full” when all its managed servers get fully occupied. The active LLM will be replaced when its state changes to “full”. The next active LLM is chosen by the parent of the current active LLM, i.e. an MLM or the RLM. For each newly arriving application, the HCMA invokes the GCMA with the current active LLM and a “new request” message. In the GCMA three types of messages are defined: “new request”, “from the parent node” and “full”. This is illustrated in Figure 5.4. Next, the GCMA sends the application request to the current active LLM by calling the CCMA, which was presented earlier. In hierarchical approach the CCMA is run on every LLM. If this default administrative domain is not able to place the entire application components, the status of the current active LLM changes to “full”. Afterward, this LLM calls the GCMA with a “full” message to its parent, indicating that the application cannot be placed onto this cluster. In this step, interaction between different management entities starts.

Global Cloud Mapping Algorithm (GCMA): The GCMA is a hierarchical algorithm, listed in Algorithm 17. Based on this algorithm when a request is received, three cases can be distinguished:

1. A request is received by the highest level manager (RLM): The request will be forwarded to the next unvisited domain with a “from the parent node” message. If all domains are full and the request is rejected the cloud system is not able to place this application.

algorithm 15: The Centralized Cloud Mapping Algorithm (CCMA), run on the management plane in the centralized approach and on each LLM in the hierarchical approach.

```
input: applications
for (c ∈ application components) do
    while (Map (c, defaultServer) = false) do
        if (Due to node limitations) then
            New defaultServer ← NSS(default Server);
            if (New defaultServer = Null) then
                Mapped_app ← false;
                return false;
            end
        else if (Due to link limitations) then
            Temp Server ← NSS (defaultServer);
            if (CheckLinks (Temp Server)) then
                New defaultServer ←Temp Server;
            else
                Mapped_app ← false;
            end
        end
        if (one of default servers = null) then
            Mapped_app ← false;
            return false;
        end
    end
end
```

2. A request is received by the mid-level manager (MLM): The request will be forwarded by applying the same policy to one dedicated lower-level manager with a "from the parent node" message until the target LLM located at the lowest level is reached. Provided that all domains are full, the status of this manager turns to "full" and the request with a "full" message will be forwarded to the parent which can be either another MLM or the RLM.

3. A request is received by the lowest level manager (LLM): At this level all request messages will be either "new request" or "from the parent node". No matter who is the request sender, the manager executes the CCMA algorithm. In a saturation case when placement of new applications is not possible, the LLM has to send the newly arriving requests to its parent and introduce itself as a "full" manager.

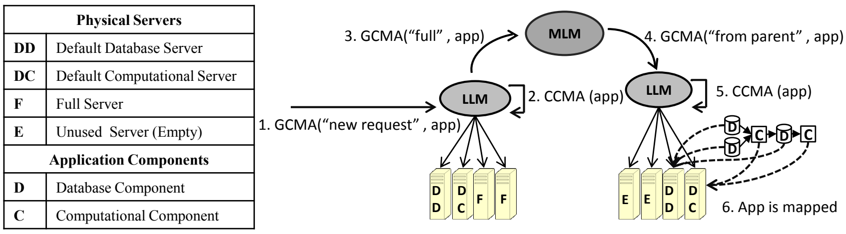


Figure 5.4: Different messages for interacting between the managers in GCMA (GCMA: Global Cloud Mapping Algorithm, CCMA: Centralized Cloud Mapping Algorithm).

algorithm 16: The Hierarchical Cloud Mapping Algorithm (HCMA), run on the hierarchical management planes.

input: Applications, Physical Infrastructure (Servers, Links), Management Plane (|S|, |SS|, |ML|);
for *($app \in applications$)* **do**
  manager← Current active LLM;
  GCMA(*app*, *manager*, *null*, *"new request"*);
end

5.6 Evaluation Details

The implemented physical cloud system is a tree-based multi-tier infrastructure, similar to current datacenter topologies [11, 12, 56], consisting of server nodes and links which we assume to be homogeneous. This means that all the servers have similar configuration of CPU, memory, storage and transmission medium (in terms of bandwidth and delay). In our evaluations, we make use of two component types. Each server can be either a database or a computational server. No backup servers are assumed. Servers are located in the lowest tier (level = 0) and the other levels consist of intermediate devices such as switches. In order to design the physical infrastructure, the number of server nodes (|S|) and the number of levels (|L|) are taken as inputs. In order to have the desired scale, these variables can be tuned. This physical cloud environment is a complete *N-ary* tree in which the *N* is the calculated branch factor (β). In addition, the number of switch ports and the number of tiers determine the number of network switches. The branch factor of each tier and the number of intermediate switches (|IS|) are calculated as follows.

algorithm 17: The Global Cloud Mapping Algorithm (GCMA), run on every manager in the hierarchical approach.

```
input: application a, manager m, requestSender r, message s
Impossibility_a ← false;
Currentstate ← Save the current system state;
while (Mapped_a = false & Impossibility_a = false) do
    Set current system state to CurrentState;
    if (m_type = LLM & s=("new request" OR "from the parent node")) then
        if (one of the default servers=null) then
            full_m ← true;
            GCMA(a, parent_m, m, "full");
        else
            CCMA(m,a);
        end
    end
    if (m_type ≠ LLM & s = ("full" OR "from the parent node" )) then
        for (ch ∈ children_m) do
            if (full_ch =false & ch≠r) then
                GCMA( a, ch, m, "from the parent node");
                return;
            end
        end
        full_m ← true;
        if (m_type = MLM) then
            GCMA (a,parent_m, m, "full");
        else
            Impossibility_a ←true;
        end
    end
    if (Impossibility_a =true) then
        Set current system state to CurrentState;
    end
end
```

The variables defined to describe physical cloud system are listed in Table 5.3.

$$\beta = \sqrt[|L|-1]{|S|} \tag{5.1}$$

$$|IS| = \sum_{level=1}^{|L|-1} \beta^{((|L|-1)-level)} \tag{5.2}$$

In our evaluations, three types of applications, shown in Table 5.6, are implemented. Type 1 refers to the 5-component applications with 3 database and 2 computational components, Type 2 refers to the 10-component applications with 7 database and 3 computational components and Type 3 refers to 20-component ap-

Table 5.3: The physical network parameters.

Variable	Description
$\lvert S\rvert \in \mathbb{N}^+$	Number of physical servers.
$\lvert IS\rvert \in \mathbb{N}^+$	Number of intermediate switches.
$\lvert L\rvert \in \mathbb{N}^+$	Number of physical switching levels.
$\beta \in \mathbb{N}^+$	physical level branch factor.

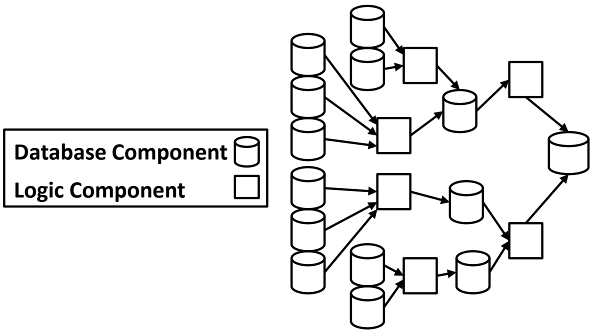


Figure 5.5: An illustrative $20-$*component application (Type 3).*

plications that consisting of 14 database and 6 computational components. These types of applications have been provided by our industrial partners based on realistic applications with deterministic characteristics, which implies that the structure of the applications is always known beforehand. To illustrate the used applications, a 20-component application is shown as a sample in Figure 5.5. Throughout this section, the number of X-component applications refer to the number of applications, submitted for a possible placement to the cloud network management system. We assume that the application are either rejected or placed in full with all X components.

This section is divided into four parts. Our proposed CCMA approach combines a set of requirements including network awareness, anti-collocation and full deployment placement constraints, which are not supported by the state-of-the-art solutions presented in literature. This makes it difficult to accurately compare our results to existing methods as the alternative network-aware solutions focus on different aspects, such as migration [25], investigation of traffic pattern [38], energy efficiency [57], SLA-awareness [58], etc. Therefore, we compare the performance of CCMA to a generic network-aware method, in which anti-collocation characteristics of applications is ignored. In the evaluation cases, we refer to this solution as ACUNA (Anti-Collocation Unaware, Network-Aware). Then, to provide an accurate validation, an evaluation of the CCMA is provided by comparing to the ILP-based optimal solution which takes all these requirements into account. Next,

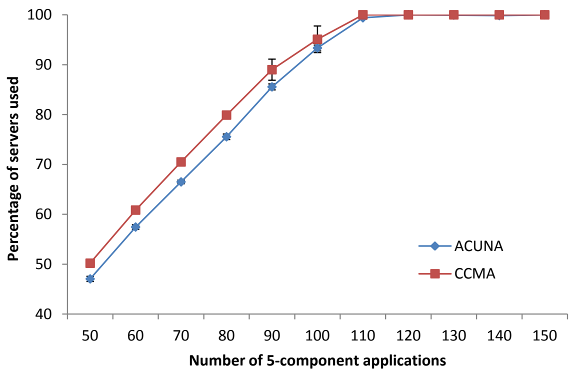


Figure 5.6: Comparing the percentage of servers used in the CCMA and the ACUNA algorithm as a function of number of application placement requests (20 iterations).

we evaluate the HCMA by comparing its performance with the CCMA. Finally, we will end the section with a large-scale evaluation of the HCMA.

The simulations are performed using the Stevin Supercomputer Infrastructure at Ghent University, containing quad core Intel Xeon L5420 servers with 16 GB RAM.

5.6.1 Comparing CCMA to the state-of-the-art solutions

5.6.1.1 Evaluation Set up

For this evaluation, we consider small 5-component applications and a 3-tier network architecture consisting of 100 servers and 10 intermediate nodes. The number of applications varies from 50 up to 150. The experiments are iterated 20 times and the percentages of serves used, mapped application and anti-collocation constraint fulfillment are captured.

5.6.1.2 Evaluation Results

Our evaluation in Figures 5.6, 5.7 and 5.8 shows that although generic network-aware approach is able to map up to 5.75% more applications and up to 4.35% lower number of servers, at least in 66.8% of evaluation cases the anti-collocation requirement of mapped applications are violated.

5.6.2 Comparing the CCMA to the ILP-based algorithm

5.6.2.1 Evaluation Set up

The optimal model and the CCMA are evaluated with a configuration of 6 servers arranged in a star topology. The specification of servers and links can be seen in

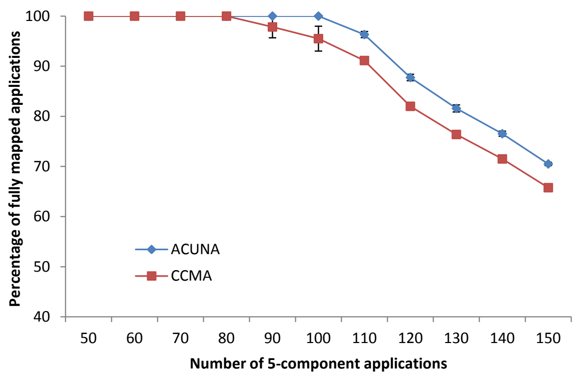


Figure 5.7: Comparing the percentage of fully mapped applications in the CCMA and the ACUNA algorithm (20 iterations).

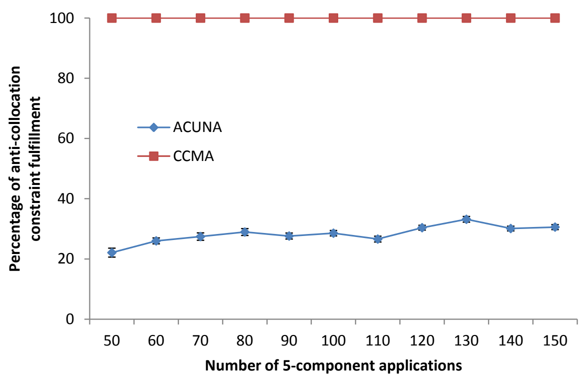


Figure 5.8: Comparing the percentage of anti-collocation application placement fulfillment in the CCMA and the ACUNA algorithm (20 iterations).

