

Robustness to Failures in Two-Layer Communication Networks

THÈSE N° 4295 (2009)

PRÉSENTÉE LE 27 FÉVRIER 2009

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 3
SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Maciej Kurant

acceptée sur proposition du jury:

Prof. A. Martinoli, président du jury
Prof. P. Thiran, directeur de thèse
Prof. P. Frossard, rapporteur
Prof. A. Jajszczyk, rapporteur
Prof. A. Markopoulou, rapporteur



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2009

To my wife Rybcia,
who pushed me towards this adventure.

“Layers! Onions have layers. Ogres have layers. Onions have layers...you get it. We both have layers.”

Shrek to Donkey, in ‘Shrek’, 2001

“If it turns out there is a simple ultimate law which explains everything, so be it – that would be very nice to discover. If it turns out it’s like an onion with millions of *layers*... then that’s the way it is.”

Richard Feynman, 1918-1988

Abstract (English)

A close look at many existing systems reveals their two- or multi-layer nature, where a number of coexisting networks interact and depend on each other. For instance, in the Internet, any application-level graph (such as a peer-to-peer network) is mapped on the underlying IP network that, in turn, is mapped on a mesh of optical fibers. This layered view sheds new light on the tolerance to errors and attacks of many complex systems. What is observed at a single layer does not necessarily reflect well the state of the entire system. On the contrary, a tiny, seemingly harmless disruption of one layer, may destroy a substantial or essential part of another layer, thus making the whole system useless in practice.

In this thesis we consider such two-layer systems. We model them by two graphs at two different layers, where the upper-layer (or *logical*) graph is mapped onto the lower-layer (*physical*) graph. Our *main goals* are the following. First, we *study* the robustness to failures of existing large-scale two-layer systems. This brings us some valuable insights into the problem, e.g., by identifying common weak points in such systems. Fortunately, these two-layer problems can often be effectively alleviated by a careful system design. Therefore, our second major goal is to *propose new designs* that increase the robustness of two-layer systems.

This thesis is organized in three main parts, where we focus on different examples and aspects of the two-layer system.

In the first part, we turn our attention to the existing large-scale two-layer systems, such as peer-to-peer networks, railway networks and the human brain. Our main goal is to study the vulnerability of these systems to random errors and targeted attacks. Our simulations show that (i) two-layer systems are much more vulnerable to errors and attacks than they appear from a single layer perspective, and (ii) attacks are much more harmful than errors, especially when the logical topology is heterogeneous. These results hold across all studied systems.

A natural next step consists in improving the failure robustness of two-layer systems. In particular, in the second part of this thesis, we consider the

IP/WDM optical networks, where an IP backbone network is mapped on a mesh of optical fibers. The problem lies in designing a *survivable* mapping, such that no single physical failure disconnects the logical topology. This is an NP-complete problem. We introduce a new concept of *piecewise survivability*, which makes the problem much easier in practice. This leads us to an efficient and scalable algorithm called SMART, which finds a survivable mapping much faster (often by orders of magnitude) than the other approaches proposed to date. Moreover, the formal analysis of SMART allows us to prove that a given survivable mapping does or does not exist. Finally, this approach helps us to find vulnerable areas in the system, and to effectively reinforce them, e.g., by adding new links.

In the third part of this thesis, we shift our attention one layer higher, to the application-over-IP setting. In particular, we consider the design of Application-Level Multicast (ALM) for interactive applications, where a single source sends a delay-constrained data stream to a number of destinations. Interactive ALM should (i) respect stringent delay requirements, and (ii) proactively protect the system against overlay node failures and against (iii) the packet losses at the IP layer. We propose a two-layer-aware approach to this problem. First, we prove that the average packet loss rate observed at the destinations can be effectively approximated by a purely *topological* metric that, in turn, drops with the amount of *IP-level and overlay-level path diversity* available in the system. Therefore, we propose a framework that accommodates and generalizes various techniques to increase the path diversity in the system. Within this framework we optimize the structure of ALM. As a result, we reduce the effective loss rate of real Internet topologies by typically 30%-70%, compared to the state of the art.

Finally, in addition to the three main parts of the thesis, we also present a set of results inspired by the study of ALM systems, but not directly related to the ‘two-layer’ paradigm (and thus moved to the Appendix). In particular, we consider a transmission of a delay-sensitive data stream from a single source to a single destination, where the data packets are protected by a Forward Error Correction (FEC) code and sent over multiple paths. We show that the performance of such a scheme can often be further improved. Our key observation is that the propagation times on the available paths often significantly differ, typically by 10-100ms. We propose to exploit these differences by appropriate packet scheduling, which results in a two- to five-fold improvement (reduction) in the effective loss rate.

Keywords: Two-layer networks, robustness to errors and attacks, complex networks, IP/WDM, overlay networks, multipath transmission.

Abstract (Rumantsch)

Cur chi's guarda ils systems chi existan da maniera plü precisa, lura as s'accorscha ch'els han plüssas stresas, e cha diversas raits interagischan e dependan l'üna da l'otra. Per exaimpel ill'internet, üna rait da la stresa d'applicaziun (per exaimpel üna rait da "Peer-to-Peer") po esser transponada illa rait dad IP chi as chatta suotvart, e quella rait es danouvamaing transponada in üna rait da fibras opticas. Quista perspectiva illümina la resistenza cunter sbagls ed attachs in systems cumplichads dad üna nouva maniera. Quai chi's observa in üna singula stresa nun sto esser ün bun purtret da tuot il sistem. Anzi, ün'interruptziun pitschnischma in üna stresa po devastar üna gronda part dad ün'otra stresa, ed uschenas render tuot il sistem indouvrabel.

In quista dissertaziun, nus contemplain systems avec duos stresas. Nus tils modellain cun duos grafs sün duos nivels differents. Il graf chi's chatta sü'l nivel sura (il graf logic) es transponà i'l graph chi's chatta sü'l nivel suot (il graf fisic). Nos böts sun ils seguaints. Il prüm, nus stübgian la robustezza cunter sbagls da gronds systems cun duos stresas chi existan fingià. Quist maina üna preziosa invista aint in quist problem. Per furtüna, id es suvent pussibel da levgiar quistas manatschas tras ün design prudent dal sistem. Nos seguond böt es da proponer novs designs chi augmentan la robustezza da systems cun duos stresas.

Quista dissertaziun es organisada in trais parts prinzipalas. Minchüna focussa sün differents exaimpels ed aspects dal sistem cun duos stresas.

Illa prüma part, nus guardain gronds systems cun duos stresas chi existan fingià, per exaimpel raits da "Peer-to-Peer", raits da tren obain il tschervé dals umans. Nos böt prinzipal es da stübgiar la sensibilità da quists systems cunter sbagls casuals ed attachs intenziunads. Nossas simulaziuns muossan cha (i) systems cun duos stresas sun plü sensibels cunter sbagls ed attachs co chi's pudess crajer dad üna perspectiva chi guarda be ün nivel, e (ii) attachs fan ün donn plü grond co sbagls, spezialmaing cur cha la topologia es eterogena. Quists resultats sun vairs per tuot ils systems cha nus vain stübgiaids.

Ûn prossem pass chi'd es natüral es da megldrar la robustezza cunter sbagls in systems cun duos stresas. Particularamaing illa seguonda part da quista dissertaziun nus guardain raits opticas dad IP/WDM. Il problem es da skizzar üna transponaziun *survivainta*, chi riva da surviver sbagls fisics. Quist problem es NP-cumplet. Ma nus preschentaun ün nouv concept da *capazität da surviver toc per toc*, ed ans chattain cun ün problem plü simpel. Nus preschentaun ün algoritmus effizient nomnà SMART. Quist algoritmus chatta üna transponaziun *survivainta* in damain temp co otras soluziuns existentas. Cull'analisa formella da SMART pudain nus cumprovar ch'üna transponaziun *survivainta* exista o na. Finalmaing, cun quista tecnica nus pudain chattar regions sensibilas dad ün sistem e tillas render plü robustas.

Finalmaing, illa terza part da quista dissertaziun, nus guardain amo üna stressa plü ot, nempe la situaziun dad applicaziuns sur IP. Nus guardain particularamaing la construcziun dad “Application-Layer Multicast” (ALM) per applicaziuns interactivas. Qua, üna funtana trametta ün fluss da datas a plüssas destinaziuns. Il ALM interactiv stess (i) respectar restricziuns da retard, e proteger il sistem cunter (ii) sbagls dad apparats e (iii) perditas illa stressa dad IP. Nus proponain üna soluziun chi guarda tuottas duos stresas. Nus muossain ch'is po approximar la frequenza da perdita da pakets per las destinaziuns cun üna metrica topologica chi dependa da la spespezza da sendas aint il sistem. Pervi da quai, nus proponain üna soluziun chi douvra diversas tecnicas per agrondir la diversità da sendas aint il sistem. Uschenas pudain nus megldrar la structura dad ALM. Cun quai, nus eschan capabels da reducir la frequenza da sbagls in topologias dad internet da 30 a 70 pertschient.

Nus agiunschain a quistas trais parts insaquants results inspirats dal studi da systems dad ALM chi nu sun directamaing in relaziun cun architectures cun duos stresas. Quists reslutats as chattan i'l Appendix. Nus guardain la transmissiun da datas urgentas dad üna funtana ad üna singula destinaziun, e nus protegain ils pakets cun “Forward Error Correction” (FEC) per tils trametter sün differentas sendas. Nus muossain cha'ls temps da transmissiun pon esser fich differentes per differentas sendas, normalmaing da 10 a 100 ms. Nus proponain dad utilizar quistas differenzas cun üna planisaziun adattada chi maina ad üna meglieraziun da la frequenza da sbagls da tschinch jadas.

Pled-clavs: Raits cun duos stresas, robustezza cunter sbagls ed attachs, raits complexas, IP/WDM, raits dad overlay, transmissiun sur plüssas sendas.

Acknowledgements

First of all, great thanks to Patrick Thiran, my supervisor, and a genuine scientist. He taught me how to do serious and rigorous research. He was also very tolerant and forgiving, when necessary.

I would also like to thank all the other members of my lab. For moral support, fruitful discussions, feedback on my work, a couple of parties, and common lunches and coffees.

Many thanks to all my friends, including the doctoral school folks, basement republic guys, the autobusiukas team, my office mate, the Polish community, all my friends back in Poland and scattered around the world, the PRO soccer team, Graduate Student Association (GSA), and all my moto/mountain/windsurfing/bridge buddies. Allow me not to give the concrete names and face the sensitive ‘whom-to-enlist’ dilemma. With one exception, Etienne Perron, who actually wrote a part of this thesis by translating the abstract to one of the four national languages of Switzerland.

Last, but not least, I would like to thank my family, especially my parents (“When are you finally going to graduate?!”), my grandmother (“It’s all nice, but when will you find a *real* job?”), my younger sister (“Maybe I should study a bit longer too?”) and my lovely wife (“Go for it!”).

Contents

1	Introduction	19
1.1	Motivations and Goals	19
1.2	Dissertation Outline and Contributions	21
1.3	Related Work	25
2	General Two-Layer Systems	27
2.1	Two-Layer Structure	27
2.1.1	Physical Graph $G^\phi = (V^\phi, E^\phi)$	27
2.1.2	Logical Graph $G^\lambda = (V^\lambda, E^\lambda)$	27
2.1.3	Mapping M	28
2.2	Failures	29
2.2.1	Edge Failures vs. Node Failures	29
2.2.2	Logical Failures vs. Physical Failures (and failure propagation)	29
2.2.3	Single vs. Multiple Failures	30
2.2.4	Errors vs. Attacks	30
2.3	Failure Protection Mechanisms	31
2.3.1	Proactive vs Reactive	31
2.3.2	Physical Layer vs. Logical Layer	31
2.4	The Notion of Vulnerability	31
2.5	Two-Layer Systems Addressed in This Thesis	33
3	Vulnerability of Existing Large-Scale Two-Layer Systems	35
3.1	Introduction	35
3.1.1	Errors vs Attacks	35
3.1.2	Error and Attack Tolerance of Single-Layer Systems	36
3.1.3	Many complex systems are layered	36
3.1.4	Our Objective and Achievements	36
3.1.5	Organization of this Chapter	37
3.2	Data Sets and Models	37
3.2.1	Railway	38

3.2.2	Gnutella	39
3.2.3	Brain	40
3.2.4	‘ER on ER’ and ‘BA on ER’	41
3.3	Failures, Protection Techniques, and Vulnerability	42
3.3.1	Failures	42
3.3.2	Protection Mechanisms	43
3.3.3	Vulnerability Metric	43
3.4	Edge Load Distribution	44
3.5	Simulation Results	45
3.6	Conclusion: Single-Layer vs. Two-Layer View	47
APPENDIX		49
3.A	Trainspotting: Extraction of Transportation Network Topologies from Timetables	49
3.B	From Diffusion MRI to a Brain Network	54
4	Survivability in IP/WDM Networks	57
4.1	Introduction	58
4.2	IP/WDM as a Two-Layer System	59
4.2.1	Physical Layer, Logical Layer and the Mapping in IP/WDM Networks	59
4.2.2	Failures in IP/WDM Networks	60
4.2.3	Failure Protection Mechanisms in IP/WDM Networks	61
4.3	Our Objectives and Achievements	62
4.3.1	We Address the IP Restoration Problem	62
4.3.2	Vulnerability \mathcal{W} and Survivability	62
4.3.3	Related Work	62
4.3.4	Our Contributions	64
4.3.5	We use the general two-layer model terminology	65
4.4	Additional Notation	65
4.4.1	Combining two mappings	66
4.4.2	Contraction and Origin	67
4.4.3	Survivability and Piecewise Survivability	68
4.5	Fundamental properties of survivable and	70
4.5.1	The Expansion of Survivability	70
4.5.2	Invariance of Survivability Under Contraction	71
4.5.3	The Existence of a Survivable Mapping	71
4.6	The SMART algorithm	72
4.6.1	The pseudo-code of SMART	72
4.6.2	SMART illustration	73
4.6.3	The Correctness of the SMART Algorithm	74
4.6.4	The Order of a Sequence of Subgraphs	74

4.7	Implementation - SMART-H	75
4.7.1	Which theoretical results hold for SMART-H?	75
4.7.2	<i>DisjointMap</i> - a heuristic for Step 2 of SMART-H	76
4.8	SMART-H Applications	78
4.8.1	Application1: Formal Verification of the Existence of a Survivable Mapping (ExSearch and SepPath)	79
4.8.2	Application 2: A Tool Tracing and Repairing the Vul- nerable Areas of the Network	80
4.8.3	Application 3: A Fast Heuristic	80
4.9	Simulation Results	80
4.9.1	Physical and Logical Topologies	80
4.9.2	ExSearch and SepPath Efficiency, and ‘Unknown Area’	81
4.9.3	Results for Application 1	84
4.9.4	Results for Application 2	84
4.9.5	Results for Application 3	85
4.10	Extension 1: Span failures	89
4.10.1	Changes: SMART-Span	89
4.10.2	Results	90
4.11	Extension 2: Node failures	91
4.11.1	Changes: SMART-Node	91
4.11.2	Results	91
4.12	Extension 3: Double-link failures	91
4.12.1	Changes: SMART-2Link	92
4.12.2	Results	92
4.13	Extension 4: Capacity constraints	93
4.13.1	Which theoretical results hold?	94
4.13.2	SMART-C	95
4.13.3	Results	95
4.14	Conclusion	96
	APPENDIX	97
4.A	Proofs	97
4.A.1	Proof of Theorem 1	98
4.A.2	Proof of Theorem 2	100
4.A.3	Proof of Theorem 3	101
4.A.4	Proof of Theorem 4	101
4.A.5	Proof of Corollary 1	102
4.A.6	Proof of Theorem 5	103

5	Maximal Path Diversity in Overlay/IP Networks	105
5.1	Introduction	105
5.2	Overlay as a Two-Layer System	106
5.2.1	The Physical Layer, Logical Layer and the Mapping in Overlay/IP Networks	107
5.2.2	Failures in Overlay/IP Networks	107
5.2.3	Failure Protection Mechanisms in Overlay/IP Networks	108
5.3	Application-Level Multicast (ALM)	108
5.3.1	Regular ALM	109
5.3.2	Interactive ALM	109
5.3.3	Design Goals in Interactive ALM	110
5.4	Our Objectives and Achievements	111
5.4.1	We consider the Interactive ALM	111
5.4.2	Vulnerability \mathcal{W}	111
5.4.3	Related work	111
5.4.4	Our Contributions	112
5.5	Additional notation and problem formulation	113
5.5.1	Overlay layer and capacity constraints	113
5.5.2	IP layer	114
5.5.3	Packet losses	114
5.5.4	Maximal allowed time $t_{max}(v)$ and average loss rate \bar{r} .	115
5.5.5	General problem formulation: Problem P1	115
5.6	Topology-based formulation: Problem P2	115
5.6.1	Critical components $C^\lambda(v)$ and $C^\phi(v)$	116
5.6.2	Vulnerability $\mathcal{W}, \mathcal{W}^\lambda$ and \mathcal{W}^ϕ	116
5.6.3	Problem P2	117
5.6.4	Equivalence of P1 and P2	117
5.7	Protection techniques	118
5.7.1	SingleTree (Fig. 5.4a)	118
5.7.2	SingleGraph (Fig. 5.4b)	120
5.7.3	MultiTree (Fig. 5.4c)	120
5.7.4	MultiGraph (Fig. 5.4d)	120
5.7.5	A common framework \mathbf{G}^λ	121
5.8	Objective functions	122
5.8.1	RAND (reference point)	122
5.8.2	1-LAYER (state of the art)	122
5.8.3	2-LAYER (our proposal)	123
5.9	Constructing a good topology \mathbf{G}^λ	123
5.9.1	Lower-bounds on \mathcal{W}^λ and \mathcal{W}^ϕ	124
5.10	Simulation results	125
5.10.1	Data sets	125

5.10.2	Setting t_{max}	125
5.10.3	Vulnerability $\mathcal{W}(\eta)$ and topology \mathbf{G}^λ as a function of η	125
5.10.4	Detailed results for $0 < \eta < 1$	126
5.11	Conclusion	129
	APPENDIX	130
5.A	Proofs	130
5.A.1	Proof of Theorem 6 [Equivalence of P1 and P2]	130
5.A.2	Proof of Observation 1	132
5.B	Lower bounds	133
5.B.1	Simple lower-bound on \mathcal{W}^ϕ	133
5.B.2	<i>CompleteGraph</i> lower-bound on \mathcal{W}^ϕ	134
5.B.3	<i>SingleGraph</i> (and <i>SingleTree</i>) lower-bound on \mathcal{W}^ϕ	134
5.B.4	<i>MultiTree</i> lower-bound on \mathcal{W}^ϕ	135
5.B.5	<i>SingleGraph</i> (and <i>SingleTree</i>) lower-bound on \mathcal{W}^λ	136
5.C	Speeding-up the heuristic	137
5.C.1	Computation of critical elements for $G_i^\lambda \in \mathbf{G}^\lambda$	137
5.C.2	Useful theoretical result	139
6	Conclusion	141
	APPENDIX	144
A	Exploiting Path Propagation Time Differences in Multipath Transmission with FEC	147
A.1	Introduction	147
A.1.1	Propagation times on direct and indirect paths may differ significantly	148
A.1.2	The differences in propagation times can be exploited by a multipath FEC system	151
A.1.3	Organization of this chapter	152
A.2	Model and problem statement	152
A.2.1	Path losses	152
A.2.2	Multipath FEC	153
A.2.3	Packet scheduling	154
A.2.4	Effective loss rate π_B^* and problem statement	155
A.3	Exact analytical derivation of loss rate π_B^*	155
A.3.1	The effective loss rate π_B^* for an arbitrary schedule	155
A.3.2	The effective loss rate π_B^* for even spacing on paths	157
A.4	The design of the schedule \mathcal{S}	160
A.4.1	‘Immediate’ packet scheduling \mathcal{S}^{imm} - state of the art	161
A.4.2	‘Spread’ packet scheduling \mathcal{S}^{spr} - our proposal	162

A.4.3	Comparison of \mathcal{S}^{imm} and \mathcal{S}^{spr} : Optimal schedules \mathcal{S}_{opt}^{imm} and \mathcal{S}_{opt}^{spr} , and loss rate improvement γ	163
A.4.4	Capacity constraints	164
A.5	Performance evaluation	164
A.5.1	Simulation results	164
A.5.2	Trace-driven PlanetLab evaluation	169
A.6	Related work	172
A.7	Conclusion	173
	Bibliography	174
	Publications	187
	Curriculum Vitae	189

Chapter 1

Introduction

1.1 Motivations and Goals

Two-Layer Systems

Many technological, social, transportation or biological systems can be naturally described in terms of graphs. However, a closer look at some of these systems reveals their two- or multi-layer nature, where a number of coexisting networks interact and depend on each other. For instance, the IP graph of the Internet and the network formed by a Peer To Peer (P2P) application, although usually studied separately, are closely related. Indeed, each P2P link virtually connects two IP nodes. These two IP nodes are usually distant in the underlying IP topology, and the virtual connection is realized as a path found by IP routers. In other words, the graph formed by an application is mapped on the underlying IP network. Moreover, the IP links are in turn mapped on the physical layer that consists of a mesh of optical fibers usually buried in the ground along roads, rails, or power lines. The resulting topologies at the three layers (application, IP, physical) are very different from each other.

Another good example of a layered system is transportation networks. Indeed, many real-life transportation systems can be interpreted as a network of traffic flows mapped on top of the graph representing the physical infrastructure (roads, railways, power lines).

This thesis is about such multi-layer systems. We focus mainly on two-layer systems and describe them by two graphs at two different layers, where the upper-layer (logical) graph is mapped onto the lower-layer (physical) graph. We give a simple example of a two-layer network in Fig. 1.1.

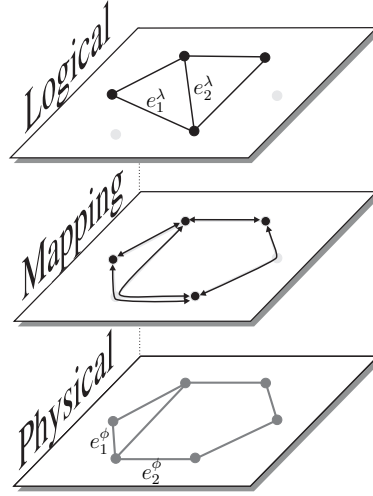


Figure 1.1: Two-layer network - the main object of study in this thesis. Every edge in the logical graph is mapped onto the physical graph. For instance, the logical edge e_1^λ is mapped as a physical path $\langle e_1^\phi, e_2^\phi \rangle$.

What Changes Under Two-Layers?

The layered view sheds new light on the tolerance to errors and attacks of many complex systems. What is observed at a single layer does not necessarily reflect well the state of the entire system. This is because a single physical component can carry multiple logical connections. For instance, one optical fiber often carries tens of IP links, one IP link can carry many application level connections, and one railway segment typically serves many trains. As a result, a tiny, seemingly harmless (from a single-layer perspective) disruption of the physical graph, may destroy a substantial or essential part of the logical graph, thus making the whole system useless in practice.

For example, in Fig. 1.1, the physical edge e_2^ϕ carries two logical links, e_1^λ and e_2^λ , and thus its single failure *propagates* to the logical layer and *multiplies*, bringing down both links.

Goals

To the best of our knowledge, the fragile nature of two-layer systems has not been comprehensively studied to date. We set the *two main goals of this thesis*, as follows.

First, we want to *study* the robustness to failures of existing two-layer systems. This should bring us some valuable insight into the problem, e.g., by identifying common weak points in such systems.

Fortunately, these two-layer problems may often be effectively alleviated by a careful system design. Therefore, our second major goal is to *propose new designs* that increase the robustness of two-layer systems.

1.2 Dissertation Outline and Contributions

General two-layer model (Chapter 2)

We begin by giving a general model of a two-layer system in Chapter 2. We discuss some of its universal aspects, such as types of failures or failure protection techniques. This chapter serves as common reference point that spans the entire thesis.

Study of Existing Large-Scale Systems (Chapter 3)

Our goal in Chapter 3 is to gain more understanding of the problem of failure propagation and multiplication by studying the existing large-scale two-layer systems. We do not constrain this study to communication networks only. On the contrary, we consider examples in fields as different as the Internet (Gnutella), transportation (railway networks) and biology (human brain).

In order to conduct this study, we first have to obtain appropriate data sets describing two-layer systems. This necessary step turned out to be relatively challenging and involved a significant *data-mining* effort. Indeed, although there are plenty of one-layer complex network data freely available, the resources are very sparse when it comes to two-layer systems. This is not surprising, as the two-layer paradigm has been little studied to date. As a result, in many cases we had to develop a dedicated methodology just to get the data to our study. To this end, we have made the following contributions.

- We propose an algorithm that constructs the topology of the physical infrastructure of a public transportation system based only on its timetables [1,2]. This is a topic on its own, and there exists at least one PhD thesis entirely devoted to it [3].
- We develop a novel methodology to infer a network of long distance cortex-to-cortex connections in the human brain, based on a diffusion MRI data [4].
- We aggregate at the Autonomous System (AS) granularity a snapshot of the Gnutella file sharing network, and we map the resulting weighted graph onto a known AS-level Internet topology.

It should be stressed that obtaining real data sets that cover both layers is crucial for our study. Indeed, it is tempting to try to generate one layer based on the information available in the other layer. However, we showed in [2] that the results of such attempts significantly differ from the real-world patterns.

Once the two-layer data sets are prepared, we can study their vulnerability to various failures. We simulate two types of physical failures: *errors* - failures of randomly chosen components, and *attacks* - failures of components that play a vital role in the system. Moreover, in order to cover the whole range of features specific to these systems, we focus on two extreme policies of system's response to failures, (i) no rerouting and (ii) full rerouting. Our main findings, common to all data sets are [5]:

- Attacks are much more harmful than errors. For example, in Gnutella with no rerouting, half of the logical topology is erased after 22% physical errors, or after only 0.04% attacks.
- When the system is attacked, the logical graph is usually affected much faster than the physical graph. For instance, in Gnutella, an attack on 5% of the physical edges hardly affects the physical graph, but deletes more than 95% of logical edges. This result underlines the fundamental difference between the single-layer (physical layer) view typically considered in the previous studies, and the two-layer view advocated in this thesis.
- A heterogeneous logical topology makes the system more vulnerable to attacks. We reach this conclusion based on two artificial two-layer system models that we included in our study.

IP/WDM Design (Chapter 4)

The study of existing large-scale systems in Chapter 3 confirms that two-layer systems are much more vulnerable to errors and intentional attacks than they appear from a single layer perspective. This raises a natural question: Can we increase the robustness of a two-layer system by its appropriate *design*? Fortunately, in most cases it is possible. However, two-layer systems come in a wide range of variations. Depending on a setting, we may face different degrees of freedom (some parameters may be fixed in some scenarios, and free to change in other scenarios), types of failures (edge/node, physical/logical), constraints (delay, capacity, technology) and performance metrics. Consequently, there is no 'one-fits-all' two-layer design solution.

For this reason, we consider the design problem in the context of two specific two-layer systems. We address them in Chapter 4 and 5, respectively.