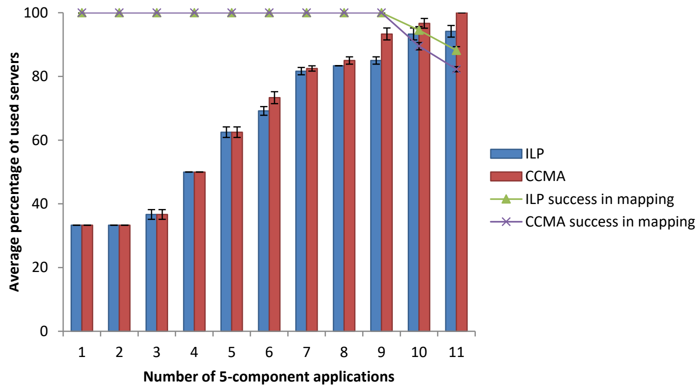


Figure 5.9: Comparing the number of servers used (as bar charts) and the application mapping success rate (as line charts) in the CCMA to the ILP-based algorithm for 5-component applications (20 iterations).

Table 5.4.

Type 1 and Type 2 applications are used to compare the performance of the proposed algorithms under light and heavy network load conditions. The number of scenarios varies from 1 up to 11. The experiments are iterated 20 times and the average percentage of used servers and percentage of algorithm success in mapping all the applications are captured.

5.6.2.2 Evaluation Results

In Figure 5.9 and 5.10 the CCMA is compared to the ILP-based optimal approach. The percentage of servers used are depicted in bars and the percentage of algorithm success in mapping all the application components are shown in lines.

As can be seen in both figures, when it comes to the physical resources usage, the CCMA provides a near-optimal solution compared to the ILP-based algorithm in this scenario. This can be clearly observed from Figure 5.9 as the network is not saturated. This figure show that in 5 out of 11 experiments the number of used servers are equal in the CCMA and the ILP-based approach and the CCMA uses at most 8.33 more number of servers when the number of applications is 9. In Figure 5.10, the percentage of algorithm success in mapping all the applications is more interesting. This figure reveals that when the network is saturated the capability of the CCMA in mapping application components stays within 10.7% of the optimal approach. This figure also shows that when both algorithms succeed in mapping all the application components (from 1 up to 4 number of applications), the CCMA uses almost the same number of servers as the optimal ILP-based approach. These results show the performance of the CCMA is close to that of the

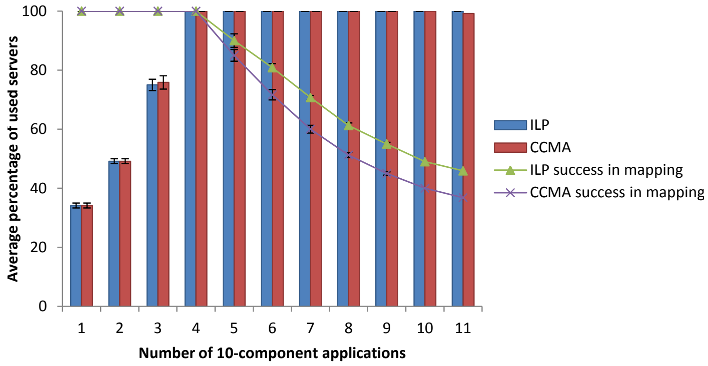


Figure 5.10: Comparing the number of servers used (as bar charts) and the application mapping success rate (as line charts) in the CCMA to the ILP-based algorithm for 10-component applications (20 iterations).

Table 5.4: The Physical Infrastructure Specifications.

Physical Infrastructure Specifications				
Case study	**|S|**	**|IS|**	**|L|**	**|SP|**
1	1000	111	4	10
2	4096	273	4	16
Physical Server Specifications			**Physical Link Specifications**	
CPU	**Storage**	**Memory**	**bandwidth**	**Delay**
$3GHZ$	$200GB$	$16GB$	$400Mbps$	$3ms$

optimal algorithm.

The execution times of the CCMA and the ILP-based approaches are compared in Figure 5.11 for Type 1 applications. As can be seen, the execution time of the ILP-based model is exponentially increasing by adding more applications, which makes it inappropriate for larger evaluations. As such, the remainder of this section is devoted to the comparison of the CCMA and the HCMA algorithms. Throughout the next subsections, the HCMA(*XX,YY*) refers to a three-tier management plane of *XX* LLMs and *YY* MLMs and the HCMA(*XX*) refers to a two-tier management plane of *XX* LLMs. In a two-tier management plane no MLM is involved and one RLM is taken into account in all experiments.

5.6.3 Comparing the hierarchical algorithm to the centralized approach

We study three case studies. In the first case, 5-component applications are placed on a cloud system with 1000 servers and the second case considers a larger sce-

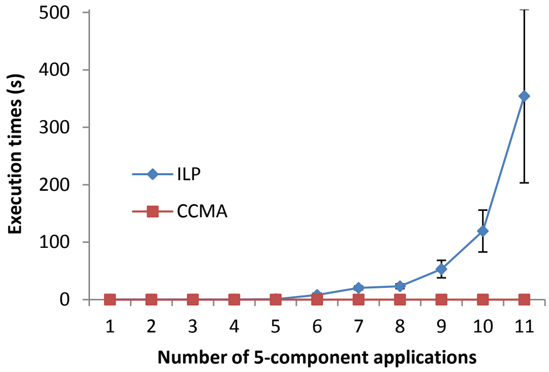


Figure 5.11: Comparing the execution times of the CCMA to the ILP-based algorithm for 5-component applications (20 iterations).

Table 5.5: Management plane infrastructure.

Case study	type	|ML|	|SS|	|LLM|	|MLM|	|RLM|	μ
1	CCMA	1	1000	1	0	0	-
	HCMA	3	10	100	10	1	10
	HCMA	3	40	25	5	1	5
	HCMA	2	100	10	0	1	10
2	CCMA	1	4096	1	0	0	-
	HCMA	3	16	256	10	1	16
	HCMA	3	64	64	5	1	8
	HCMA	2	256	16	0	1	16

Table 5.6: Application specifications.

Type	# component	# link	# database	# computational	
1	5	4	3	2	
2	10	9	7	3	
3	20	19	14	6	
Type	**Component demands(Random)**			**Link demands(Random)**	
	CPU	**Storage**	**Memory**	**Delay**	**BW**
1	(1-1000)MHZ	(1-20000)MB	(1-2000)MB	1s	(1-50)Mbps
2	(100-500)MHZ	(100-20000)MB	(100-1000)MB	1s	(1-50)Mbps
3	(1-200)MHZ	(1-10000)MB	(1-300)MB	1s	(1-20)Mbps

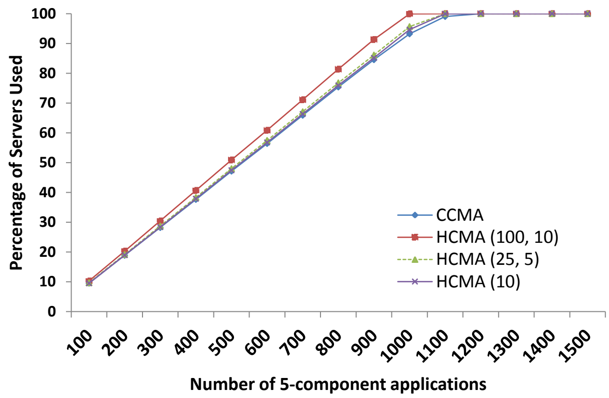


Figure 5.12: The percentage of servers used (Case study 1 with 1000 servers and 20 iterations).

nario with 4096 servers and 20-component applications. In the experiments, we measure the percentage of servers used, the percentage of mapped applications and the execution times per application. Afterward, the impact of different physical infrastructures on the average number of fully mapped applications and the execution time for 1000 up to 4096 servers are analyzed. Due to negligible standard errors for the reminder of evaluations, standard error bars are left out.

5.6.3.1 Evaluation Set up

The configuration of the simulated network, the management plane and the application structure are as follows. For the evaluation, the configuration of physical infrastructure is considered to be a 4-tier hierarchical tree topology. For the first scenario, the physical cloud system consists of 1000 servers (respectively 4096 for case study 2) in the lowest tier. The number of ports in each intermediate device is 10 (resp. 16) which results in 1+10+100 (resp. 1+16+256) switches in the first three tiers. Consequently, the number of physical nodes is 1111 (resp. 4369) in the entire cloud system. The specifications of the physical cloud resources are shown in Table 5.4.

To make a better comparison apart from the central management system, three different hierarchical management planes are generated. The hierarchical management planes are defined as follows and are listed in Table 5.5.

- a 3-tier management plane with 100 (resp. 256) LLMs, 10 (resp. 16) MLMs and 1 RLM. Each LLM supports 10 (resp. 16) servers in this case.
- a 3-tier management plane with 25 (resp. 64) LLMs, 5 (resp. 8) MLMs and 1 RLM. In this scenario 40 (resp. 64) servers are supported by each LLM.

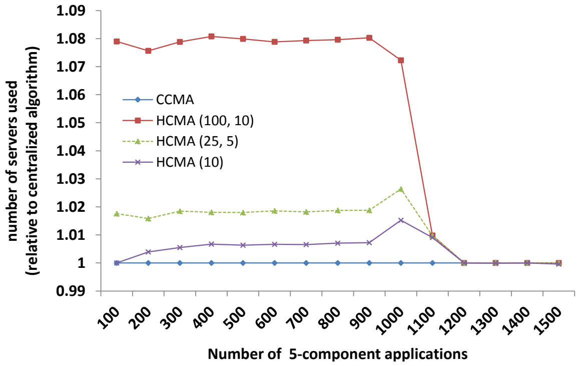


Figure 5.13: The percentage of used servers relative to the CCMA (Case study 1 with 1000 servers and 20 iterations).

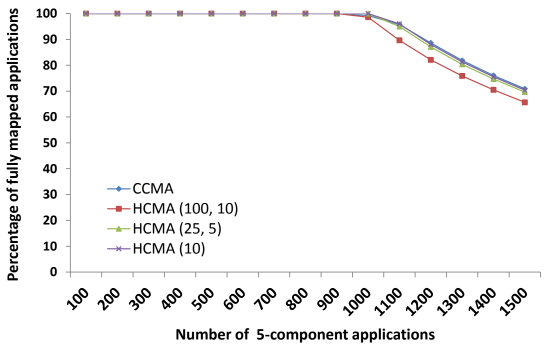


Figure 5.14: The percentage of fully placed applications (Case study 1 with 1000 servers and 20 iterations).

- a 2-tier management plane with 10 (resp. 16) LLMs, no MLM and 1 RLM. Each administrative domain consists of 100 (resp. 256) servers here.

The implemented applications are of Type 1 (resp. Type 3), the number of which varies from 100 up to 1500 (resp. 400 up to 4000). Each application component has different CPU, memory, storage and QoS demands which are randomly taken within a predefined interval, provided in Table 5.6.