In Chapter 4, we consider IP/WDM networks. The IP/WDM technology allows the owner to set up an IP backbone network on top of a mesh of physical fibers spanning some geographical region. Such a system is typically owned by a single ISP (Internet Service Provider) and consists of 10-100 nodes. As the physical network is composed of the existing fibers, and the logical graph is closely related to the traffic demands, these two layers are often considered given and fixed. Then the problem boils down to designing an appropriate mapping, called a *survivable* mapping.

Finding a survivable mapping is known to be NP-complete, making it impossible in practice to assess the existence or absence of such a mapping for large networks. We propose a novel approach to this problem, which results in the following contributions:

- We introduce a new concept of *piecewise survivability*, which makes the problem much easier in practice (although still NP-complete).
- Through a graph-theoretic analysis we demonstrate a number of properties of a piecewise survivable mapping. This allows us to *formally prove* that a given survivable mapping does or does not exist [6,7]. A crucial feature of our approach is that it does not have to backtrack and exhaustively search the solution space, which results in a great scalability.
- We provide a technique that finds vulnerable areas in the system, and that helps to effectively strengthen them, e.g., by adding new links. As a result, we enable a survivable mapping in systems for which such a mapping originally does not exist.
- We give an efficient and scalable algorithm that finds a survivable mapping much faster (often by orders of magnitude) than the other approaches proposed to date [8].
- Our approach covers the types of various physical components, such as links, nodes, spans, or double-failures [9].

Overlay/IP Design (Chapter 5)

An Overlay/IP system is an application-level network (typically spanning a number of end-hosts running the application) built on top of the Internet. As the logical links are virtual, we are free to construct and change the logical

layer. In contrast, as regular Internet users (and not an ISP), we have no direct control over the physical (IP) network nor over the mapping (IP paths are determined by IP routers).

In particular, we consider the Application-Level Multicast (ALM) for interactive applications, where a single source sends a delay constrained data stream to a number of destinations. Such a system should (i) respect stringent delay requirements, and (ii) proactively protect the system against overlay node failures and against (iii) the packet losses at the IP layer. We propose a two-layer-aware approach to this problem, which results in the following contributions [10]:

- We define a metric called Vulnerability \mathcal{W} , based purely on the *topology* of the ALM system. We prove that, under some assumptions, \mathcal{W} is equivalent to the average packet loss rate observed at the destinations. This crucial observation allows us to focus directly on minimizing \mathcal{W} , which significantly simplifies the problem.
- We note that \mathcal{W} drops with the amount of *IP-level and overlay-level path diversity* available in the system. Therefore, we consider various techniques to create a number of alternative paths in ALM, such as adding redundant cross-links, or using a set of multiple distribution trees. We propose a framework that accommodates and generalizes these approaches.
- Within this framework, we optimize the structure of ALM. As a result, we reduce the Vulnerability \mathcal{W} (and thus the effective loss rate) of real Internet topologies by typically 30%-70%, compared to the state of the art.
- Moreover, we develop a set of lower-bounds on \mathcal{W} and show that our approach is nearly optimal.
- Finally, we study factors that naturally limit the available IP-path diversity, such as the system size and maximal allowed delay.

Finally, in Chapter 6 we conclude the thesis.

Multipath Transmission with FEC (Abstract A)

There are many side products of this thesis, but one is especially worth mentioning. It is a set of results related to ‘Multipath Transmission with FEC’. Although our work on this topic was originally inspired by the Overlay/IP

design problem described in Chapter 5, it does not fall in the ‘two-layer’ core of this thesis. For this reason, we moved this part to the Appendix.

By ‘Multipath Transmission with FEC’ we mean the transmission of a delay-sensitive data stream from a single source to a single destination, where the data packets are protected by Forward Error Correction (FEC) code and sent over multiple paths. It has been shown before that such an approach improves the transmission’s robustness to bursty packet losses - the predominant type of failures in today’s Internet.

We show that the performance of such a multipath FEC scheme can often be further improved. In particular, we make the following contributions [11]:

- First, based on real-life data, we observe that the propagation times on the available paths often significantly differ, typically by 10-100ms.
- We propose to exploit these differences by appropriate packet scheduling that we call ‘Spread’.
- We evaluate our solution with a precise, analytical formulation, and by trace-driven simulations. Our studies show that Spread substantially outperforms state-of-the-art solutions. It typically achieves a two- to five-fold improvement (reduction) in the effective loss rate.
- Or conversely, keeping the same level of effective loss rate, Spread significantly decreases the observed delays and helps fight against the delay jitter.

1.3 Related Work

Two-layer structures (as defined above) may arise in various scenarios, ranging from a number of Internet aspects (optical networks, P2P streaming) to transportation and biology. One of the challenges of this thesis is to go across these scenarios and research communities built around them, using the two-layer model as a spanning factor. To the best of our knowledge, to date nobody has ever done this. Consequently, there is no related work that speaks about two-layer systems *in general*.

In contrast, each of these specific fields is covered by a fair amount work, that, sometimes, explicitly or implicitly, speaks about two layers. Due to its content-specific nature, we review the related work in each chapter separately.

Chapter 2

General Two-Layer Systems

In this chapter we give a general description of a two-layer system. This is a common reference point that spans all chapters of this thesis.

2.1 Two-Layer Structure

A two-layer system consists of two distinct layers, each containing a different graph. The *logical graph* at the logical layer is *mapped* onto the *physical graph* at the physical layer. We explain these terms below. Please refer to Fig. 2.1 for illustration, and to Table 2.1 for a summary of notation.

2.1.1 Physical Graph $G^\phi = (V^\phi, E^\phi)$

The physical graph $G^\phi = (V^\phi, E^\phi)$ is the transportation network that actually carries the traffic. We use the superscript ϕ to denote components that belong to the physical layer. The physical graph can be directed or undirected, depending on the application.

2.1.2 Logical Graph $G^\lambda = (V^\lambda, E^\lambda)$

The logical graph $G^\lambda = (V^\lambda, E^\lambda)$ is a network built on top of the physical graph. Its typical purpose is to simplify the management of the system or to enable services that are not directly available at the physical layer. We use the superscript λ to denote components that belong to the logical layer.

There are at most $|V^\lambda| \leq |V^\phi|$ logical nodes. Every logical node $v^\lambda \in V^\lambda$ has its unique physical counterpart $v^\phi \in V^\phi$. To simplify the notation we will sometimes (where it is obvious from the context) use v^ϕ to denote the physical node associated with v^λ ; consequently, we can write that $V^\lambda \subseteq V^\phi$.

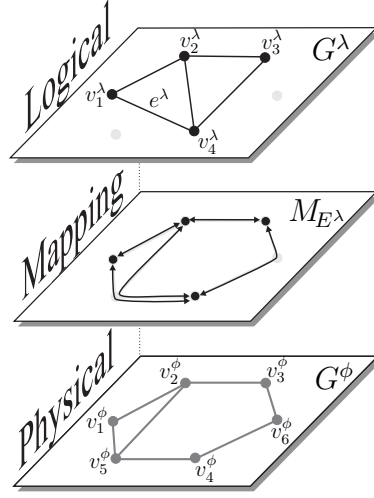


Figure 2.1: From top to bottom: the logical graph $G^\lambda = (V^\lambda, E^\lambda)$, the mapping M_{E^λ} , and the physical graph $G^\phi = (V^\phi, E^\phi)$. The logical nodes are a subset of physical nodes $V^\lambda \subseteq V^\phi$.

In contrast to logical nodes, the logical edges E^λ can be (and usually are) completely different from the physical edges E^ϕ .

Finally, the logical graph can be directed or undirected, depending on the application.

2.1.3 Mapping M

Let $p_{v,u}$ denote a path from vertex v to vertex u . Let P^ϕ be a set of all paths in the physical graph G^ϕ . We say that the logical graph is *mapped* onto the physical graph. In its widest sense, a mapping $M : \{\langle u^\lambda, v^\lambda \rangle : u^\lambda \neq v^\lambda\} \mapsto P^\phi$ associates with every ordered pair $\langle u^\lambda, v^\lambda \rangle$ of logical nodes a physical path $M(\langle u^\lambda, v^\lambda \rangle) = p_{u^\lambda, v^\lambda}^\phi$ going from u^λ to v^λ in G^ϕ . (To stress that the mapping M is defined for all pairs of logical nodes, we will sometimes denote it by a more explicit term M_{all} .) If the node pair is actually some existing logical edge $e^\lambda = \langle u^\lambda, v^\lambda \rangle$, then we can write $M(\langle u^\lambda, v^\lambda \rangle) = M(e^\lambda) = p_{u^\lambda, v^\lambda}^\phi$. For example, in Fig. 2.1 we have $M(e^\lambda) = M(\langle v_1^\lambda, v_4^\lambda \rangle) = \langle v_1^\phi, v_5^\phi, v_4^\phi \rangle$.

However, in some applications we will be interested not in all possible node pairs, but only in some subset $A \subset \{\langle u^\lambda, v^\lambda \rangle : u^\lambda \neq v^\lambda\}$ of them. This is denoted as $M_A : A \mapsto P^\phi$. For arguments beyond A , M_A is not defined. In particular, we often speak of a mapping $M_{E^\lambda} : E^\lambda \mapsto P^\phi$ of the set E^λ of existing logical edges. The example in Fig. 2.1 actually presents M_{E^λ} , and not M_{all} .

$G = (V, E)$	graph G with a set of nodes V and edges E
ϕ	upper index denoting the physical layer
λ	upper index denoting the logical layer
$G^\phi = (V^\phi, E^\phi)$	physical graph
$G^\lambda = (V^\lambda, E^\lambda)$	logical graph
$v^\lambda \in V^\lambda$	a logical node; sometimes used to refer to the underlying (and always unique) physical node
$p_{u,v}$	path from node u to v
$M(\langle u^\lambda, v^\lambda \rangle)$	mapping of a node pair $\langle u^\lambda, v^\lambda \rangle$, i.e., a physical path $p_{u^\lambda, v^\lambda}^\phi$
P^ϕ	all paths in the physical graph G^ϕ
M, M_{all}	mapping of all node pairs, $M : \{\langle v^\lambda, u^\lambda \rangle : v^\lambda \neq u^\lambda\} \mapsto P^\phi$
M_A	restricted mapping $M_A : A \mapsto P^\phi$

Table 2.1: General notation used in this thesis.

2.2 Failures

There are many types of potential failures in a two-layer system. Not all of them are equally important. For example, a given failure may be common in one application, and irrelevant in another one. For every two-layer scenario considered in this thesis, we address different types of failures (see Table 2.2 at the end of this chapter).

Below, we discuss the types of potential failures with respect to three different aspects. Please, refer to Fig. 2.2 for an illustration.

2.2.1 Edge Failures vs. Node Failures

First, there are two basic elements that can fail in a graph: edges and nodes. In practice, the failure of a node is equivalent to the failure of all edges adjacent to it. For instance, in Fig. 2.2a, the failure of v_1^λ deletes its two neighboring edges and reduces the logical topology to a triangle.

2.2.2 Logical Failures vs. Physical Failures (and failure propagation)

Second, the failures may occur at the logical and/or at the physical layer. Logical failures affect the logical layer only, and are transparent to the physical layer, as presented in Fig. 2.2a and Fig. 2.2b, respectively.

In contrast, physical failures affect not only the physical layer, but also *propagate* to the logical layer and possibly *multiply*. For example, in Fig. 2.2b,

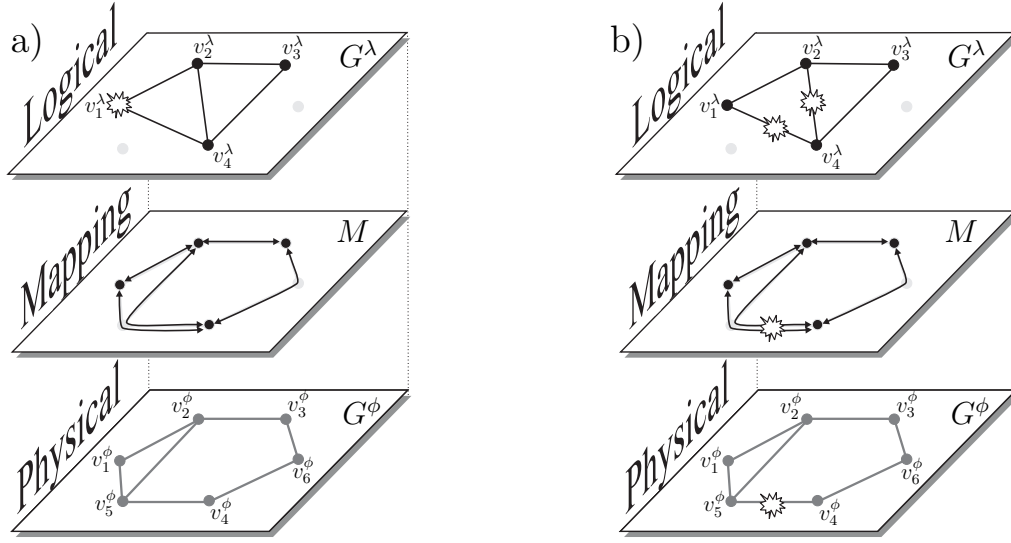


Figure 2.2: A failure of a logical node (a) and of a physical link (b).

the failure of a single physical link $\langle v_5^\phi, v_4^\phi \rangle$ brings down two logical links, $\langle v_1^\lambda, v_4^\lambda \rangle$ and $\langle v_2^\lambda, v_4^\lambda \rangle$.

2.2.3 Single vs. Multiple Failures

Third, we distinguish single and multiple failures. The ‘single failure’ category is important, because the failure probability is often very small, making multiple failures very rare (and possibly negligible) events, especially in small systems. In contrast, considering multiple failures allows us to study the behavior of large systems under typical or high stress.