5.6.3.2 Evaluation Results

Figure 5.12 and Figure 5.13 show the percentage of used servers for different management planes. As can be observed from Figure 5.12, the number of used servers grows linearly with the number of applications until all the resources are completely occupied. Among all, the CCMA uses the fewest and the HCMA with

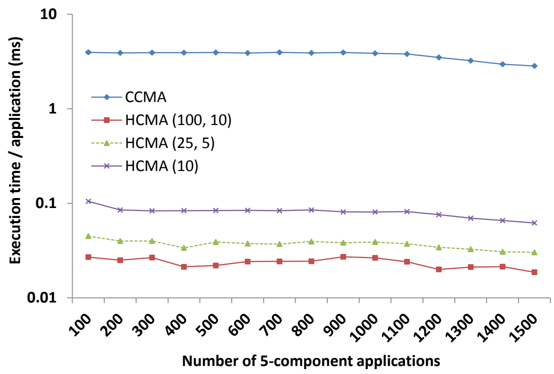


Figure 5.15: The percentage of fully placed applications (Case study 1 with 1000 servers and 20 iterations).

higher numbers of LLMs, uses the highest percentage of servers. In the worst case the hierarchical scenario with 100 LLMs uses 6.7% more servers. Moreover, the average standard errors is 0.025% for the CCMA and 0.031% on average for the HCMAs.

In Figure 5.14, the percentages of placed applications is depicted. As the results show, the CCMA offers the best performance and the HCMA with 100 LLMs the worst. Additionally, application placement failures are expected due to the fixed number of servers and resource saturation after 1000 applications. Nonetheless, in both figures even in the worst case, the result is within 8% of the best result.

The execution time of the hierarchical approaches is promising. As can be clearly seen in Figure 5.15, the time in which an application is placed in the CCMA is much higher than the hierarchical approaches, especially in the hierarchical management plane with more LLMs.

In Figure 5.16, Figure 5.17, Figure 5.18 and Figure 5.19 the percentage of servers used, the percentage of mapped applications, the percentage of mapped applications relative to the centralized approach and the average execution time per application are depicted respectively for the second case study. As can be observed from Figure 5.16, the percentage of used servers increases by adding more applications up to when the servers are fully occupied. Afterwards, the percentage of mapped applications declines as the newly arriving applications are immediately rejected due to saturated resources. Although the CCMA shows better performance, the hierarchical management planes use at most 5.6% more resources. Figure 5.17 and Figure 5.18 compare the percentage of mapped applications in the hierarchical approaches to the centralized solution. As can be seen, in the worst case the result of the hierarchical management planes is within 7% of the best

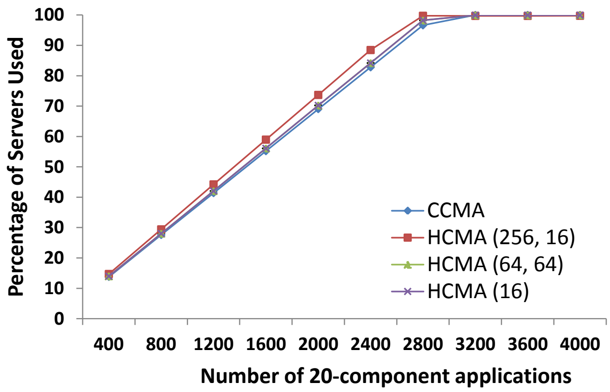


Figure 5.16: The percentage of servers used (Case study 2 with 4096 servers and 20 iterations.)

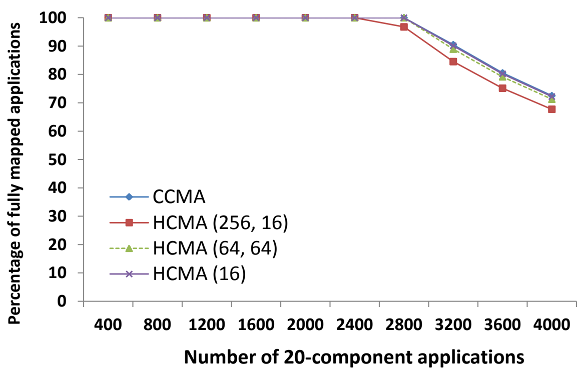


Figure 5.17: The percentage of fully placed applications (Case study 2 with 4096 servers and 20 iterations).

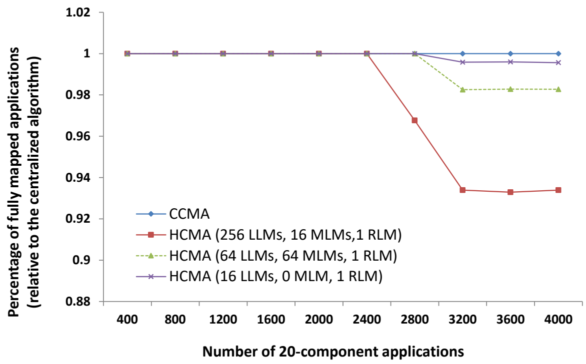


Figure 5.18: The percentage of fully placed applications relative to the CCMA (Case study 2 with 4096 servers and 20 iterations).

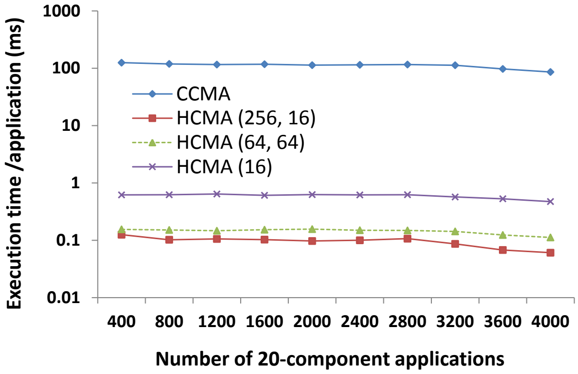


Figure 5.19: The percentage of servers used (Case study 2 with 4096 servers and 20 iterations).

result. Also, Figure 5.19 indicates that the execution time of the CCMA is high compared to the hierarchical scenarios.

We also evaluated the execution time and the number of fully mapped applications in different physical cloud systems with different numbers of servers and different numbers of switch ports. The applications are of Type 1 based on Table 5.6. The number of servers and the number of intermediate switches are provided in Table 5.7 and the implemented management planes are presented in Table 5.8.

In Figure 5.20 the number of fully mapped applications is depicted. As the branch factor (β) and consequently the number of servers increases, the number of mapped applications grows. This evaluation shows that the HCMA with β LLMs is able to achieve the same performance of the CCMA, in terms of the number of

Table 5.7: The number of physical devices based on different β values.

β	10	11	12	13	14	...	25
|S|	1000	1331	1728	2197	2744	...	15625
|IS|	101	122	145	170	197	...	626
β	**20**	**30**	**40**	**50**	**60**	**70**	**80**
|S|	$8K$	$27K$	$64K$	$125k$	$216K$	$343K$	$512K$
|IS|	401	901	1601	2501	3601	4901	6401

Table 5.8: The management plane parameters based on different β values.

Type	|ML|	|SS|	|LLM|	|MLM|	|RLM|
CCMA	1	β^3	1	0	0
HCMA (β)	3	β^2	β	0	1
HCMA ($\beta * \beta, \beta$)	3	β	β^2	β	1

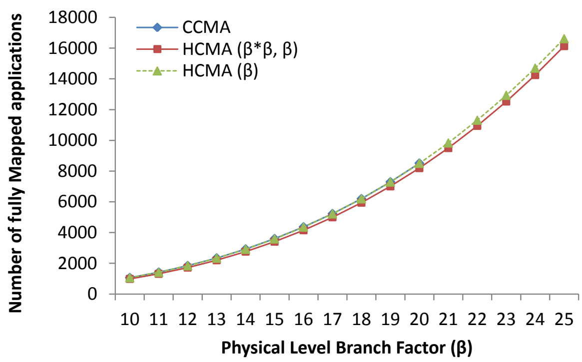


Figure 5.20: The percentage of fully placed applications (20 iterations). Number of physical servers = β^3.

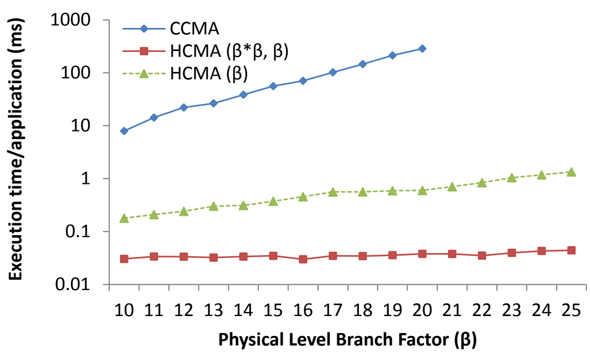


Figure 5.21: The execution time per application (20 iterations). Number of physical servers = β^3.

fully mapped applications (with only 9 fewer applications on average). However, comparing to the HCMA with β^2 LLMs, the CCMA is able to map on average 6.2% more applications. In Figure 5.21, the execution time of the different approaches are shown. As can be seen the execution time of the CCMA dramatically grows when the number of servers increases which makes the centralized algorithms inefficient in large scale cloud systems. Due to the increasing execution duration, we stop executing the CCMA once $\beta = 20$, indicating that the CCMA approach is not appropriate for a network larger than 8000 servers. Instead in this evaluation, the HCMA with β number of low level managers has made a desired trade-off between the quality of application mapping and the execution time.

5.6.4 Large scale scenarios

In this phase, we focus on the scalability of the presented algorithms. We extend the scale of the experiments up to 512000 servers and more than 540000 5-component applications. In these experiments, the number of fully mapped applications is evaluated and the execution time per application is captured. The results are the average value of 10 experiments.

5.6.4.1 Evaluation Set up

The experiments are conducted for an increasing number of servers from 1000 up to 512000 servers. The assumptions of the applications, of the physical networks and of the management planes are provided in Table 5.6, Table 5.7 and Table 5.8 respectively.

5.6.4.2 Evaluation Results

Figure 5.22 compares the number of fully mapped applications for two different hierarchical management plane architectures. The numbers of successfully mapped applications are close, but the management plane with a larger number of supported servers in each administrative domain allocates on average 3.4% more applications. Nonetheless, while the execution time of this approach grows exponentially, the HCMA with more LLMs shows better performance, as can be clearly seen in Figure 5.23. As a result, for experiments larger than 125000 servers, only the second hierarchical architecture is evaluated. This evaluation shows that the management architecture with β^2 LLMs is the most appropriate management plane for very large datacenters.

5.6.5 Evaluation discussion

We have extensively assessed the CCMA and HCMA approaches. Our evaluation studies show that the best performance is constantly achieved by the cen-

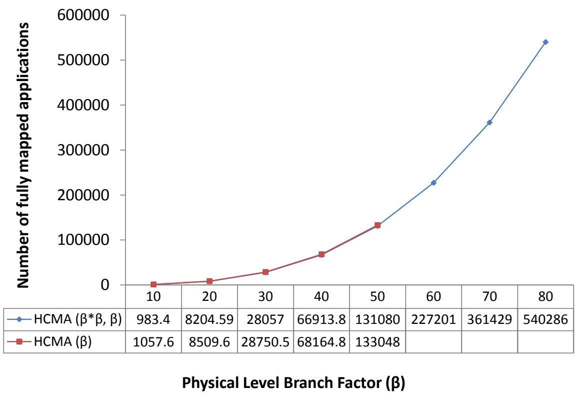


Figure 5.22: The number of fully placed applications (10 iterations). Number of physical servers = β^3.

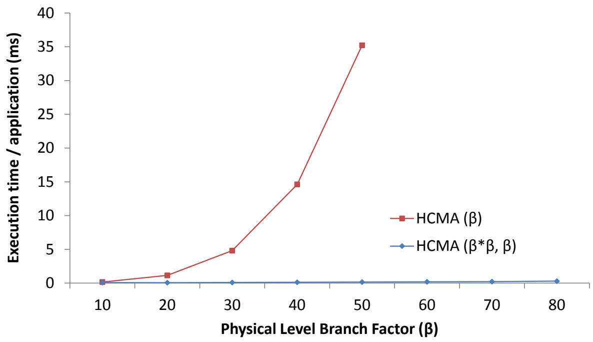


Figure 5.23: The execution time per application (10 iterations). Number of physical servers = β^3.

tralized CCMA approach, compared to the hierarchical management planes, in terms of percentage of application placement and network resource usage. However, the execution time of CCMA dramatically grows when the number of servers increases. This makes the centralized algorithms inefficient in large scale cloud systems. While the CCMA approach is not appropriate for a network larger than $8k$ servers (enough capacity to fully map 8512 small applications), HCMA with β number of LLMs has made a desired trade-off between the quality of application mapping and the execution time. Moreover, a larger scale evaluation reveals that although the HCMA with β LLMs is able to achieve the same performance of the CCMA, in terms of the number of fully mapped applications, this hierarchical architecture shows limited scalability up to $125k$ servers with $133k$ fully mapped applications. Our large-scale evaluation case studies indicate that the management architecture with β^2 LLMs is the most appropriate management plane for very large datacenters ($512k$ servers and more than 2.7 million application components).