2.2.4 Errors vs. Attacks

Finally, we distinguish errors and attacks. *Errors* are failures of randomly chosen components in the system. This is our basic (and default) category, because errors are a part of the regular operation of every system.

In contrast, *attacks* are failures of components that play a vital role in the system. This scenario assumes the presence of some adversary that knows the weak points of the system. For instance, the physical edge $\langle v_4^\phi, v_5^\phi \rangle$ in Fig. 2.2 is a good target for an attack, because it is the only physical edge that carries more than one logical edge.

2.3 Failure Protection Mechanisms

In the previous section we described the types of failures that we may encounter in a two-layer system. Here we overview the general techniques to make the system more robust to such failures.

2.3.1 Proactive vs Reactive

First, we distinguish proactive and reactive techniques. *Proactive* failure protection mechanisms constantly keep the system prepared for failures, before they actually occur. Typically, this is achieved by keeping a set of pre-computed backup paths, or by applying some redundant coding.

In contrast, *reactive* mechanisms are activated only after a failure occurs and is detected. Usually they trigger a retransmission of the lost data (for short-lived failures), or they dynamically search for a new path (for long-lived failures).

As a result, proactive techniques are less resource efficient (resources are committed without prior knowledge of the next failure) but fast, whereas reactive techniques are more resource efficient and slower.

2.3.2 Physical Layer vs. Logical Layer

Failure protection mechanisms can be provided at different layers. *Logical* layer mechanisms can handle failures that occur at both layers, contrary to *physical* layer mechanisms that are transparent to the logical topology.

In Fig. 2.3 and Fig. 2.4 we illustrate the failure protection mechanisms at the physical and logical layer, respectively.

2.4 The Notion of Vulnerability

Although in this thesis we consider a number of very different two-layer systems, in all cases the high-level goal and performance metric can be defined as follows:

Vulnerability \mathcal{W} is a metric that reflects the level of service degradation under the presence of typical failures in the system. Our main goal is to study \mathcal{W} and minimize it by an appropriate system design.

This sentence captures only a very general intuition behind the Vulnerability metric. The specific and rigorous definition depends on the context

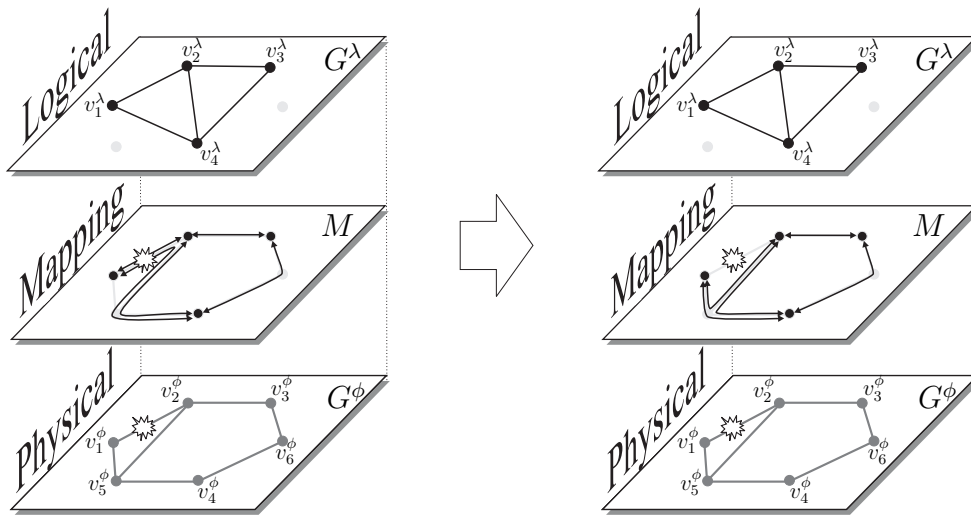


Figure 2.3: Failure protection mechanisms at the physical layer. A physical failure is detected directly at the physical layer. The affected paths are changed (to avoid the failing link) by (i) replacing by precomputed paths (proactive technique) or (ii) rerouting on the fly (reactive technique). This is in general transparent to the logical layer.

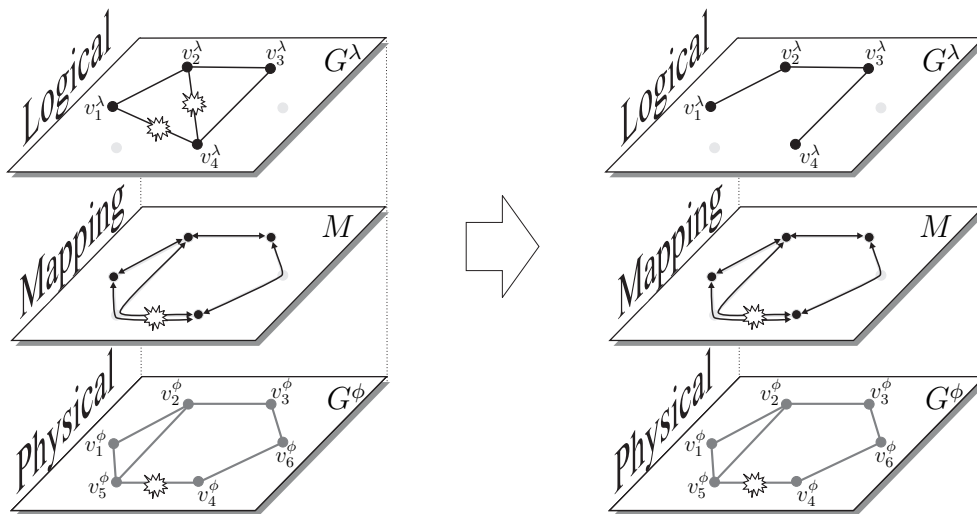


Figure 2.4: Reactive failure protection at the logical layer. A physical failure propagates to the logical layer where it is detected. The logical nodes recalculate the routes in the logical topology to avoid the failing logical link(s). For example, when v_1^λ learns about the failure of the logical edge $\{v_1^\lambda, v_4^\lambda\}$, it updates its routing table and starts sending the packets destined to v_4^λ via v_2^λ rather than directly. This is in general transparent to the physical layer.

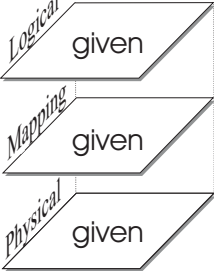
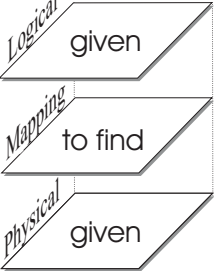
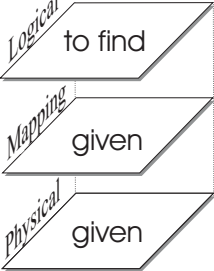
	Large-scale	IP/WDM	Overlay/IP
Chapter	3	4	5
input			
mapping domain	M_{E^λ}	M_{E^λ}	M_{all}
failures	multiple failures of physical links	single or multiple failures of physical links and nodes	single failures of logical nodes and physical links
errors or attacks?	errors and attacks	errors and attacks	errors only
failure protection	reactive, physical	reactive, logical	proactive, logical
Vulnerability \mathcal{W}	number of logical links brought down, $\mathcal{W} \in \{0, \dots, E^\lambda \}$	indication of a partition of the logical graph, $\mathcal{W} \in \{0, 1\}$	number of missed destinations, $\mathcal{W} \in \{0, \dots, V^\lambda - 1\}$
goal	Given G^ϕ , G^λ and M_{E^λ} , study \mathcal{W} of existing large-scale systems	Given G^ϕ and G^λ , find M_{E^λ} that minimizes \mathcal{W} .	Given G^ϕ and M_{all} , find G^λ that minimizes \mathcal{W} .

Table 2.2: Various aspects of the two-layer system under the three main settings considered in this thesis.

(network setting, application) and will be given in each of the following chapters.

2.5 Two-Layer Systems Addressed in This Thesis

In Chapter 1 we described three types of two-layer systems that we consider in this thesis. Table 2.2 positions these systems with respect to all the two-layer aspects discussed above. We hope it will serve as a quick and useful reference for the reader.

Chapter 3

Vulnerability of Existing Large-Scale Two-Layer Systems

In this chapter we study the vulnerability of existing large-scale two-layer systems. Our goal is to gain a better understanding of the problem of failure propagation and multiplication. We focus on large-scale systems that count thousands of nodes. Moreover, we do not constrain our studies to the field of communication only - we consider examples from transportation and biology, as well as some artificial models based on classic and power-law random graphs.

3.1 Introduction

In the last decade, the growing popularity of the Internet has made many interesting data sets easily accessible. As a result, numerous large-scale complex systems have been described in terms of graphs for the first time. The examples range from the Internet [12] and World Wide Web [13] to social [14,15], transportation [16] and biological networks. Due to the large size and nontrivial connectivity patterns of these graphs, they are often referred to as *complex networks* [17].

3.1.1 Errors vs Attacks

A natural step in the study of complex networks is to test their robustness, i.e., their behavior under stress. We distinguish two general categories of such stress, often referred to as errors and attacks. An *error* is the failure of a randomly chosen component in the system. In contrast, an *attack* is the failure of a component selected by an adversary, which usually plays a

vital role in the system (see for example the recent attacks on TGV railway lines [18]).

3.1.2 Error and Attack Tolerance of Single-Layer Systems

Complex networks may greatly differ in their response to failures. Consider for instance ‘scale-free’ networks, i.e., networks whose node degree distribution is heavy-tailed [15]. This property was found in numerous real-life systems, such as the World Wide Web, Internet, protein networks, social networks or cellular networks. Interestingly, the scale-free networks exhibit remarkable robustness to errors, but at the same time, they are very vulnerable to attacks such as the removal of the most highly connected nodes [19–22]. Subsequent work studied other attack strategies [23,24], cascading failures [25,26], defensive strategies [25,27–30], and vulnerability of weighted networks [31]. These findings give us valuable insights into the robustness of complex networks treated as distinct objects.

3.1.3 Many complex systems are layered

However, as we point out in this thesis, many complex networks are only a part of larger systems, where a number of coexisting topologies interact and depend on each other. We have introduced this concept in the context of complex networks in [2]. An obvious example is the Internet, where a graph formed by an application (such as WWW or Peer-To-Peer) is mapped onto the IP network that, in turn, is mapped onto a physical mesh of cables and optical fibers. The topology at each layer is often very different. However, there exist examples beyond communication networks too. For instance, it is convenient to view a transportation network as a two-layer system, with a network of traffic demands mapped onto the physical infrastructure.

3.1.4 Our Objective and Achievements

This layered view sheds new light on the tolerance to errors and attacks of many complex systems. Recall from Section 2.2, that one of the basic features (and problems) of a two-layer system is that the failures at the physical layer *propagate* to the logical layer and *multiply*. Therefore, the response of a layered system to failures is much more complex than what is observed at a single layer.

Our goal in this chapter is to gain insight into this phenomenon. We achieve this by simulating physical errors and attacks in five examples of two-

layer complex systems: three *real-life* data sets in the fields of communication (the Internet), transportation (the European railway system) and biology (the human brain), and two *artificial* models based on random graphs.

We show that what is observed at a single layer does not necessarily reflect well the state of the entire system. For example, a tiny, seemingly harmless (from one-layer perspective) disruption of the lower layer graph can destroy a substantial part of the logical graph making the whole system useless in practice.

3.1.5 Organization of this Chapter

In Section 3.2, we describe the real-life data sets and artificial models that we study in this chapter (the data mining details are moved to the Appendix). In Section 3.3, we specify the types of failures and protection techniques that we consider, as well as our main performance metrics. In Section 3.4, we study the distribution of load in our data sets, which gives us the first clues about the system robustness. In Section 3.5, we verify these hints by simulating directly the effect of errors and attacks on the system. Finally, we conclude the chapter in Section 3.6.

3.2 Data Sets and Models

In this section we describe the studied data sets (one per page). We also overview the data-mining techniques we used to collect some of these data sets.

3.2.1 Railway

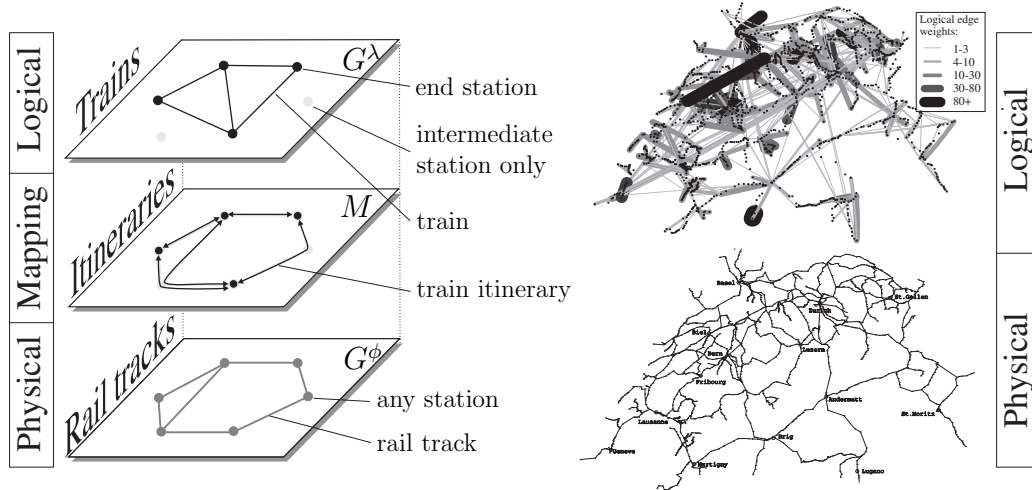


Figure 3.1: An illustration of two layers in ‘Railway’. This data set contains the entire railway network of Europe; we present its Swiss part on the right.

In [1] we have proposed an algorithm for extracting both the real physical topology and the network of traffic flows from timetables of public mass transportation systems. The algorithm consists of three phases. First, it creates the initial version of the physical topology G_0^ϕ . Typically, G_0^ϕ contains not only the existing physical links, but also some virtual *shortcut links* that result from e.g., express trains that do not stop at every station. Next, the physical graph is iteratively refined by detecting and erasing the shortcut links. Finally, we group the vehicles with identical routes, and obtain a weighted logical graph reflecting the traffic flows in the system. We give more details on this algorithm in the Appendix 3.A.

Interestingly, the problem of extracting the physical topology from timetables is non-trivial and this sole problem was the topic of another Ph.D. thesis [3]. The heuristic approach we proposed in [1] is based on simple observations that were omitted in [3], and turned out to be a much simpler and more effective algorithm.

We apply our technique to timetables of 60’775 trains in central Europe. The resulting physical graph reflects the real infrastructure that consists of 4’853 nodes (stations) and 5’765 edges (rail tracks). The logical graph contains 7’038 edges, each connecting the first and the last station of a train. The logical edge weight is the number of trains following the same route. The route itself is the mapping of this edge on the physical graph. We refer to this data set as ‘Railway’.

3.2.2 Gnutella

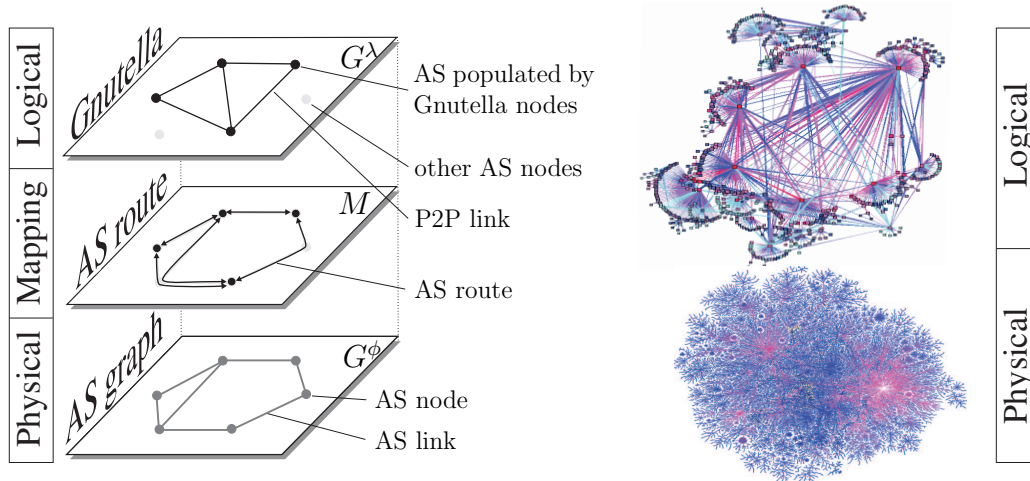


Figure 3.2: An illustration of two layers in the Gnutella data set. AS - Autonomous System

The second data set, called ‘Gnutella’, is an example of a large Peer-To-Peer (P2P) application in the Internet. In a P2P system, the links between users are virtual and are usually created independently of the underlying Internet structure, thus forming a very different topology. Due to its immense size and dynamics, the existing maps of the Internet at the IP level (i.e., where the nodes and IP routers and hosts) are very incomplete. Therefore we focus on its aggregated version, where each node is an Autonomous System (AS - usually an Internet Service Provider), and where edges reflect the connections between the ASes. The topology of the AS-level Internet is well known thanks to numerous Internet mapping projects such as DIMES [32] or CAIDA [33]. For our physical graph we take the 09/2004 topology provided by CAIDA, which consists of 16’911 nodes and 37’849 edges. For the logical graph we take a snapshot of the Gnutella P2P network collected in September 2004 by the crawler developed in [34]. It consists of around 1 million users, connected by several million links. In order to obtain the AS-level version of this network, we translated the IP addresses of the users into the corresponding AS numbers. All users with the same AS number are grouped in a single node of the logical graph, and all links connecting the same pair of ASes become one logical edge of weight equal to the number of contributing links. As a result, we obtain an AS-level logical graph of Gnutella with 1’214 nodes and 31’193 edges. The mapping of each logical edge is defined by the shortest path in the physical graph connecting its end-nodes.

3.2.3 Brain

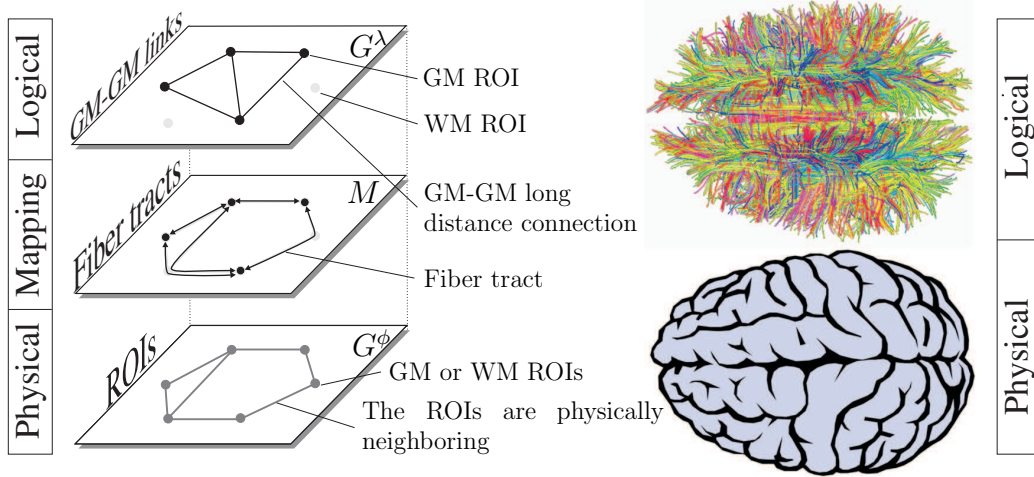


Figure 3.3: An illustration of two layers in the Brain setting. WM - White Matter, the interior part of the brain. GM - Gray Matter, the exterior part of the brain, i.e., the brain cortex. ROI - Region Of Interest, small and compact part of GM or WM.

Our third data set, called ‘Brain’, is a millimetric scale map of the structural connectivity of the entire human brain. It was inferred from a diffusion MRI scan with the approach that we developed in [4]. This methodology partitions the brain gray matter (GM) and white matter (WM) into a set of compact regions of comparable size, called ROIs (‘Regions Of Interest’). There are 1’013 ROIs in the gray matter and 3’432 ROIs in the white matter. Every (GM or WM) ROI becomes a node in the physical graph (i.e., $|V^\phi| = 4445$) and every GM ROI becomes a node in the logical graph. The logical edges E^λ in this data set reflect the existing fiber tracts (bundles of axons) connecting different gray matter regions. Each such tract e^λ traverses the white matter; the sequence of white matter ROIs on its path defines the mapping $M(e^\lambda)$. At the physical layer, two nodes are connected by a physical edge e^ϕ if they appear to be directly connected (i.e., they are consecutive in the sequence $M(e^\lambda)$ of ROIs) in at least one mapping $M(e^\lambda)$. By this procedure, we obtain a two-layer structure, where the logical graph consists of the long-range GM-to-GM connections in the brain and is mapped on the physical layer that reflects the ‘axonal wiring’ used to establish these connections.

We give more details on inferring the brain topology from the diffusion MRI data in in the Appendix 3.B.

3.2.4 ‘ER on ER’ and ‘BA on ER’

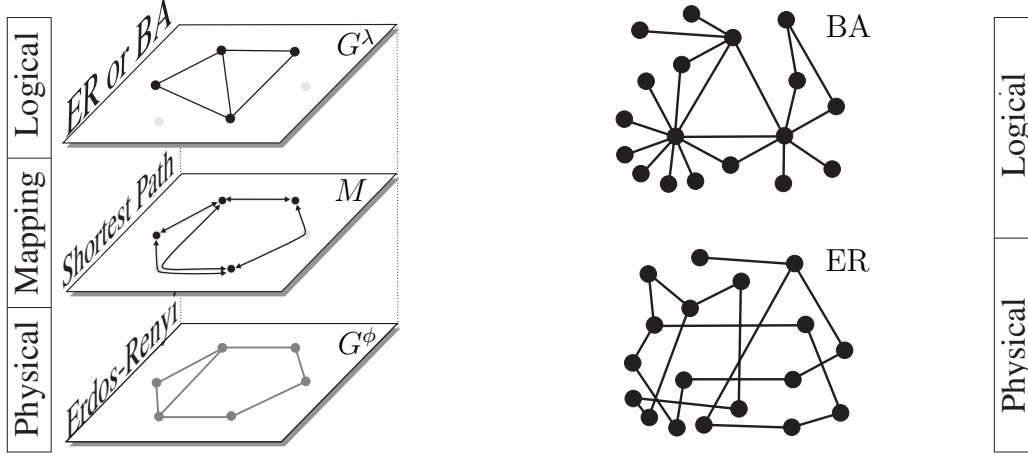


Figure 3.4: Two layers constructed by two artificial graph models: ER - classic Erdős-Rényi random graph; BA - Barabási-Albert random graph where the node degree distribution follows a power-law. We use a shortest path mapping.

Finally, as a reference point, we also study two artificial systems. ‘ER on ER’ is the classic unweighted Erdős-Rényi (ER) random graph on top of another ER graph of the same number of nodes. We consider only the largest connected component of these graphs. The nodes in the logical and the physical layer are randomly paired. ‘BA on ER’ is constructed in the same way, except that the logical graph is now the Barabási-Albert (BA) power-law random graph [15].

In Table 3.1 (below) we present an overview of the systems we study.

Data set	$ V^\phi $	$ E^\phi $	$\langle l \rangle$	$ V^\lambda $	$ E^\lambda $	$\langle m \rangle$
Railway	4'853	5'765	53.8	2'509	7'038	9.9
Gnutella	16'911	37'849	3.7	1'214	31'193	2.8
Brain	4'445	20'967	9.1	1'013	15'369	10.3
ER on ER	2'000	4'000	5.7	2'000	10'000	5.7
BA on ER	2'000	4'000	5.7	2'000	10'000	5.7

Table 3.1: The overview of the two-layer data sets studied in this chapter. $\langle l \rangle$ is the average shortest path length; $\langle m \rangle$ is the average mapping length.

3.3 Failures, Protection Techniques, and Vulnerability

In this section we specify the types of failures and protection techniques that we consider. We also define the Vulnerability metric \mathcal{W} . Please, refer to Chapter 2 for a general overview.

3.3.1 Failures

We consider failures of *physical edges*. To study the whole spectrum of stress strength applied to the system, we allow for *multiple* failures. To simplify the interpretation, we speak about the *failing fraction* of physical edges rather than their absolute number.

Errors

Simulating errors is straightforward - the failing physical links are chosen at random.

Attacks

However, it is not obvious how to select the targets of an attack. Our approach is based on the intuitive notion of *load*. The load l of the physical edge e^ϕ is the sum of weights w of all the logical edges whose mappings traverse e^ϕ [2]:

$$l(e^\phi) = \sum_{e^\lambda: e^\phi \in M(e^\lambda)} w(e^\lambda). \quad (3.1)$$

In our scenario, the adversary selects the failing physical edges one by one, removing the most loaded one at each iteration.

It is worth mentioning that there have been also various attempts to predict the load under the limited information, e.g., based purely on the topology of the physical graph. Classical load estimators are *node degree* and *betweenness* [23,25,26,35–37]. However, we have shown in [2] that the patterns formed by the distributions of the real load, node degree and betweenness significantly differ. We explained this by recasting these metrics in a two-layer perspective, which revealed some fundamental differences between them. We concluded that it is in general impossible to predict the real load based purely on the topology of the physical graph.

3.3.2 Protection Mechanisms

Many real-life systems have mechanisms to partially or fully recover from failures. In order to capture the whole spectrum of failure protection techniques, we study two extreme policies: *no rerouting* and *full rerouting*.

Under *no rerouting* we make no failure recovery attempt - we delete immediately all logical edges affected by a physical failure. This is close to what we could expect in railway networks. Indeed, when a railtrack fails it may be difficult to find an appropriate detour, because for a train its entire path is important, not only its end-points. Although it is sometimes possible to slightly change the itinerary of the train or to organize alternative means of transportation (e.g., a bus) around the failing section, the common practice is to halt all the trains that use it [18].

In contrast, *full rerouting* is equivalent to the physical-level failure restoration described in Chapter 2. We delete an affected logical edge e^λ only when there is no path in the physical graph G^ϕ between the end-nodes of e^λ , i.e., when the end-nodes of e^λ belong to different components of G^ϕ . Otherwise, the logical edge e^λ remains in the graph, and its mapping is updated by the shortest path in G^ϕ , with the failing physical edge removed.

By studying the two extreme policies, no rerouting and full rerouting, we also capture the specific features of our three real-life data sets. For instance, in the Gnutella data set, the AS graph routing depends on the internal policy of involved ASes and peering relationships established between the ASes [38]. This results in routes that are not necessarily the shortest possible and makes some of the routes invalid. These additional constraints naturally limit the Gnutella performance below the ‘full rerouting’ level. The brain also has some ability to reroute around broken connections by activating parallel pathways; this is called plasticity. For example, after a stroke in primary or secondary motor cortices some limb functions can be recovered in the animal as well as in the human by recruiting alternative pathways. These processes take, however, substantial time. Therefore, the brain response can be described as moving from the no rerouting policy just after the insult, to a partial rerouting policy during the recovery process.

In other words, all responses of real systems to physical failures are located somewhere between the no rerouting and the full rerouting policy.

3.3.3 Vulnerability Metric

What ultimately counts in a two-layer system is the logical layer. Indeed, it directly reflects the service provided by the system to its users, such as trains, P2P application and the long distance connections in the brain. Therefore,

our primary goal is to study how the physical edge failures affect the logical layer. We capture it by observing the *total weight of the remaining logical edges* in the logical graph affected by the failures.