5.7 Conclusions

This chapter focused on the problem of component-level application placement in large-scale cloud environments. Our approach takes the characteristic of the underlying network into account and works with multi-component applications, taking into account the application workflow with a distinction between application component types. To offer an optimal solution, we first presented an ILP-based model and to have a scalable solution, a near-optimal centralized approach was proposed and compared to the optimal solution. Due to limited scalability of the centralized approaches, a hierarchical heuristic was also designed to be deployed in large-scale cloud management systems. The experimental results showed that in large-scale clouds our proposed approach works efficiently compared to a centralized and optimal management systems in terms of resource usage and quality of application placement. The percentage of nodes used and the percentage of mapped applications remain close to that of the centralized algorithm, in the worst case within 6.7% and 8% respectively.

Acknowledgment

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government - department EWI. The work is also partly supported by the iMinds DMS2 project and the FP7 NoE FLAMINGO project.

References

[1] Y. Li, F.-H. Chen, X. Sun, M.-H. Zhou, W.-P. Jiao, D.-G. Cao, and H. Mei. *Self-adaptive resource management for large-scale shared clusters*. Journal of Computer Science and Technology, 25(5):945–957, 2010.

[2] C. P. Chen and C.-Y. Zhang. *Data-intensive applications, challenges, techniques and technologies: A survey on Big Data*. Information Sciences, 275:314–347, 2014.

[3] L. Shi, B. Butler, R. Wang, D. Botvich, and B. Jennings. *Optimal placement of virtual machines with different placement constraints in IAAS clouds*. In ICT and Energy Efficiency and Workshop on Information Theory and Security (CIICT 2012), Symposium on, pages 35–40. IET, 2012.

[4] B. Urgaonkar, A. L. Rosenberg, and P. Shenoy. *Application placement on a cluster of servers*. International Journal of Foundations of Computer Science, 18(05):1023–1041, 2007.

[5] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. *Towards predictable datacenter networks*. In ACM SIGCOMM Computer Communication Review, volume 41, pages 242–253. ACM, 2011.

[6] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. *The only constant is change: incorporating time-varying network reservations in data centers*. ACM SIGCOMM Computer Communication Review, 42(4):199–210, 2012.

[7] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari. *Server workload analysis for power minimization using consolidation*. In Proceedings of the 2009 conference on USENIX Annual technical conference, pages 28–28. USENIX Association, 2009.

[8] Z. Usmani and S. Singh. *A Survey of Virtual Machine Placement Techniques in a Cloud Data Center*. Procedia Computer Science, 78:491–498, 2016.

[9] C. Pham, N. H. Tran, M. N. Nguyen, J. H. Son, and C. S. Hong. *Hosting Virtual Machines on Distributed Datacenters*. In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication, page 85. ACM, 2016.

[10] R. P. Esteves, L. Z. Granville, H. Bannazadeh, and R. Boutabai. *Paradigm-based adaptive provisioning in virtualized data centers*. In 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pages 169–176. IEEE, 2013.

[11] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani. *Data center network virtualization: A survey*. IEEE Communications Surveys & Tutorials, 15(2):909–928, 2013.

[12] B. Jennings and R. Stadler. *Resource management in clouds: Survey and research challenges*. Journal of Network and Systems Management, 23(3):567–619, 2015.

[13] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. *A scalable application placement controller for enterprise data centers*. In Proceedings of the 16th international conference on World Wide Web, pages 331–340. ACM, 2007.

[14] T. Kimbrel, M. Steinder, M. Sviridenko, and A. Tantawi. *Dynamic application placement under service and memory constraints*. In Experimental and Efficient Algorithms, pages 391–402. Springer, 2005.

[15] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé. *Utility-based placement of dynamic web applications with fairness goals*. In IEEE Network Operations and Management Symposium (NOMS), pages 9–16. IEEE, 2008.

[16] Z. Zhou, Z. Hu, and K. Li. *Virtual Machine Placement Algorithm for Both Energy-Awareness and SLA Violation Reduction in Cloud Data Centers*. Scientific Programming, 2016, 2016.

[17] F. Wuhib, R. Stadler, and M. Spreitzer. *Gossip-based resource management for cloud environments (long version)*. KTH Royal Institute of Technology, Tech. Rep, 2010.

[18] A. Nathani, S. Chaudhary, and G. Somani. *Policy based resource allocation in IaaS cloud*. Future Generation Computer Systems, 28(1):94–103, 2012.

[19] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs. *Resource allocation for distributed cloud: concepts and research challenges*. Network, IEEE, 25(4):42–46, 2011.

[20] M. Masdari, S. S. Nabavi, and V. Ahmadi. *An overview of virtual machine placement schemes in cloud computing*. Journal of Network and Computer Applications, 66:106–127, 2016.

[21] Q. Zhang, M. Li, and X. Hu. *Network traffic-aware virtual machine placement with availability guarantees based on shadows*. In Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pages 542–543. IEEE, 2014.

[22] Z. Zhuang and C. Guo. *OCPA: An Algorithm for Fast and Effective Virtual Machine Placement and Assignment in Large Scale Cloud Environments*. In Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on, pages 254–259. IEEE, 2013.

[23] K.-y. Chen, Y. Xu, K. Xi, and H. J. Chao. *Intelligent virtual machine placement for cost efficiency in geo-distributed cloud systems*. In Communications (ICC), 2013 IEEE International Conference on, pages 3498–3503. IEEE, 2013.

[24] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen. *Reducing electricity cost through virtual machine placement in high performance computing clouds*. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, page 22. ACM, 2011.

[25] J. T. Piao and J. Yan. *A network-aware virtual machine placement and migration approach in cloud computing*. In Grid and Cooperative Computing (GCC), 2010 9th International Conference on, pages 87–92. IEEE, 2010.

[26] L. Shi, B. Butler, D. Botvich, and B. Jennings. *Provisioning of requests for virtual machine sets with placement constraints in IaaS clouds*. In 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pages 499–505. IEEE, 2013.

[27] D. Breitgand and A. Epstein. *SLA-aware placement of multi-virtual machine elastic services in compute clouds*. In 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pages 161–168. IEEE, 2011.

[28] J. Xu and J. A. Fortes. *Multi-objective virtual machine placement in virtualized data center environments*. In Green Computing and Communications (GreenCom), IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom), pages 179–188. IEEE, 2010.

[29] C. Adam and R. Stadler. *Service middleware for self-managing large-scale systems*. IEEE Transactions on Network and Service Management, 4(3):50–64, 2007.

[30] M. Korupolu, A. Singh, and B. Bamba. *Coupled placement in modern data centers*. In IEEE International Symposium on Parallel & Distributed Processing (IPDPS), pages 1–12. IEEE, 2009.

[31] G. Foster, G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer. *The Right Tool for the Job: Switching data centre management strategies at runtime*. In

2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pages 151–159. IEEE, 2013.

[32] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera. *A stable network-aware vm placement for cloud systems*. In Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid), pages 498–506. IEEE Computer Society, 2012.

[33] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang. *Joint VM placement and routing for data center traffic engineering*. In INFOCOM, 2012 Proceedings IEEE, pages 2876–2880. IEEE, 2012.

[34] T. Benson, A. Akella, A. Shaikh, and S. Sahu. *CloudNaaS: a cloud networking platform for enterprise applications*. In Proceedings of the 2nd ACM Symposium on Cloud Computing, page 8. ACM, 2011.

[35] L. Hu, K. D. Ryu, D. Da Silva, and K. Schwan. *v-bundle: Flexible group resource offerings in clouds*. In Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on, pages 406–415. IEEE, 2012.

[36] G. Koslovski, S. Soudan, P. Gonçalves, and P. Vicat-Blanc. *Locating virtual infrastructures: users and InP perspectives*. In 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pages 153–160. IEEE, 2011.

[37] M. F. Zhani, Q. Zhang, G. Simona, and R. Boutaba. *Vdc planner: Dynamic migration-aware virtual data center embedding for clouds*. In 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pages 18–25. IEEE, 2013.

[38] X. Meng, V. Pappas, and L. Zhang. *Improving the scalability of data center networks with traffic-aware virtual machine placement*. In INFOCOM, 2010 Proceedings IEEE, pages 1–9. IEEE, 2010.

[39] M. Alicherry and T. Lakshman. *Network aware resource allocation in distributed clouds*. In Infocom, 2012 proceedings IEEE, pages 963–971. IEEE, 2012.

[40] X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. Mckee, C. Hyser, D. Gmach, R. Gardner, et al. *1000 islands: an integrated approach to resource management for virtualized data centers*. Cluster Computing, 12(1):45–57, 2009.

[41] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh. *A cyber-physical systems approach to energy management in data centers*. In Proceedings of the

1st ACM/IEEE International Conference on Cyber-Physical Systems, pages 168–177. ACM, 2010.

[42] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu. *Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures*. In Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, pages 62–73. IEEE, 2010.

[43] A. Beloglazov and R. Buyya. *Energy efficient resource management in virtualized cloud data centers*. In Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing, pages 826–831. IEEE Computer Society, 2010.

[44] B. Viswanathan, A. Verma, and S. Dutta. *CloudMap: workload-aware placement in private heterogeneous clouds*. In 2012 IEEE Network Operations and Management Symposium, pages 9–16. IEEE, 2012.

[45] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible. *Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement*. In Services Computing (SCC), 2011 IEEE International Conference on, pages 72–79. IEEE, 2011.

[46] H. Moens, J. Famaey, S. Latre, B. Dhoedt, and F. De Turck. *Design and evaluation of a hierarchical application placement algorithm in large scale clouds*. In Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, pages 137–144. IEEE, 2011.

[47] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck. *Hierarchical Network-Aware Placement of Service Oriented Applications in Clouds*. Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS), 2014.

[48] M. Barshan, H. Moens, S. Latre, and F. De Turck. *Algorithms for efficient data management of component-based applications in cloud environments*. In IEEE Network Operations and Management Symposium (NOMS), pages 1–8. IEEE, 2014.

[49] M. Barshan, H. Moens, and F. De Turck. *Design and evaluation of a scalable hierarchical application component placement algorithm for cloud resource allocation*. In 10th International Conference on Network and Service Management (CNSM), pages 175–180. IEEE, 2014.

[50] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. *Jellyfish: Networking data centers randomly*. In Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pages 225–238, 2012.

[51] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. *Dcell: a scalable and fault-tolerant network structure for data centers*. ACM SIGCOMM Computer Communication Review, 38(4):75–86, 2008.

[52] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma. *Application-driven bandwidth guarantees in datacenters*. In ACM SIGCOMM Computer Communication Review, volume 44, pages 467–478. ACM, 2014.

[53] L. Yu and H. Shen. *Bandwidth guarantee under demand uncertainty in multi-tenant clouds*. In IEEE 34th International Conference on Distributed Computing Systems (ICDCS), pages 258–267. IEEE, 2014.

[54] I. AMPL. *CPLEX software*. ILOG website: www. ilog. com/products/cplex.

[55] T. Cormen. *Introduction to Algorithms*. MIT Press, 2009. Available from: http://books.google.be/books?id=Jwr8jwEACAAJ.

[56] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya. *Network-Aware Virtual Machine Placement and Migration in Cloud Data Centers*. Emerging Research in Cloud Distributed Computing Systems, 42, 2015.

[57] A. Beloglazov, J. Abawajy, and R. Buyya. *Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing*. Future generation computer systems, 28(5):755–768, 2012.

[58] J. Zhang, Z. He, H. Huang, X. Wang, C. Gu, and L. Zhang. *SLA aware cost efficient virtual machines placement in cloud computing*. In Performance Computing and Communications Conference (IPCCC), 2014 IEEE International, pages 1–8. IEEE, 2014.

6 Conclusions and future work

In this dissertation, several contributions to the field of cloud resource allocation and advance bandwidth reservation schemes in media-centric industries were presented. This chapter summarizes how this dissertation has dealt with unaddressed problems and challenges and also provides an outlook of future work. The main conclusion of the work comprised in this dissertation can be stated as follows:

6.1 Advance reservation in media-centric networks

In this dissertation, we first proposed optimal ILP-based advance reservation algorithms based on fixed timeslot sizes which operate in both offline (i.e., SARA) and online (i.e., DARA) manners. Based on simulation results, the viability of AR scheduling in media production networks was assessed. Results showed that when a significant portion of requests is known at the start of the day, AR significantly increases bandwidth efficiency and request admittance. In case all requests are known at the start of the day, request admittance can be increased up to 6.02% compared to when requests are only known one hour before their desired start time. Time granularity increases algorithm accuracy and optimality in terms of request admittance. However, it also affects the ILP problem size, resulting in an exponential execution time increase. Concretely, a time slot size of 20-min resulted in up to 4.4% more request admittance than one of 60-min. To resolve the high computational complexity and scalability issue of the optimal solution, we then proposed static and dynamic heuristic solutions and compared their performance to the ILP-based algorithms. Our evaluation studies showed that the proposed heuristic techniques 1) approach at least within 8.78% of the optimal admittance rate, 2) can achieve higher scalability in terms of the size of the physical network as well as the time slot sizes, 3) offer lower operational overhead in terms of problem complexity and execution time. The size of time intervals can be fine-grained up to 1-min. Moreover, in case all requests are known beforehand (SARA), request admittance of the heuristic solution can be increased by up to 5.7%.

6.2 Flexible and fixed time slot size AR approaches

In advance bandwidth reservation approaches, management of the time domain is of great importance. As an efficient solution, a timeslot-based approach is introduced, based on which the entire time span is discretized into a set of timeslots. Timeslot-based solutions can be implemented based on fixed or flexible timeslot sizes. Although the majority of timeslot-based proposed approaches in literature have followed the static solution, it is inefficient for advance reservation systems with a small number of reservation requests.

Analysis of the impact of flexible and fixed size timeslot-based advance bandwidth reservation in media-centric networks has been one of the main concerns of Chap. 4. We have thoroughly discussed the benefits and disadvantages of flexible and fixed size timeslot-based approaches for a media-centric advance reservation system, detailed as follows:

6.2.1 Fixed timeslot sizes

Using fixed time slot size approaches leads to regular and periodic reconfigurations of network routers and switches. The number, start time and duration of timeslots is not altered, even in highly dynamic network traffic conditions. The size of the timeslots can be set to a suitable value in networks with predictable traffic patterns. This makes fixed-size timeslot approaches appealing for network managers due to the easy implementation and predictable performance.

However, we have distinguished the factors which restrict the capabilities of fixed timeslot-based approach and make the variable time slot sizes more suitable for advance bandwidth reservation in media-centric networks. These factors are related to the characteristics of requests in media production industries and nature of fixed timeslot-based advance reservation approaches in general.

6.2.1.1 Request characteristics in media production industries

Type of requests: In fixed timeslot-based approach, the duration of reservations has to be tuned to the size of the timeslots. Since in our fixed timeslot-based approach the amount of allocated bandwidth is unchangeable within each time interval, for the streams the reservations have to be made from the start of the timeslot in which the start time of a request fits, till the end of the timeslot to which the request's end time belongs. However, for file-based transfers, the reservation has to be started after the time when the file is ready to be transferred and it must be finished by the request's deadline, so the start of reservation for a file is restricted to the beginning of the next timeslot and the start of the timeslot in which the request deadline fits. This restriction implies that the file has a tighter time opportunity for transmission and therefore the probability of timely transfer is decreased. Contrary to this, the use of flexible time windows can eliminate these restrictions for both request types. Regardless of the type of request, the start and the end of time windows can be tuned up to the start and end time of each request.

Dependencies among requests: The second point is that the fixed size of timeslots is more restrictive when there are dependencies among different transfers. We showed that the flexible timeslot-based approach can improve the result of advance reservation scheduling approaches when dependencies among requests exist.

6.2.1.2 Predefined size of fixed-size time slots

The size of time slots has a relatively high impact on the quality and complexity of the advance bandwidth reservation schedule. We showed that, although more timeslot size granularity increases the resource utilization in our proposed approach, the complexity of the algorithms significantly increases as well.

Quality of the schedule: We have analyzed how the size of the different time intervals, ranging from 1-min to 60-min, impacts the quality of the reservation system in media-centric networks. We showed that the longest timeslot size of 60-min shows the worst performance in terms of number of admitted requests and this quality is improved as the time interval size is more fine-grained. For example, for the 12-times shorter time interval size (i.e. 5-min), the SARA approach achieves up to 15.47% higher percentage of admitted requests.

Execution time for producing the schedule: The fixed-size algorithms have a high computational overhead, particularly with fine-grained timeslot sizes and large scale networks. The fine-grained experiment with shortest timeslot size results in the highest request admittance ratio. However, the execution time of the algorithm also increases. The SARA approach with 60-min timeslot granularity is between 12.3 up to 16.7 times faster than the solution with 5-min timeslot sizes.

Optimized timeslot size: In the fixed size timeslot-based approach, we need to make an informed decision on the optimal size of the timeslots, which is not an issue with variable time window. We showed that a time slot size of 5-min optimizes the trade-off between optimality and complexity of the solution. Although based on the results for this timeslot size, we stay within 1.6% of the optimum in all evaluated cases, this was not the most optimal value for all possible configurations. This is a challenging issue as other values for the size of timeslots might suit other evaluation scenarios better. The available network capacity is an important factor in low-demand networks. As such, longer timeslot sizes are preferred as long as all the requests can be scheduled.

Delay prior to request processing: Newly submitted requests to the fixed timeslot-based advance bandwidth reservation system have to wait till the beginning of the next timeslot for processing. The maximum wait time depends on the size of the timeslots. The bigger the timeslot size, the higher the potential processing delay.

Unnecessary periodic computations for long transfers: Another issue with fixed timeslot sizes relies on the periodic nature of these solutions. In fixed size advance bandwidth reservation approaches, the residual demand of ongoing requests are periodically updated at each timeslot, and new and updated requests are reallocated together. For long-term streaming requests and large files, this may lead to unnecessary periodic computations. The flexible timeslot-based approach shows higher network utilization because only the start time of a new request or the end of an ongoing transfer tears down the connection and creates new timeslot allocations.

6.2.2 Flexible timeslot sizes

We argued that due to the imposed restrictions of fixed size timeslot-based approach, the quality of schedules in flexible timeslot-based approaches should be higher when compared to fixed size approaches. We have however identified drawbacks of using flexible timeslot based approaches in 3 cases detailed below.

6.2.2.1 Dependency to the network load

In fixed size approaches, the number of timeslots stays unaffected when increasing the network load. To calculate the number of timeslots we need to know which factors play a role. Typically, timeslots are started with any request start time and end with either the arrival of a new request or the earliest end time of current requests. As such, worse case the number of time slots is twice of the number of requests. The dependency of the flexible approach on the number of requests could be detrimental in networks with growing number of resource reservations as it may lead to unpredictable complexity.

Unpredictable Complexity: The number of timeslots is a factor which directly increases the complexity and computational overhead of timeslot-based advance reservation systems. In flexible timeslot-based advance reservation approaches, the number of timeslots and the complexity of the schedule highly depends on the number of requests. In highly dynamic environments, the number of future timeslots is not predictable and therefore the complexity of the flexible timeslot-based algorithms is unmanageable. The benefit of using variable timeslot size approaches may be defeated due to excessive complexity and here is where the fixed-size timeslots plays its trump card: the complexity of the scheduling can be easily managed by consciously adjusting the timeslot size.

6.2.2.2 Irregular network devices' reconfiguration

Contrary to the fixed-size timeslots in which the number and duration of timeslot remains unchanged, flexible time window approach results in unpredictability of future timeslots. The number and duration of timeslots is frequently being adjusted during the scheduling process, as soon as new requests are admitted to be scheduled.

6.2.2.3 Impractical timeslot duration

The next negative feature of the flexible time slot approach comes to light when a large quantity of short-lived or low-demand requests with overlaps or with a very short time-based gap in between, needs to be scheduled. This leads to a considerable number of time slots with quite brief duration, which is impractical. For example, duration of transmission of a video file of 1GB, in a network with 10 Gbps leftover capacity, is 800 ms which is not a practical timeslot size in operational bandwidth reservation systems. It should be noted that this issue can be resolved by defining a threshold for minimum timeslot size.

6.2.3 Discussion

To sum up, we can conclude that the use of flexible time slots can improve the success rate of advance bandwidth reservation systems. However, the complexity of this approach should not be neglected. The flexible size timeslot-based approach is highly beneficial when the media-centric network deals with long-term downtimes and bursty traffic conditions. Burst traffic in this context is defined as large and high-bandwidth transfers over a short time period. Having said that, with large numbers of small-size file transfers and short-lived video streaming requests, not only the complexity of the scheduling process is unpredictable, but also the highly frequent reconfiguration of physical network devices is impractical or at least expensive. Therefore, fixed size timeslots can be considered as a convenient solution in such situations. The advantages and disadvantages of each approach are summarized in Table 6.1.

Table 6.1: The summary of the benefits and disadvantages of flexible and fixed-size timeslot based advance reservation approaches in media-centric industries.

	Benefits	Disadvantages
Fixed-size timeslots	• Predictable periodic reconfigurations of network devices • Independence number of timeslots • Easy to implement	• Quality and execution time dependency on timeslot size • Hard to find optimized timeslot size • Delay prior to request processing • Unnecessary periodic computations for long transfers • High computational complexity for long-term schedules
Flexible timeslots	• Compatibility with media network transfers • Suitable for low demand networks with bursty traffic • Higher expected quality	• Unpredictable complexity due to dependency on network load • Irregular network devices' reconfiguration • Impractical timeslot duration

6.3 Resilient advance reservation approaches

To enable a quick response to sudden changes such as failures in the network, we proposed a resilient AR scheduling approach in Chapter 3. We rely on a protection mechanism, which means that backup paths are found in advance, before any failure happens in the network. Using Shared Backup Path Protection (SBPP), the proposed scheme aims at minimizing the resource usage of backups, while it guarantees 100% recovery against any single link failure. This approach is further optimized in Chapter 4, resulting in a significant improvement in network utilization as follows: In the first version, for a given file transfer, if both primary and backup demands can not be fulfilled, the algorithm is repeatedly executed with 50% of primary demand until both demands are fulfilled or the request is rejected. We then proposed to make use of binary search instead of halving the primary demand and showed that this optimization improves the performance of the timeslot-based advance reservation system in terms of request admittance ratio. Based on our evaluations on the fixed size timeslot-based AR approach, we concluded that the optimized solution specifically performs well under limited network capacity and with fine-grained timeslot sizes. The optimized resilient approach outperforms the original one both in terms of the execution time, with 5-min timeslot size, and the percentage of admitted requests, up to 9.2%.