In the other words, to put in in the terminology of Chapter 2, we define the Vulnerability \mathcal{W} as the total weight of logical edges that were brought down by the failures.

For a comparison, we will also study the largest connected component in the physical graph G^ϕ , which is a classic error tolerance metric used in one-layer complex networks [19].

3.4 Edge Load Distribution

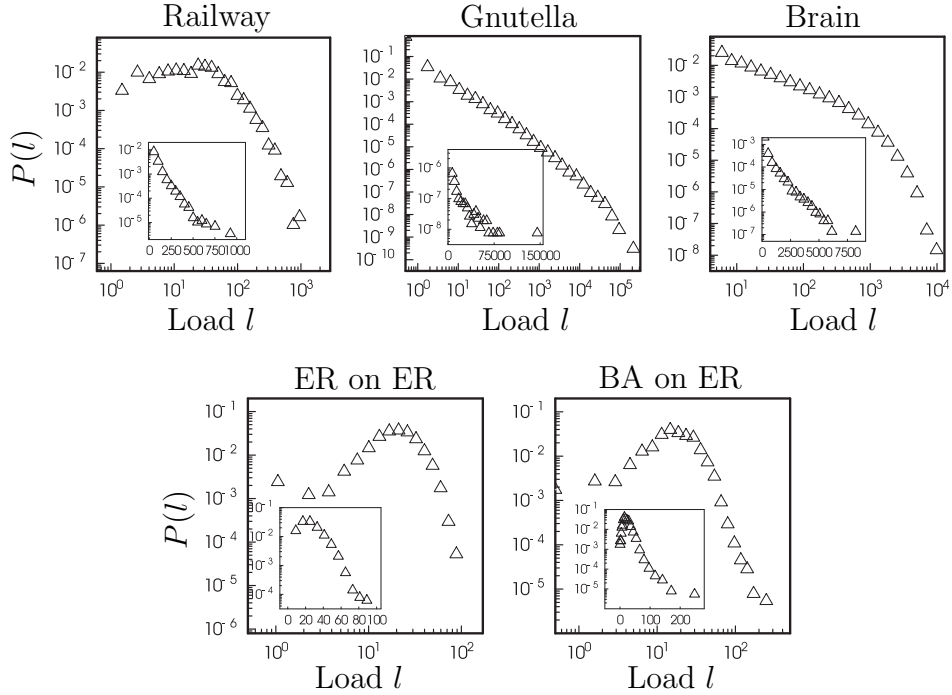


Figure 3.5: Edge load distribution in the five studied systems. The main plots are in \log_{10} – \log_{10} scale (\log_{10} -binned); the insets present the same distributions in \log_{10} -lin scale (lin-binned).

Before we simulate the effect of failures on our systems directly, we try to predict the results by studying the load distribution in our data sets. It is a very important parameter for a system under stress. Clearly, the higher the load of a failing physical component, the more it perturbs the logical layer.

If the load is distributed evenly in the physical graph, a random failure will not be very different from an intentional attack. Conversely, if the load distribution is very uneven, the highly loaded parts become an obvious target for an efficient attack.

In Fig. 3.5 we present the load distribution in the layered systems we study. In each case the distribution is broad and heavily right-skewed (maybe except ‘ER on ER’). This means that there is a significant number of physical links that carry a lot more traffic than the other links. Consequently, we can anticipate that an attack targeted on the most loaded links will harm the system much more importantly than a random error.

3.5 Simulation Results

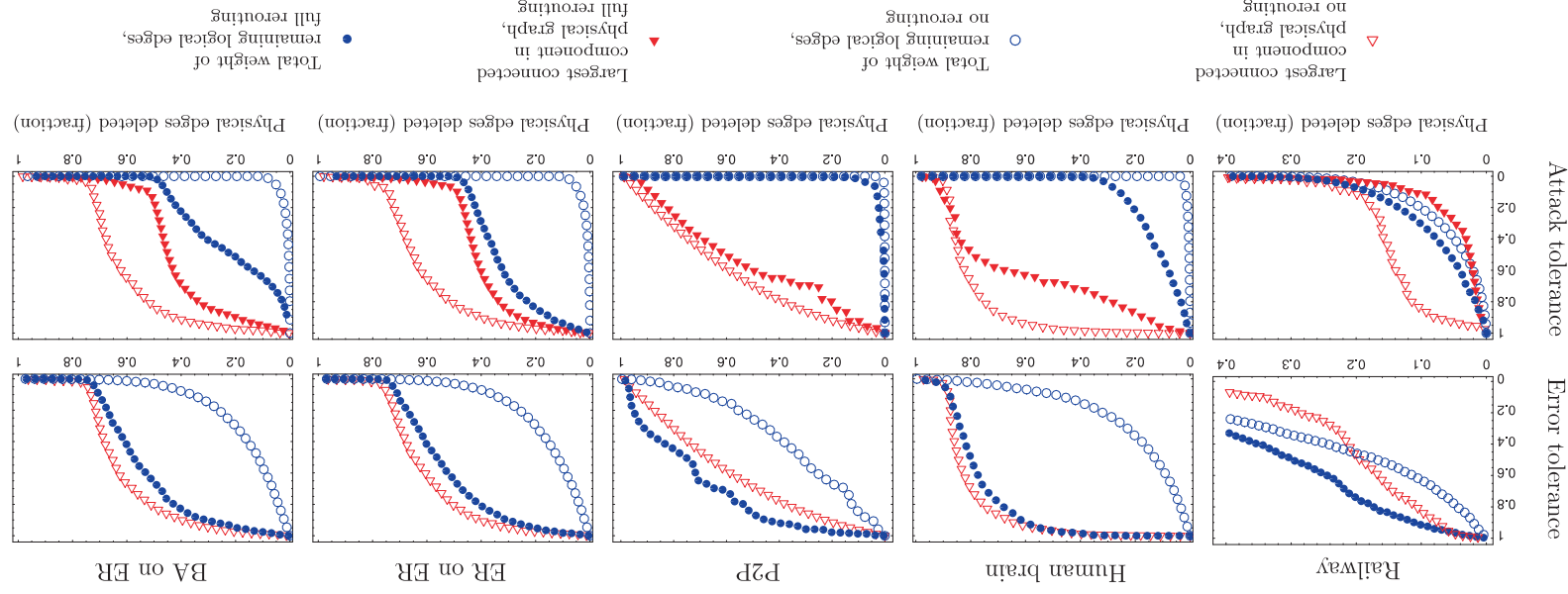
In this section we simulate the error and attack scenarios on the five studied systems. The results are presented in Fig. 3.6. Although the system responses differ in all five cases, they share a number of common features that lead us to the following conclusions:

Attacks are much more harmful than errors

Consider, for instance, the Gnutella network. Under ‘no rerouting’, half of the logical mass (total edge weight) is erased after 22% physical edges randomly fail, or after only 0.04% most loaded edges are attacked. Although under the ‘full rerouting’ policy this difference is smaller, we still need about 60 times more random failures than attacks to achieve the same goal.

When the system is attacked, the logical graph is usually affected much faster than the physical graph

For instance, in Gnutella, an attack (with or without rerouting) on 5% of the physical edges hardly affects the physical graph - the largest connected physical component covers almost the entire original graph. At the same time, this seemingly unharmed attack deletes more than 95% of logical edges! We obtain similar results when we consider the size of the largest connected component in the logical graph as the measure of robustness. (These results are not shown in Fig. 3.6 for better readability.)



The attack under the full rerouting policy affects the physical graph more than under no rerouting

When rerouting is allowed, the logical edges are deleted only when the physical graph gets partitioned. This, in turn, effectively reduces the size of the largest connected physical component.

Rerouting does not always help much

This observation is expressed by the proximity of the filled and unfilled circles under attack in Fig. 3.6 (see e.g., Railway and Gnutella). As any real-life failure recovery policy falls between these two extremes (no rerouting and full rerouting), such systems are especially vulnerable to attacks.

A heterogeneous logical topology makes the system more vulnerable to attacks

This can be observed in the two random-graph-based examples. As the node degree distribution of the BA graph follows a power-law, there is no typical node (or ‘scale’), and hence ‘BA on ER’ is a system with a heterogeneous logical topology. In other words, there is a non-negligible probability of existence of hubs (nodes of a very high degree) in the BA graph. As we assume no correlation between the logical and physical node degrees, in the vicinity of such logical hubs the load of physical edges is usually high. This makes the ‘BA on ER’ system vulnerable to attacks, which is reflected by the fast initial drop of both circle curves in the last subfigure in Fig. 3.6. In contrast, the degree distribution of the ER graph is concentrated around the average value making it much more homogeneous. So there are no hubs in the logical graph of ‘ER on ER’ and the load is distributed more evenly. This can be observed in Fig. 3.5 - with the same number of nodes and edges, the maximal load in ‘BA on ER’ is roughly three times higher than in ‘ER on ER’. Consequently, ‘ER on ER’ is more robust to attacks than ‘BA on ER’.

3.6 Conclusion: Single-Layer vs. Two-Layer View

The one-layer analysis, as typically done to date [19], is often insufficient to capture all the crucial features of a system, such as the end-to-end traffic patterns or recovery schemes. As a result, it may often lead to biased conclusions. In contrast, the two-layer view offers much better modeling capabilities and more insight into the system.

As a takeaway example of this difference between the single-layer and the two-layer framework, recall that even a tiny disruption of the physical graph can destroy a substantial part of the logical graph. This effect would be completely unnoticed within the single-layer physical graph framework. Fortunately, the two-layer approach directly captures the changes in the logical graph. This is very important, because what ultimately counts in a two-layer system is the logical layer; it directly reflects the service provided by the system to its users, such as trains, P2P application, and the long distance connections in the brain.

APPENDIX

3.A Trainspotting: Extraction of Transportation Network Topologies from Timetables

In this section we give more details of our algorithm to extract the physical and logical network data from timetables. We used it to construct the ‘Railway’ data set. The full description can be found in [1].

Spaces

In order to position our contribution in the range of works in the field, we begin with a systematic definition of the topology of transportation systems. The set of nodes is defined by the set of all stations (train stations, bus stops, etc). It is not obvious, however, what should be interpreted as an edge. Its choice depends on what we want to be reflected by the topology of the physical graph. In the literature there are essentially three approaches that define three different ‘spaces’: here we call them ‘space-of-changes’, ‘space-of-stops’ and ‘space-of-stations’.

In **space-of-changes**, two stations are considered to be connected by a link when there is at least one vehicle that stops at both stations. In other words, all stations used by a single vehicle are fully interconnected and form a clique. This approach neglects the physical distance between the stations. Instead, in the resulting topology, the length of a shortest path between two arbitrary stations A and B is the *number of changes* of mean of transportation one needs to get from A to B . This approach was used in [39–41]; in the latter the authors used the term *space P*.

In **space-of-stops**, two stations are connected if they are two consecutive stops on a route of at least one vehicle [41]. Here the length of a shortest path between two stations is the minimal *number of stops* one needs to make. Note that the number of stations traversed on the way might be larger, because the vehicles do not necessary stop on all of them.

In **space-of-stations**, two stations are connected only if they are physically directly connected (with no station in between). This reflects the topology of the real-life infrastructure. Here, the length of a shortest path between two stations is the minimal *number of stations* one has to traverse (stopping or not). This approach was used in [42–45].

In Fig. 3.7 we give an illustration of the three spaces. It is easy to see that the graph in space-of-stations is a subgraph of the graph in space-of-stops, which in turn is a subgraph of the graph in space-of-changes.

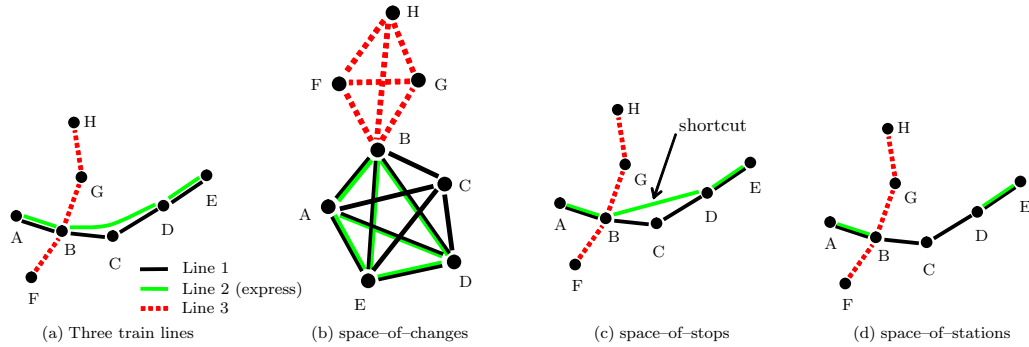


Figure 3.7: (Color online) An illustration of the transportation network topology in three spaces. (a) The routes of three vehicles. The route of Line 2 passes through node C on the way from B to D, but the vehicle does not stop there. (b) The topology in space-of-changes. Each route results in a clique. An edge is indicated by two colors, when it originates from two routes, but is merged into a single link. (c) The topology in space-of-stops. The “shortcut” B-D is a legitimate edge in this space. (d) The topology in space-of-stations. This graph reflects the topology of the real-life infrastructure.