6.4 Impact of runtime adaptation approach

The resilient advance reservation approach enables the reservation system to deliver reliable and consistent performance in the presence of failures, however, using redundancy imposes significant performance overheads and extra costs. In order to mitigate these side-issues, we designed and evaluated a dynamic event-driven approach aiming to increase network utilization and request admittance ratio. In this dynamic approach, which we referred to as Runtime Adaptation (RA), a constant monitoring, adaptation and re-optimization is applied at runtime. We exploit underutilized network capacities to transfer more data than what had been scheduled as long as no failure is detected. This dynamic solution is a complementary component which can be used in conjunction with both fixed-size and flexible timeslot-based advance reservation approaches, shown in Chap. 3 and Chap. 4 respectively. We have extensively evaluated the impact of using the Runtime Adaptation approach and the experimental results showed that our approach works efficiently both in stable and failure-prone networks and deploying this approach will noticeably increase the performance of the advance reservation systems by increasing the number of succeeded requests. In stable network conditions, deploying this approach will noticeably increase the performance of the fixed size timeslot-based advance reservation system and percentage of admitted requests up to 13.9%. In failure-prone environments, the proposed approach leads to a significant increase in the success rate of admitted requests, up to 6.77 times, compared to the resilient advance reservation algorithms. With a processing time of less than 3ms for all evaluated cases, the Runtime Adaptation solution is fast enough to react immediately and re-configure the network in response to sudden changes.

We have concluded that the combination of the RA and flexible approaches is more challenging, compared to the combination with fixed size timeslots. In the fixed size approach, the use of RA only updates the reservations and the number and duration of timeslots are not affected during the RA process. However, in the flexible approach, the RA approach entirely affects the schedule, meaning that not only the reservations, but also the number and duration of timeslots are impacted. In addition, since the number of timeslots determines the frequency of the periodic update and periodic adaptation phases in the RA approach, in combination with flexible timeslots, the number of execution of these phases is also affected by each individual invocation of the re-scheduling process, making it more difficult to manage compared to a fixed number of timeslots.

We would like to note that the combination of dynamic, flexible, resilient and runtime adaptation approaches results in a highly dynamic environment, including 1) online entrance of reservation requests, 2) flexible timeslots of variable numbers and durations, 3) irregular updates of the schedule, 4) re-utilization of network resources at runtime, 5) making distinction between streaming requests' reservations and usages, 6) reacting to sudden changes such as network failures / repairs and streaming requests' resumes / playbacks.

6.5 Cloud-based application placement

In Chap. 5, we studied the problem of initial placement of application components, taking into account the placement constraints and network demands of interacting components, proposing optimal and near-optimal centralized solutions. Centralized resource allocation systems suffer from scalability issues, making them inappropriate for large-scale cloud environments. Considering the need for new scalable management paradigms, we also proposed a hierarchical solution that scales well to large size cloud systems. We showed that while a centralized solution always outperforms the hierarchical approaches in terms of number of mapped applications and servers used, its execution time dramatically grows when the number of servers increases. This makes the centralized algorithms inefficient for large scale cloud systems. The architecture of the hierarchical management plane depends on the number of supported servers by each administrative domain. The more servers needs to be supported, the higher the complexity of the approach and the more application placements are required. As such, this can be seen as a trade-off between quality and complexity of the cloud resource management systems. Our evaluation studies showed that in large-scale cloud systems the hierarchical approaches work efficiently compared to a centralized management system in terms of resource usage and quality of application placement. The percentage of nodes used and the percentage of mapped applications remain at least within 6.7% and 8% respectively, compared to that of the centralized management plane.

6.6 Future research

In the future, we expect work to continue on extensions to some of the key aspects discussed in this dissertation as follows.

6.6.1 Alternative use cases

This dissertation provides multiple advance bandwidth reservation solutions mainly for media transfers, resulting in a high flexibility when dealing with media. For example, media production companies will be able to work concurrently on the same video sequence. Additionally, in producing deadline sensitive video such as a news broadcast, producers will be able to work longer on the production of the video before sending it to the broadcaster. While this dissertation mainly focuses on customized solutions for the media production area, the results can also be applied to other use cases (buffering capability or other slight changes might be required), allowing new and improved applications. For example, in the medical field, our proposed solutions will enable applications such as the remote and pervasive participation of medical experts in surgeries.

6.6.2 Timeslot-based technique

In Chapter 4, we showed that flexible timeslots are by nature more compatible with requests in media related networks. Our simulations have proven that flexible time windows not only result in slightly higher request admittance ratio, but also execute faster, compared to the execution time of the best result achieved by the fixed size approach, i.e. with 5-min time intervals. However, we argue that a combination of flexible and fixed size approaches is a well-suited solution in the situations where a large number of requests may result in an excessive number of timeslots. The flexible approach is defeated due to uncontrolled complexity. This hybrid approach can be designed by introducing a minimum timeslot size, providing a trade-off between the number of timeslots and quality of the results. This way we gain the benefits of both approaches and the complexity of the flexible approach is manageable, especially in combination with the runtime adaptation approach.

6.6.3 The underlying technology

In order to implement AR scheduling, the underlying network has to support bandwidth reservations. Current research on the topic mostly focuses on optical networks in combination with wavelength division multiplexing. However, Software Defined Networking (SDN) has recently revolutionized dynamic network management aspects by decoupling the control and data planes and assigning the control to a programmable software unit. SDN-based techniques such as OpenFlow, provide high-level bandwidth reservation abstractions, hiding the details of the underlying physical mechanisms. This dissertation presents an AR-based media production platform that is generic in terms of the underlying reservation techniques (e.g., wavelength- or time-based multiplexing), which can be used in conjunction with

SDN. In contrast to most existing work, we believe that SDN can be employed as an efficient enabler for a generalized AR scheduling approach, rather than one specifically aimed at optical WDM technologies. However, this has remained as an open research problem regarding the way that SDN-based solutions can be used to automate the provisioning of advance reservations. The methods that allow the SDN controller to translate the produced schedule into flow rules and install these rules in the data plane has not been addressed in this dissertation. The functions which are already available by means of e.g. OpenFlow command and the controlling methods to start a transfer, hold a transfer, check residual bandwidth, etc. have to be investigated and defined.

6.6.4 Live migration of application components

In Chapter 5, the initial placement of deterministic application has been taken into account in which the structure of the application is always known beforehand. After the initial application placement, there is a necessity for monitoring application behavior and adjusting the placement decisions at runtime to avoid performance degradations. This capability is known as live migration i.e. the capability to relocate virtualized application components at runtime. The provisioning of live migration brings benefits such as improved performance, manageability and fault tolerance, while allowing workload movement with a short service downtime. However, service levels of applications are likely to be negatively affected during a live migration. Our presented work on the initial network- and constraint-aware application component placement can be extended to consider dynamic migration of virtualized components. A comprehensive investigation needs to be conducted to identify the requirements and potential issues. In addition, it is interesting to consider non-deterministic applications for future dynamic placement work as their workflow is determined at run time.

A

Single-path versus Multi-path Advance Reservation in Media Production Networks

In Chapter 2, we proposed advance bandwidth reservation approaches based on multi-path routing scheme. Restrictions on the underlying network can force the use of single-path routing mechanisms over multi-path approaches. In this appendix, we investigate the influence of using single-path routing compared to multi-path routing in deadline-aware advance reservation (AR) systems for media production networks. We have modified our previously designed optimal multi-path advance reservation model to incorporate the single-path mechanism and heuristic alternatives are presented and thoroughly evaluated. The experimental results reveal that the single-path optimal model can only provide satisfactory performance when the network is not in contention. With the heuristic approach, when adequate bandwidth is provided, the multi-path approach outperforms the single-path by up to 7.3%.

⋆ ⋆ ⋆

M. Barshan, H. Moens, B. Volckaert and F. De Turck

Published in the proceeding of the 6th international conference on Network Of the Future (NOF), pages 1–6, Sep. 2015.

A.1 Introduction

Exchanging a large number of media files is daily business in media production companies of all types, ranging from production houses to broadcasters. Traditional ways of transporting data, i.e. using dedicated and expensive point to point high speed links or even using physical transportation systems (vehicles and accompanying human resources) is costly and highly inefficient. As a standard media transport medium, wide area IP-based shared networks are being used more and more within these media-centric process flows.

The work in this appendix has been performed within the context of ICON MECaNO project, which aims to provide solutions for the transmission of large file-based media files and streaming sessions over IP-based network infrastructure, tailored to the quality and timing requirements of current and future media process flows.

In our previous work [1] we have shown the viability of using advance bandwidth reservation techniques in media production industry. We proposed an ILP model to solve the AR scheduling problem. Based on this model, two static and dynamic scheduling algorithms were presented: SARA which assumes all requests and their requirements are known in advance and DARA which supports rescheduling to incorporate new requests. In both algorithms we assumed that the flows can be split over multiple paths, making it possible to fully use the network. Using multi-path is however not always feasible when there is no full control over the underlying network and network devices. According to network flow theory, flows can be split into a number chunks, to be transferred over different paths through the network, this effect is undesired or even forbidden in some applications [2]. In the Internet, wireless networks, or overlay networks built on top of the Internet, traffic is mostly sent over a single path and generally splitting the flows is avoided due to the problem of packet reassembly at the receiver [3]. In addition, it is often not possible to use multi-path solutions due to limitations in the configuration of intermediate devices (e.g. the forwarding behavior on the routers cannot be modified to support multi-path routing).

While multi-path approaches are not always feasible, single path reservation cannot always meet the end-to-end QoS requirements in bandwidth-limited networks [4]. In this appendix, we analyse the impact and importance of supporting multi-path network flows in media production networks. To achieve this we have provided a modified version of the previously designed ILP-based models and algorithms to support unsplittable flows. This allows us to compare the performance of our advance network reservation system, using single-path versus multi-path bandwidth reservation approaches, to determine the importance of supporting multi-path flows.

The remainder of this appendix is structured as follows. In Section A.2, we discuss related work. Section A.3, provides extensions to our models to allow for single path reservations. The heuristic-based AR scheduling algorithms are described in Section A.4. Section A.5 provides simulation results, comparing the proposed algorithms. Finally, Section A.6 concludes the appendix.

A.2 Related work

The AR scheduling problem has been well studied in literature. While some have focused on rescheduling [5–7] and multi domain reservation [8], others have paid particular attention to real-world deployments [9–12], and WDM optical networks [13]. However, according to Charbonneau et al. [13], only two advance reservation algorithms support elastic reservation, and both consider fixed start time for the flows, while we consider flexible flow start times.

Moreover, the problem addressed in this work is related to the multi-commodity flow problem (MCFP). Comprehensive surveys on the approaches to solve multi-commodity flow problems (MCFP) and their variants are provided in [14, 15]. In [16], unsplittable flow and single path MCFPs are studied. Our approach extends this by dealing with the problem of flow variation over time and solves an MCFP as a sub-problem. Dynamic flows or flow variation over time are primarily introduced by Ford and Fulkerson [17, 18]. Our work differs by introducing support for variable reservations over time, elastic reservations, flexible start times and dependencies among different flows.

Another work [19] has compared single-path and multi-path routing approaches and concluded that multi-path routing provided limited gain compared to single-path routing. However, there are multiple differences which make their comparison inapplicable to the problem addressed in this appendix. First, their investigation is not about advance reservation, but about routing under certain traffic conditions. The authors focus on a comparison when all node-pairs generate traffic, while we focus on large file transfers within large networks where only limited numbers of nodes act as source and sink nodes, and finally no transfer deadlines are present in their approach.

This work is an extension of our previous work [1], in which the multi-path version of ILP-based models with two objective functions are proposed and thoroughly evaluated. The MaxA objective function maximizes the request admittance ratio while the ASAP objective in addition to maximizing the number of admitted requests, also tries to schedule the requests as soon as possible. As results showed that ASAP outperforms MaxA by up to 3.27%, in this appendix evaluation is only performed for the ASAP objective function. In addition, the heuristic approaches are proposed and their performance is compared to the optimal algorithms.

A.3 AR scheduling model

In [1] we proposed a formal model for the advance reservation scheduling of network bandwidth. In this section we show how this model can be extended to perform single-path reservations.

The model can be used to schedule collections of requests, that consist of multiple interdependent and deadline-constrained network transfers. Requests are grouped into scenarios, that represent a complex workflow. The workflows must be executed in their entirety, so when a scenario is admitted, all requests must be scheduled. The model only admits those scenarios for which sufficient bandwidth can be guaranteed during the reservation period. When a scenario is rejected, none of its requests are executed. The various requests within a scenario may depend on each other, meaning that one request can only start when other requests have finished. The model supports two types of network transfers: video streaming and large file transfers. The requests of all scenarios are stored in R. Consequently R consists of both types. To make distinction between the two types R_f which refers to file-based flows and R_s which refers to the streaming requests are defined. The network is represented as a graph with nodes N and edges E.

In this model the n^{th} request is denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$ comprising of the source of the request s^n, the destination node d^n, the time when the data for file-based request is ready to transfer t_s^n (or fixed start time for video streaming request), the deadline for the transmission of the data of file-based request t_e^n (or fixed end time for video streaming request), the duration of each request i^n and finally the bandwidth demand of the request b^n. In particular, r_f^n and r_s^n refer to file-based and video streaming requests respectively. Moreover the volume of the files are denoted by v^n and the time slot size by I. Table A.1 lists other notations which have been used to define this extension.

Table A.1: Symbols and notations used in the formal model.

Variable	**Description**
$\beta^{n,e,k}$	Dedicated Bandwidth between link *e*, request r^n and time slot *k*.
A^n	Binary variable, 1 iff request r^n is admitted, 0 otherwise.
t_s^{min}	Minimum start time of all reservations.
t_e^{max}	Maximum end time of all reservations.
B^e	Bandwidth capacity of link *e*.
E_v^{out}	This collection contains all edges starting from node v (egress).
E_v^{in}	This collection contains all edges ending in node v (ingress).

This model is partially similar to the multi-path reservation model. All the decision variables, objective functions and constraints of former model are valid and applicable to this model. Therefore, only additions are discussed.

A.3.0.1 Additional decision variable

We need to make sure that only a single path is reserved for each request. To achieve this, in addition to the 6 decision variables, one more binary decision variable, $P^{n,e,k}$, is defined which indicates whether there is any reservation for request n in time slot k over link e.

$$P^{n,e,k} \in [0,1] \qquad \forall r^n \in R, \forall e \in E, k \in [t_s^{min}, t_e^{max}]$$

A.3.0.2 Additional constraints

In addition to the constraints which ensure that capacity limitation, network flow concepts and dependencies among requests are respected, 5 extra constraints concerning single-path flow conservation of P values and linking P values to β values are defined. Constraints 1, 2 and 3 ensure that only a single-path is chosen to be reserved for a request during each time interval.

$$\sum_{e \in E_v^{out}} P^{n,e,k} = \sum_{e \in E_v^{in}} P^{n,e,k} \tag{A.1}$$
$$\forall r^n \in R, \forall k \in [t_s^n, t_e^n], \{\forall v \in N | v \notin \{s^n, d^n\}\}$$

$$\sum_{e \in E_{s^n}^{out}} P^{n,e,k} = A^n \qquad \forall r^n \in R, \forall k \in [t_s^n, t_e^n] \tag{A.2}$$

$$\sum_{e \in E_{d^n}^{in}} P^{n,e,k} = A^n \qquad \forall r^n \in R, \forall k \in [t_s^n, t_e^n] \tag{A.3}$$

Constraint 4 ensures if the type of the request is video streaming, the dedicated and requested bandwidth must be equal. Also Constraint 5 is defined for linking P values to β flows.

$$\beta^{n,e,k} = b^n \times P^{n,e,k} \;\; \forall e \in E, \forall r_s^n \in R_s, \forall k \in [t_s^n, t_e^n] \tag{A.4}$$

$$\beta^{n,e,k} \leq B^e \times P^{n,e,k} \;\; \forall e \in E, \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \tag{A.5}$$

A.4 AR scheduling algorithm

The Sequential Priority Based (SPB) advance reservation algorithm is a heuristic solution which is proposed due to the high computational overhead and scalability issue of the ILP approach. Individual components of the SPB algorithm are shown in Figure A.1. As can be seen, new scenarios enter the reservation system through an API. In the next step any transformation can be applied. For example in the

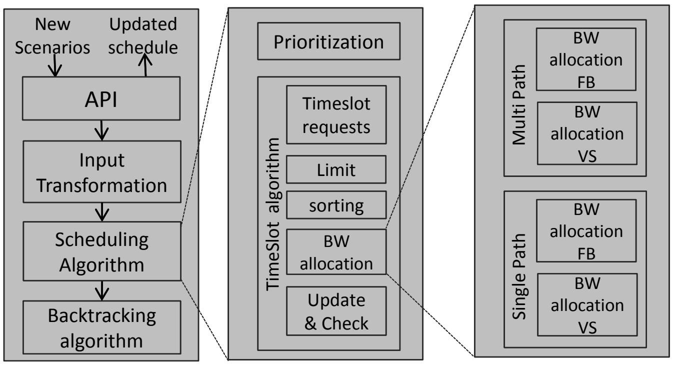


Figure A.1: Components of the Sequential Priority Based Advance Reservation Algorithm (SPB).

dynamic approach before the scheduling algorithm invocation, the previously admitted scenarios' demand needs to be updated. Then the scheduling algorithm is sequentially invoked for each scenario. If this process is successfully terminated the new scenario is admitted, and the schedule is updated. Otherwise, the previous schedule and network state remain untouched and the scenario is rejected. The scheduling algorithm consists of two components: The prioritization and the TimeSlot algorithms. The prioritization algorithm assigns priorities to the scenario's requests based on the estimated hard deadline and the volume. Since the deadline may not be specified for all requests, the latest possible deadline for those with no specific deadline should be estimated. Then all the scenario's requests are given to the TimeSlot algorithm. This algorithm consists of 5 sub-algorithms for each time interval.

TimeSlotRequests: First, the algorithm determines which requests can be served in the current time slot. For independent requests the algorithm looks at the start time. If the current interval is greater than or equal to the request start time, these requests are eligible to be added to the list of current requests. For requests with start time dependencies, the algorithm checks whether the requests on which this request depends are finished or not.

Limit: The limit for video streams is their required demand, which is fixed and non-variable and for file-based requests is their residual demand.

Sorting: In this step requests are sorted based on their previously assigned priorities.

BWallocation: We have defined four variations of bandwidth allocation algorithm for video streams and video files using single-path and multi-path routing. This algorithm first assigns cost to the network links using the Cost allocation component which tries to find the most desired paths and give them the highest cost. Then according to the approach and type of the request multi-path BWallocationFB, multi-path BWallocationVS, single-path BWallocationFB or single-path BWallocationVS algorithms is invoked.

Single-path BWallocation algorithms: The single-path *BWallocationFB* algorithm is in charge of the FB requests. As we aim at transferring the video files in earlier timeslots, the single-path BWallocationFB algorithm first tries to find the most available bandwidth between the source and destination of the request. If multiple candidates with maximum bandwidth are available, this algorithm looks for the least-cost single-path that could carry this maximum flow using the modified version of Dijkstra to consider the cost of the paths, which are previously assigned to the network links, and ignoring the paths with lower capacity than the maximum flow.

The single-path *BWallocationVS* algorithm deals with video stream requests. For video streams, this algorithm first removes all the network links with capacities lower than the request demands, and then the least-cost path is determined. If no path is found, rescheduling is unsuccessful and the new scenario is rejected.

Multi-path BWallocation algorithms: The multi-path *BWallocationFB* algorithm is based on maxflow and least-cost path algorithms. If the maxflow, which is calculated based on the Edmonds-Karp algorithm, is lower than the request limit, all the maxflow paths are reserved for this request. Otherwise, the algorithm forms a graph out of the maxflow paths and the k-shortest path is the second alternative. Finding the least-cost path is repeated until the total bandwidth offered by the paths is sufficient for the request.

The multi-path *BWallocationVS* algorithm iteratively looks for the least cost path on the whole graph and sums up the minimum available bandwidth of the paths. These steps are repeated over the residual graph while the total bandwidth provided by the paths fulfils the request demand.

Update and check feasibility: based on the provided result of the BWallocation component, and by calculating the residual demands, the requests requirements are updated and the feasibility of the results is checked. If the hard deadline of a request is reached, but part of the request has not been transferred yet and the residual demand is not zero, the hard deadline has not been met and rescheduling is infeasible.

A.5 Experimental results

This section evaluates the single-path and multi-path approaches for ILP-based and SPB scheduling algorithms. The influence of bandwidth availability, network load, and the time granularity are assessed.

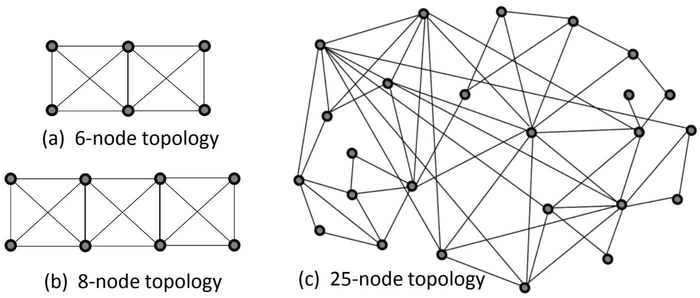


Figure A.2: Media production network topologies used in the evaluation.

A.5.1 Evaluation Setup

We found that the 12-node topology which is used in our previous evaluation yields identical results for the single-path and multi-path approaches both for ILP-based and SPB algorithms. In this evaluation we have used 3 other topologies for media production networks which are depicted in Figure A.2.

Throughout this section, *XX*[*YY*,*ZZ%*] denotes that approach *XX* (i.e. ILP or SPB), routing mechanism *YY* which can either be Single-Path (SP) or Multi-Path (MP) is used and *ZZ%* of the use case instances are known at the start of the simulation. Each simulation run covers a 24 hour period. All results are averaged over 50 runs with different randomized inputs, error bars denote the standard error.

A.5.2 ILP evaluation of single-path versus multi-path

In this evaluation the number of use case instances equals 6, resulting in of 62 requests. The 8-node topology and a fixed time interval granularity of 1 hour is used.

Figure A.3 compares the percentage of admitted requests of ILP-based single-path and ILP-based multi-path approaches. From the figure, we can conclude that the multi-path approach significantly outperforms the single-path approach when network contention happens (bandwidth capacity lower than 200 Mbps). In this situation there is insufficient capacity for the scheduler to reserve a single-path for some flows by their deadline. However, this has no impact on the multi-path approach as the flows can be split and sent over multiple paths. The result shows that there is up to 24.3% differences in request admittance ratio.

A.5.3 Comparison of ILP-based model with SPB approach

For this evaluation the impact of network capacity is assessed. The 6-node network topology is used and the number of scenarios is 8 (85 requests in total). Figure A.4

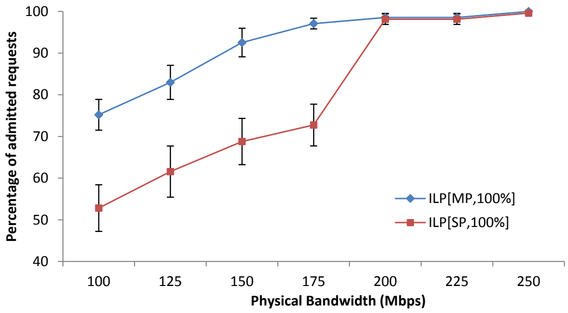


Figure A.3: Comparing single-path versus multi-path in ILP-based approach.