The Difficulty of the Problem

The topologies in space-of-changes and space-of-stops can be directly obtained from timetables. In space-of-changes, for each vehicle, we fully connect all stations it stops at. Then we simplify the resulting graph by deleting multi-edges. In space-of-stops, we connect every two consecutive stops in routes of vehicles. As shown in Fig. 3.7c, the topology in space-of-stops can have shortcut links that do not exist in the real-life infrastructure. These shortcuts should be eliminated in the space-of-stations topology, which makes it more challenging to obtain. To the best of our knowledge, the only work on extracting the real physical structure (the topology in space-of-stations) from timetables was done in the context of railway networks in the PhD dissertation of Annegret Lebers [3]. The proposed solution first obtains the physical graph in space-of-stops. Next, specific structures in the initial physical graph, called *edge bundles*, are detected. The Hamilton paths¹ within these bundles should indicate the real (non-shortcut) edges. Unfortunately, the bundle recognition problem turned out to be NP-complete. The heuristics proposed in [3] result in a correct real/shortcut classification of 80% of edges in the studied graphs. The approach we propose in this paper is based on simple observations that were omitted in [3]. This results in a much simpler and more effective algorithm.

¹*Hamilton path* is a path that passes through every vertex of a graph exactly once

Timetable Data

We take a list of all vehicles departing in the system within some period (e.g., one weekday). Denote by $R = \{r_i\}_{i=1..|R|}$ the list of routes followed by these vehicles, where $|R|$ is the total number of vehicles. A route r_i of i th vehicle is defined by the list of nodes it traverses. Note that since there are usually more vehicles (than one) following the same path on one day, some of the routes may be identical.

Algorithm

The algorithm consists of four phases, as follows:

Phase 1 - Initialization

In this phase we interpret every two consecutive nodes in any route $r_i \in R$ as directly connected. Consequently, we connect these nodes with a link, which can be written as $E_{stop}^\phi = \bigcup_{i=1..|R|} E(r_i)$, where $E(r_i)$ is the set of all pairs of adjacent nodes in r_i (i.e., all edges in r_i). This results in the physical topology $G_{stop}^\phi = (V^\phi, E_{stop}^\phi)$ in space-of-stops.

Phase 2 - Deleting shortcuts

In this phase, at each iteration, we detect a shortcut in the set of physical edges, delete it, and update all routes r_i that use this shortcut. Denote by $e_{(1)}^\phi, e_{(2)}^\phi$ the two end-nodes of e^ϕ , and by $\text{Rev}(P_{e^\phi})$ the reversed version of P_{e^ϕ} (the sequence from the last node to the first one). The algorithm is as follows:

1. $E_{stat}^\phi := E_{stop}^\phi$
2. Find a tuple (e^ϕ, r_i) such that e^ϕ is a shortcut for r_i :
 $e_{(1)}^\phi \in r_i$ and $e_{(2)}^\phi \in r_i$ and $e^\phi \notin E(r_i)$.
3. IF no (e^ϕ, r_i) found THEN RETURN E_{stat}^ϕ and R .
4. $P_{e^\phi} :=$ subpath of r_i from $e_{(1)}^\phi$ to $e_{(2)}^\phi$
5. FOR all $r_j \in R$ DO:
 - If $(e_{(1)}^\phi, e_{(2)}^\phi) \in r_j$ THEN replace it with P_{e^ϕ}
 - If $(e_{(2)}^\phi, e_{(1)}^\phi) \in r_j$ THEN replace it with $\text{Rev}(P_{e^\phi})$
6. $E_{stat}^\phi := E_{stat}^\phi \setminus \{e^\phi\}$
7. GOTO 2

In Step 2, we look for a physical link that is a shortcut. We declare a physical link e^ϕ to be a shortcut, if there exists a route $r_i \in R$, such that e^ϕ connects two *nonconsecutive* nodes in r_i . For example, in Fig. 3.7c, $e^\phi = \{B, D\}$ is a shortcut because it connects two not neighboring nodes in the route r_1 of Line 1. If no physical edge can be declared a shortcut, the algorithm quits in Step 3, returning E_{stat}^ϕ and R . Otherwise, in Step 4, we find the path P_{e^ϕ} that this shortcut should take. In Fig. 3.7c this path is $P_{e^\phi} = (B, C, D)$. In Step 5, we update the set of routes R by replacing every shortcut link e^ϕ in every route using it with the corresponding path P_{e^ϕ} . In our example, the updated route of Line 2 becomes $r_2 = (A, B, C, D, E)$. It is thus identical to the route of Line 1. Finally, in Step 6 we delete the shortcut e^ϕ from the physical graph. We iterate these steps until no shortcut is found (Step 2). The resulting physical graph $G_{stat}^\phi = (V^\phi, E_{stat}^\phi) \subset G_{stop}^\phi$, is a graph in space-of-stations.

Phase 3 - Grouping the same routes together

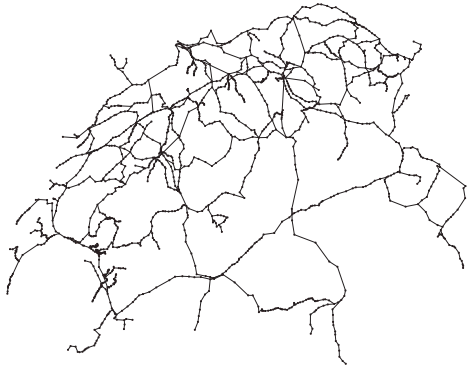
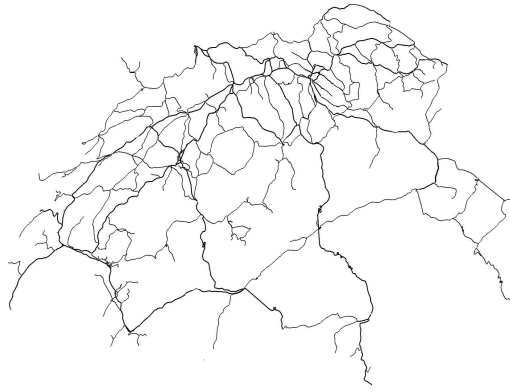
Finally, based on the list R of routes updated in phase 2, we find groups of vehicles that follow the same path (in any direction). Each such group defines one edge e^λ in the logical graph; e^λ connects the first and the last node of the route, omitting all the intermediate stations. The number of vehicles that follow this route becomes the weight $w(e^\lambda)$ of the logical edge e^λ ; the route itself becomes the mapping $M(e^\lambda)$ of e^λ on the physical graph.

Denote by $r_{i(first)}, r_{i(last)}$ the first and the last nodes in r_i , and by $E(M(e^\lambda))$ the set of all physical edges in the mapping of e^λ . Now, Phase 3 can be stated as follows:

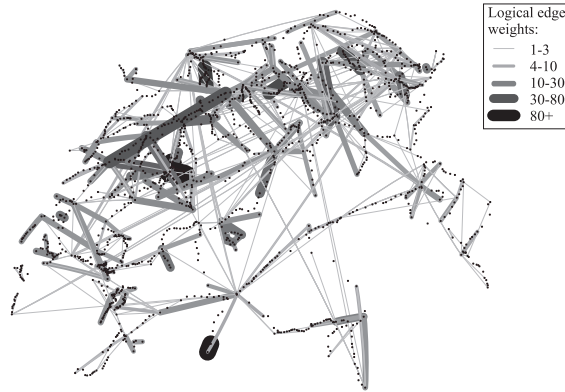
1. $E^\lambda = \emptyset, M = \emptyset$
2. FOR $i = 1$ TO $|R|$ DO:
 - $e_i^\lambda = \{r_{i(first)}, r_{i(last)}\}$
 - IF $e_i^\lambda \in E^\lambda$ THEN $w(e_i^\lambda) := w(e_i^\lambda) + 1$
 ELSE $E^\lambda = E^\lambda \cup \{e_i^\lambda\}, M(e_i^\lambda) = r_i, w(e_i^\lambda) = 1$
3. $E_{stat}^\phi = \bigcup_{e^\lambda \in E^\lambda} E(M(e^\lambda))$

In the example in Fig. 3.7, after phase 2 the routes of Line 1 and Line 2 become identical; therefore in phase 3 they are grouped together defining a logical edge $e_1^\lambda = \{A, E\}$ with the weight $w(e_1^\lambda) = 2$ and the mapping $M(e_1^\lambda) = (A, B, C, D, E)$. A second logical edge is $e_2^\lambda = \{F, H\}$ with $w(e_2^\lambda) = 1$ and $M(e_2^\lambda) = (F, B, G, H)$.

We present an example output of our algorithm for real input data in Fig. 3.8.

(a) Graph G_{change}^{ϕ} in space-of-changes(b) Graph G_{stop}^{ϕ} in space-of-stops(c) Graph G_{stat}^{ϕ} in space-of-stations

(d) Real physical map



(e) Logical graph

Figure 3.8: (Color online) The railway network in Switzerland (CH). (a) Physical graph in space-of-changes. (b) Physical graph in space-of-stops, returned by Phase 1 of our algorithm. (c) Physical graph in space-of-stations, returned by Phase 2 of our algorithm. (d) The real map of the rail tracks in Switzerland. Note the similarity with (c), which confirms the effectiveness of the algorithm. (e) The logical graph returned in Phase 3. Every edge connects the first and the last station of a particular train route; its weight reflects the number of trains following this route in any direction.

3.B From Diffusion MRI to a Brain Network

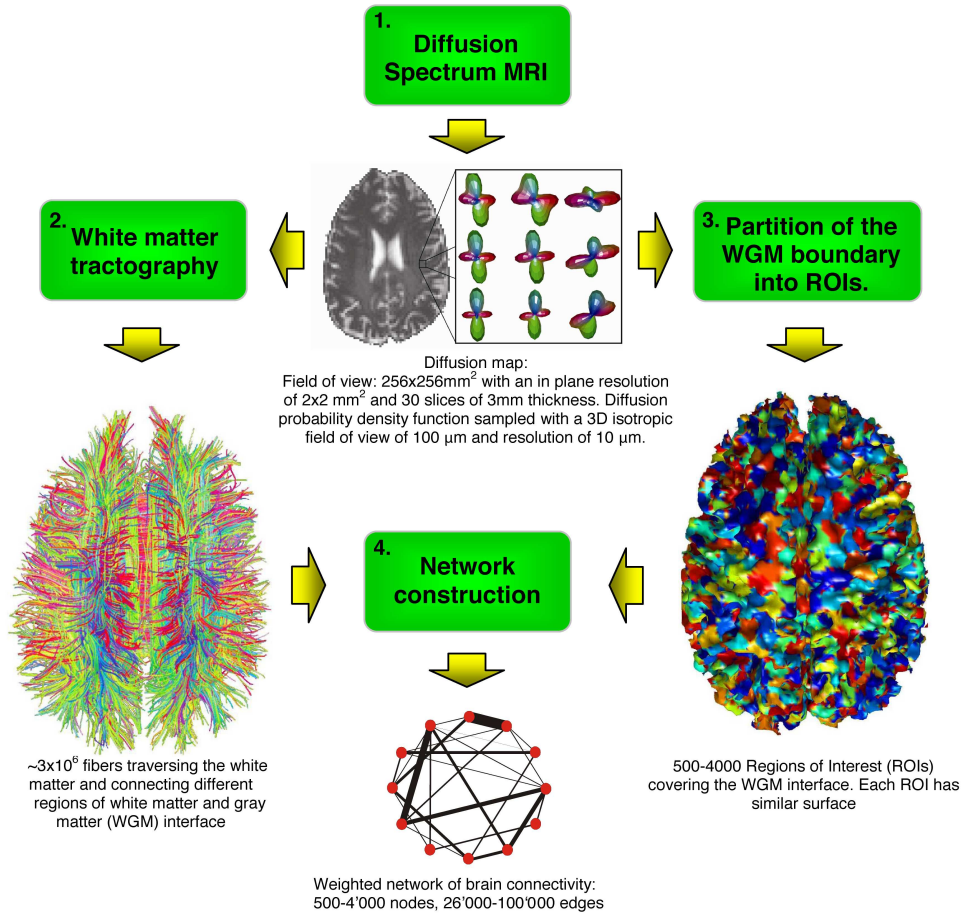


Figure 3.9: Mapping the network of brain structural connectivity with diffusion MRI is a process made of four steps. First, Diffusion Spectrum MRI (DSI) is performed on a subject or sample. This acquisition provides a 3D diffusion function at every location in the brain. This data set is called a diffusion map. It is shaped by the local tissue characteristics, in particular by the orientation of axonal bundles existing in the brain. Second, based on this map we generate a number of 3D curves (called fibers) that follow the path laid by the white matter axonal bundles. Third, independently from the previous step, we use a heuristic that partitions the brain white matter-gray matter interface into small areas of equal surface (called Regions Of Interest-ROIs) covering the whole cortex and deep cerebral nuclei boundaries. Finally, in the fourth step, we combine the output of steps two and three: the ROIs become nodes and the fibers are transformed into edges in the resulting graph. This graph estimates the density of white matter connections between any two regions of gray matter.

In this section we give more details on the methodology that we developed to create the ‘Brain’ data set. An exhaustive description with some analysis can be found in [4].

The path from diffusion MRI to a graph mapping brain connectivity is a four step process. We present a general scheme of our methodology in Fig. 3.9, and we describe in more details each step below.

Step 1: MRI acquisition

We use Diffusion Spectrum Imaging (DSI). It is a diffusion MRI method that images the 3-dimensional diffusion function in every brain voxel (a cube of size $2 \times 2 \times 3 \text{ mm}^3$) and results in a 6-dimensional image called a diffusion map. This new method has, contrary to Diffusion Tensor MRI (DTI), sufficient angular resolution to map accurately the diffusion with a non-Gaussian behavior. Accordingly, it can see intra-voxel diffusion heterogeneity caused by crossing neuronal tracts, which is essential for an accurate mapping of axonal trajectories.