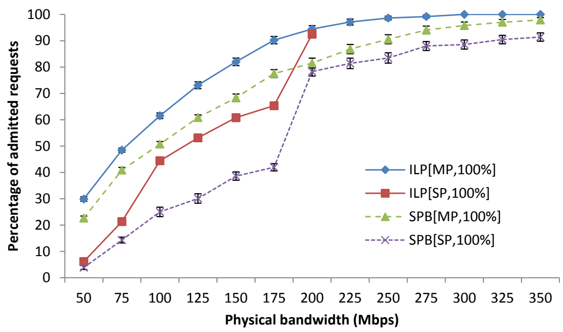


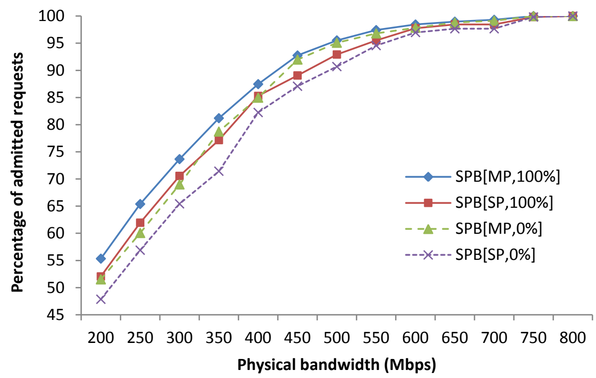
Figure A.4: Comparison of optimal ILP with SPB approach.

compares the ILP-based algorithms to the SPB ones for both multi-path and single-path approaches. This figure shows that the result of the SPB algorithm is within 13.6% of the ILP approach when multi-path routing mechanisms are used. For the single-path method, a similar trend can be observed.

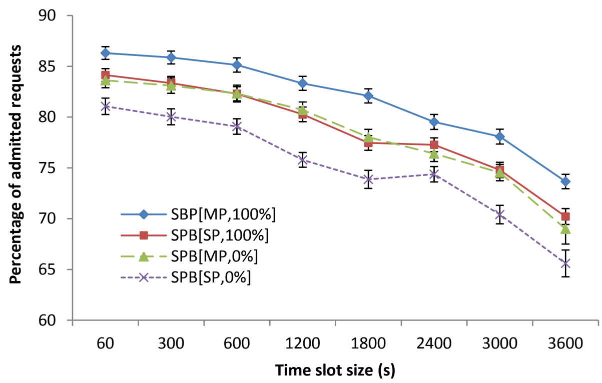
A.5.4 Evaluation of single-path and multi-path in SPB approach

In this section we evaluate the impact of network capacity, time slot granularity and network load on the performance of the SPB algorithms. Two topologies of the 8-node and 25-node serve as media production infrastructures in Figure A.5 and Figure A.6 respectively.

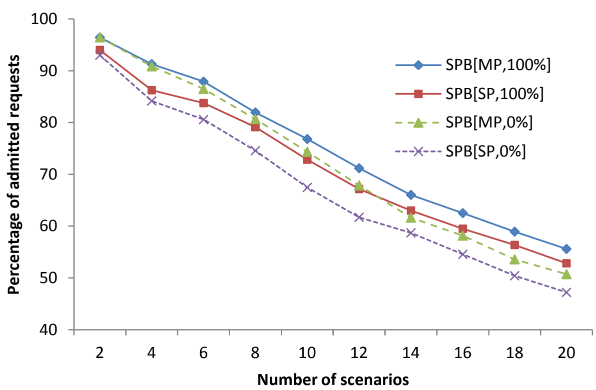
In part (a) of both figures, the media network infrastructures have been configured for different available bandwidths to investigate the impact of network capacities on the performance of our algorithms. In both plots a time slot size of 1 hour is used and the number of scenarios is 20 (209 requests) and 50 (519 requests) for the 8-node and 25-node topologies respectively. The result shows that the perfor-



(a) Time slot size=1 hour, Number of scenarios=20

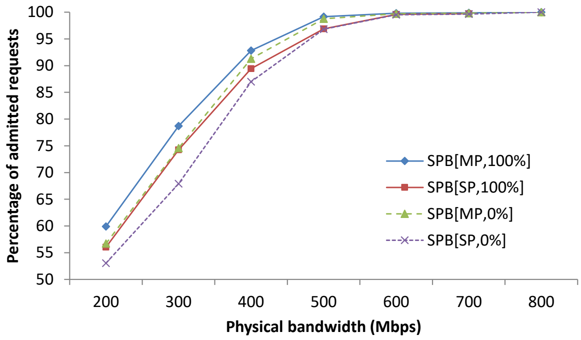


(b) Bandwidth=200 Mbps, Number of scenarios=20

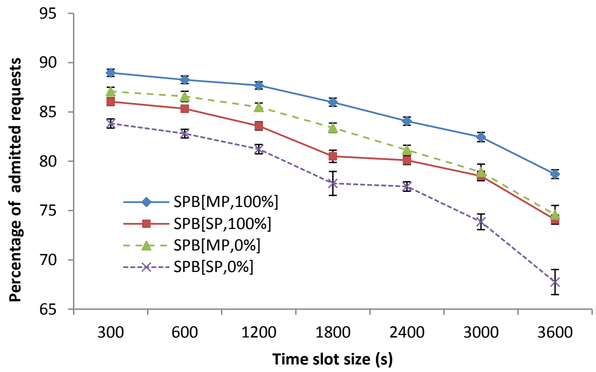


(c) Time slot size=1 hour, Bandwidth=300 Mbps

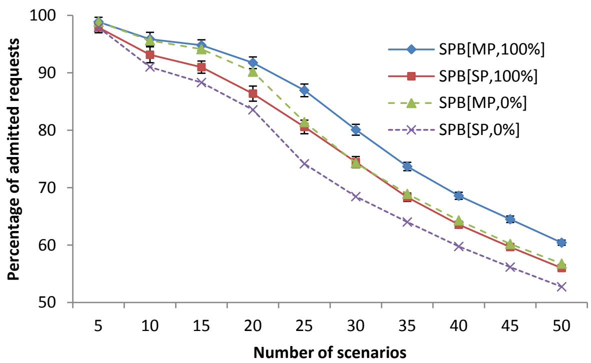
Figure A.5: Impact of bandwidth capacity, time slot granularity and network load on admission rate in 8-node topology



(a) Time slot size=1 hour, Number of scenarios=50



(b) Bandwidth=200 Mbps, Number of scenarios= 50



(c) Time slot size=1 hour, Bandwidth=300 Mbps

Figure A.6: Impact of bandwidth capacity, time slot granularity and network load on admission rate in a 25-node topology.

mance of single-path approach is within 7.3% and 6.7% of multi-path approach for 8-node and 25-node topologies respectively.

In part (b) the impact of time slot granularity is studied. While the number of use case instances is considered 20 and 50 for smaller and larger topologies respectively, a link capacity of 200 Mbps is used in both evaluations. The results show that the experiment with shortest time slot leads to the best performance and the multi-path approach outperforms the single-path approach up to 4.9% and 6.8% in Figure A.5 and Figure A.6 respectively.

Finally, part (c) evaluates the impact of network load when the number of scenarios increases up to 20 and 50 scenarios in Figure A.5 and Figure A.6 respectively. In both figures the timeslot size of 1 hour and network capacity of 300 Mbps is used. The results show that for both smaller and larger topologies the multi-path approach outperforms the single-path approach by up to 6.9% and 7.3% respectively.

A.6 Conclusion

In this appendix, the impact of using single-path routing mechanisms in advance reservation system for media production networks, is compared to a multi-path approach. We extended the optimal advance bandwidth reservation model to incorporate single-path routing and provided equivalent heuristic solutions. The impact of physical capacity, time interval granularity and network load were evaluated. Based on our simulation results, a multi-path approach is beneficial, improving the request admittance rate by up to 24.3% compared to when by using single-path solutions. However, if multi-path routing is not a viable solution, our evaluation showed that increasing the network capacity will significantly improve the performance of a single-path advance reservation system. The evaluation of our heuristics indicated that the single-path approach can achieve performance levels which remain within 7.3% of multipath routing mechanism.

Acknowledgment

The research leading to these results was performed within the context of ICON MECaNO. It is a project co-funded by iMinds, a digital research institute founded by the Flemish Government. Project partners are SDNsquare, Limecraft, Video-House, Alcatel-Lucent, and VRT, with project support from IWT under grant agreement no. 130646.

References

[1] M. Barshan, H. Moens, J. Famaey, and F. De Turck. *Algorithms for Advance Bandwidth Reservation in Media Production Networks*.

[2] M. Martens and M. Skutella. *Flows on few paths: Algorithms and lower bounds*. Networks, 48(2):68–76, 2006.

[3] B. Awerbuch, Y. Azar, and A. Epstein. *The price of routing unsplittable flow*. In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pages 57–66. ACM, 2005.

[4] J. Chen, S.-H. Chan, and V. O. Li. *Multipath routing for video delivery over bandwidth-limited networks*. Selected Areas in Communications, IEEE Journal on, 22(10):1920–1932, 2004.

[5] K. Rajah, S. Ranka, and Y. Xia. *Advance Reservations and Scheduling for Bulk Transfers in Research Networks*. IEEE Trans. Parallel Distrib. Syst., 20(11):1682–1697, November 2009. Available from: http://dx.doi.org/10.1109/TPDS.2008.250, doi:10.1109/TPDS.2008.250.

[6] C. Xie, H. Alazemi, and N. Ghani. *Rerouting in advance reservation networks*. Computer Communications, 35(12):1411–1421, 2012.

[7] L. Zuo, M. M. Zhu, and C. Q. Wu. *Fast and Efficient Bandwidth Reservation Algorithms for Dynamic Network Provisioning*. Journal of Network and Systems Management, 2013.

[8] H. Alazemi, F. Xu, C. Xie, and N. Ghani. *Advance reservation in distributed multi-domain networks*. IEEE Systems Journal, 2013.

[9] C. Guok, E. N. Engineer, and D. Robertson. *ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS)*. Internet2 Joint, 2006.

[10] B. Gibbard, D. Katramatos, and D. Yu. *TeraPaths: end-to-end network path QoS configuration using cross-domain reservation negotiation*. In Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on, pages 1–9. IEEE, 2006.

[11] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee. *StorNet: Integrated Dynamic Storage and Network Resource Provisioning and Management for Automated Data Transfers*. In Journal of Physics: Conference Series, volume 331, page 012002. IOP Publishing, 2011.

[12] S. Sharma, D. Katramatos, D. Yu, and L. Shi. *Design and Implementation of an Intelligent End-to-end Network QoS System*. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, pages 68:1–68:11, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press. Available from: http://dl.acm.org/citation.cfm?id=2388996.2389089.

[13] N. Charbonneau and V. M. Vokkarane. *A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks*. Communications Surveys & Tutorials, IEEE, 14(4):1037–1064, 2012.

[14] A. Ouorou, P. Mahey, and J.-P. Vial. *A survey of algorithms for convex multicommodity flow problems*. Management science, 46(1):126–147, 2000.

[15] J. L. Kennington. *A survey of linear cost multicommodity network flows*. Operations Research, 26(2):209–236, 1978.

[16] H. Masri, S. Krichen, and A. Guitouni. *A multi-start variable neighborhood search for solving the single path multicommodity flow problem*. Applied Mathematics and Computation, 251:132–142, 2015.

[17] L. R. Ford Jr and D. R. Fulkerson. *Constructing maximal dynamic flows from static flows*. Operations research, 6(3):419–433, 1958.

[18] L. Ford and D. R. Fulkerson. *Flows in networks*, volume 1962. Princeton Princeton University Press, 1962.

[19] X. Liu, S. Mohanraj, M. Pióro, and D. Medhi. *Multipath Routing From a Traffic Engineering Perspective: How Beneficial is It?* In Network Protocols (ICNP), 2014 IEEE 22nd International Conference on, pages 143–154. IEEE, 2014.