Step 2: White matter tractography

Tractography is a post-processing method that based on the diffusion map, constructs 3-dimensional curves of maximal diffusion coherence. These curves, called fibers, are the estimates of the real white matter axonal bundle trajectories. We use a tractography algorithm specifically designed for DSI data to create a set of such fibers for the whole brain, which is summarized below:

First, at each voxel, we detect a set of directions of maximum diffusion. Next, we initiate the same number of fibers for every direction of maximum diffusion in every white matter voxel. The starting points are chosen at random within the voxel. From each such point we begin growing a fiber in two opposite directions with a fixed step of 1 mm. On entering a new voxel, the fiber growth continues along the direction of the vector whose orientation is the closest to the current direction of the fiber. If this results in a change of direction sharper than $15^\circ/\text{mm}$, the fiber is stopped. The growth process of a valid fiber finishes when both its ends leave the white matter.

In each data set we use about 3 million initialization points, which results in about 2 million successful fibers. For a graph of 1’000 nodes they translate into about 50’000 edges. The number of edges in the final network depends on the number of initialized fibers. To investigate network properties over a wider range of connection densities we devised two ways to filter edges. The first approach varies the number of initialized fibers, whereas the second approach keeps only the top-weight edges.

Step 3: White matter-gray matter (WGM) boundary partition into ROIs

The goal of the third step is to partition the WGM interface in a number of areas that we call Regions Of Interest (ROIs). In this step we use exclusively the 3D mask of the brain WGM interface (i.e., the cortex and the thalamus for simplicity). The ROIs should be compact and of similar surface (counted in the number of voxels), which is a non-trivial task to achieve for the complex, strongly folded shape of the brain. We have developed a two-phase partitioning heuristic, as follows. First, we choose a WGM interface voxel at random and iteratively connect it to the neighboring WGM interface voxels until it reaches the desired size; this structure becomes our first ROI. Similarly, we grow other ROIs, one by one, always starting near the ones that have already been created. We repeat this procedure until all the WGM interface is covered with ROIs. This gives us already quite a good partition, however, it can be easily further improved. Therefore, in the second phase of our heuristic we restart the ROI growth process. This time we grow all the ROIs simultaneously, starting from the centers of gravity of the ROIs found in the first phase. This results in a much better compactness of the ROIs with surface variations of less than 10%. An example of the final result is shown in step 3 of Fig. 3.9.

Step 4: Network construction

Finally, in the fourth step, we combine the output of steps two and three and create the graph of brain connectivity. Every ROI constructed in step three becomes a node in the graph. We denote by $\text{ROI}(v)$ the ROI that is associated with the node v . Two nodes v and u are connected with an edge $e = (v, u)$ if there exists at least one fiber f with end-points in $\text{ROI}(v)$ and $\text{ROI}(u)$. For each edge e we define its length $l(e)$ and weight $w(e)$, as follows. Denote by F_e the set of all fibers connecting $\text{ROI}(v)$ and $\text{ROI}(u)$ and hence contributing to the edge e . The *length* $l(e)$ of the edge e is the average over the lengths of all fibers in F_e . The *weight* $w(e)$ captures the connection density (number of connections per unit surface) between the end-nodes of the edge e , and is defined as $w(e) = \sum_{f \in F_e} 1/l(f)$. The correction term $l(f)$ in the denominator is needed to eliminate the linear bias towards longer fibers introduced by the tractography algorithm.

Chapter 4

Survivability in IP/WDM Networks

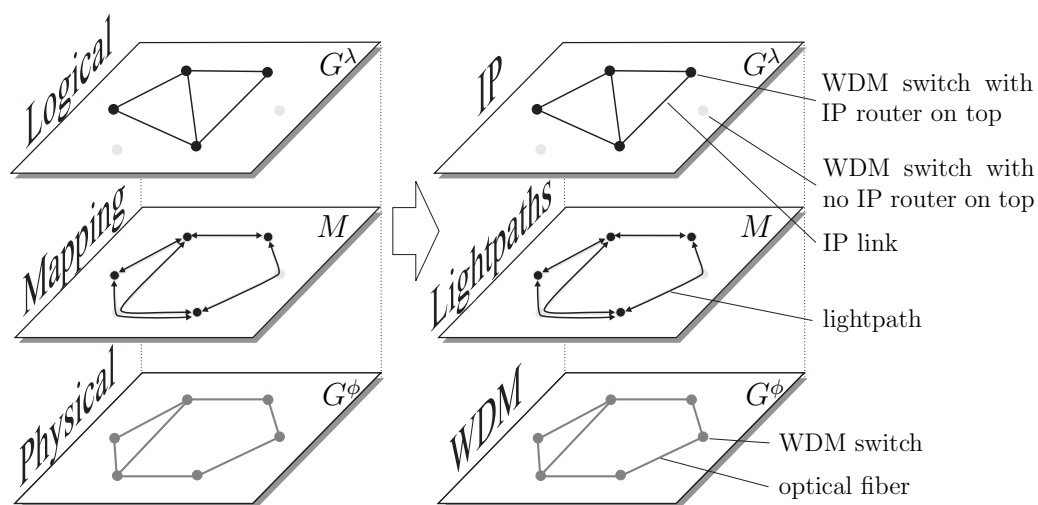


Figure 4.1: An illustration of two layers in the IP/WDM setting.

The study of existing large-scale systems in Chapter 3 confirms that two-layer systems are much more vulnerable to errors and intentional attacks than they appear from a single layer perspective. This raises an obvious question: Can we increase the robustness of a two-layer system by an appropriate *design*? In this chapter we address this question in the context of IP/WDM networks.

4.1 Introduction

An IP-over-fiber network is a typical building block of the Internet's backbone. It usually belongs to a single Internet Service Provider (ISP) and is centrally monitored and managed. The physical infrastructure of an IP-over-fiber network consists of a mesh of optical fibers usually put in the ground along roads, rails, or power-lines. Currently, with the help of the Wavelength Division Multiplexing (WDM) technique, a single optical fiber can carry many signals independently, each using a different wavelength (color). The IP links are realized as end-to-end connections routed on this physical mesh. The topology formed by the IP links is a result of a centralized optimization process and reflects the user demands. This stack is called an IP-over-WDM network, or shortly 'IP/WDM' for short.

Traffic engineering in such a two-layer system is an important yet challenging task [46]. Indeed, as under the WDM technology every fiber potentially carries many signals, a single failure of a physical fiber might have very significant consequences on the network and should be carefully handled. This can be achieved by an appropriate design of the IP/WDM network, which is the topic of this chapter. In particular, we are interested in finding a *survivable mapping*, where the logical graph remains connected after any single physical link failure. We introduce a novel approach to the problem and prove some of its properties. This leads us to an algorithm called SMART. Its three major applications are: (i) formal verification of the existence of a survivable mapping, (ii) a tool tracing and repairing vulnerable areas in the system, and (iii) efficient and scalable heuristic that finds a survivable mapping.

Organization of this Chapter

We proceed as follows. First in Sections 4.2 and 4.3 we present a high-level introduction to the IP/WDM setting and the problems that we address. In particular, in Section 4.2 we describe the two layers, mapping, types of failures and failure protection mechanisms in the IP/WDM context. Next, in Section 4.3 we overview our objectives, contributions and related work.

In the remaining sections we rigorously address the specific problem, as follows. In Section 4.4, we introduce the additional notation and formalize the basic version of the problem. In Section 4.5, we give three core theorems that SMART builds on. For better readability all proofs are moved to the Appendix at the end of this chapter. In Section 4.6, we introduce the SMART algorithm and discuss its properties. In Section 4.7, we describe our implementation of SMART. In Section 4.8, we discuss a number of possible

applications of SMART. In Section 4.9, we present the simulation results. Next, we extend the analysis and simulation results to more elaborate types of failures (Sections 4.10-4.12) and to capacity constraints (Section 4.13). Finally, in Section 4.14 we conclude the chapter.

4.2 IP/WDM as a Two-Layer System

In this section we describe how the IP/WDM system corresponds to the general two-layer setting introduced in Chapter 2. We consider all aspects, from the specification of the two layers and their mapping, to common failures and typical ways of handling them.

4.2.1 Physical Layer, Logical Layer and the Mapping in IP/WDM Networks

In what follows, please refer to Fig. 4.1.

Physical Layer G^ϕ

The *physical layer* $G^\phi = (V^\phi, E^\phi)$ is a set of optical switches (nodes) and optical fibers (links) interconnecting them.

The simplest type of physical node is an optical crossconnect (OXC). It switches the optical signal from an input port to the output port with no wavelength conversion, leading to the *wavelength continuity constraint* [47]. Physical nodes can be also equipped with *wavelength converters* to alleviate this constraint: some can offer a full conversion capability (that is, any wavelength can be converted to any other wavelength), others only offer limited conversion capability (that is, a wavelength can only be converted to the neighboring wavelengths on the spectrum). In this chapter, we assume that the full wavelength conversion at every node is available.

In our basic analysis, we also assume infinite capacities of optical fibers, meaning unlimited number of wavelengths per fiber. We introduce the capacity constraints to the analysis in Section 4.13.

The physical network G^ϕ is composed of the existing infrastructure and is thus considered given and fixed.

Logical Layer G^λ

The *logical layer* $G^\lambda = (V^\lambda, E^\lambda)$ consists of a set of IP routers (nodes) spanned by IP links. Not every physical node has an IP routing capability, which implies $V^\lambda \subseteq V^\phi$. All the results in this chapter hold for $V^\lambda \subseteq V^\phi$,

however, for the sake of simplicity we have chosen to keep V^ϕ and V^λ identical, i.e., $V^\phi \equiv V^\lambda \equiv V$.

The logical graph is closely related to the external traffic demands and thus is considered given and fixed.

Mapping M

Each logical link $e^\lambda \in E^\lambda$ is mapped on the physical topology G^ϕ as a physical path $M(e^\lambda)$, which is called a *lightpath* in the IP/WDM setting. The mapping is defined only for the existing logical links E^λ , thus we denote it by M_{E^λ} .

Recall that in an IP/WDM system, the physical and logical layers are fixed. However, we are free to design the mapping M_{E^λ} .

4.2.2 Failures in IP/WDM Networks

Here we overview the typical failures that may be encountered in IP/WDM networks. The approach we introduce later in this chapter is suitable to deal with each of them.

Single-Link Failures

The most common is a *single physical link failure*. This can be caused by a fiber (physical link) cut, a fault of a single interface card in an optical switch, or a fault of an optical amplifier.

Single physical link failure is the basic error type addressed in this chapter; all the remaining types (see below) are considered only in Sections 4.10-4.12.

Span Failures

If we allow for the physical location of the fibers, we will extend single link failures to single *span failures* [48]. A *span* is a collection of all fibers co-located in the same conduit, between two consecutive points of access (such as a manhole or an amplifier site). To limit costs, different physical links are sometimes put in the same span (e.g., along railway and electricity lines); consequently, a single cut can break all of them at once. So a span failure can be either a failure of a single physical link or simultaneous failure of all physical links that are put in the same span.

Node Failures

We can also encounter *node failures*; they are the consequence of a failure of equipment at nodes, such as switches.

Double-Link Failures

Finally, we consider the *double-link failures*, i.e., independent failures of any two physical links. Usually such a situation takes place when the second failure occurs before the first one is repaired. This is not very common, but possible. For example, in the Sprint network, the time between two successive optical failures ranges from 5.5 sec to 7.5 days with a mean of 12 hours [49]. Most of them are repaired automatically within several minutes, but those requiring human intervention (e.g., after a fiber cut) may last hours or days. It is quite probable that during this period another physical failure occurs.

4.2.3 Failure Protection Mechanisms in IP/WDM Networks

In Section 2.3, we reviewed the failure protection techniques in two-layer systems. IP/WDM networks employ both the *proactive* and the *reactive* approaches [50]. Following the commonly used terminology, we will refer to them as ‘protection’ and ‘restoration’, respectively. Protection uses pre-computed backup paths applied in the case of a failure. Restoration finds dynamically a new path, once a failure has occurred. Protection is less resource efficient (the resources are committed without prior knowledge of the next failure) but fast, whereas restoration is more resource efficient and slow.

Protection and restoration mechanisms can be provided at different layers. *IP layer* (i.e., logical layer) survivability mechanisms can handle failures that occur at both layers, contrary to *WDM layer* (i.e., physical layer) mechanisms that are transparent to the IP topology. In practice, we usually speak of either the protection at the physical layer (‘WDM protection’), or restoration at the logical layer (‘IP restoration’).

Under WDM protection, the failures are detected at the physical layer and the traffic is automatically switched to pre-computed backup paths.

In contrast, under IP restoration, a physical failure propagates to the logical layer (possibly as multiple logical failures), where it is detected; next the alternative routes in the logical topology are found. We review these techniques in more details in [51].

4.3 Our Objectives and Achievements

4.3.1 We Address the IP Restoration Problem

It is not obvious which failure management technique (WDM protection or IP restoration) is better; each has pros and cons [50,52]. IP restoration is more resource efficient. Unfortunately it is also inherently slow due to the long failure detection time at the IP layer. Still, many real network operators choose to deploy only IP restoration and find it an effective and cost-efficient solution (see e.g., Sprint network [53]). Moreover, IP restoration captures directly the interaction between two-layers, which is the main object of study in this thesis. For these reasons, we address the IP restoration problem in this chapter.

4.3.2 Vulnerability \mathcal{W} and Survivability

In order to enable IP restoration, two requirements must be met. First, the links cannot be fully utilized at the IP layer, in order to be able to absorb the additional traffic rerouted after a failure. This is called overprovisioning. Second, a single physical failure cannot cut the connectivity at the IP layer. This may be guaranteed by an appropriate mapping of logical links on the physical topology. Finding such a mapping is our main goal.

More precisely, following [54] we say that a tuple $[e^\lambda, e^\phi]$ is a *broken pair*, if a single failure of e^ϕ partitions the logical graph G^λ in such a way that the end-nodes of e^λ belong to two different components of G^λ . Now, we define our basic Vulnerability metric \mathcal{W} (see Section 2.4) as the number of broken pairs in the system.

In particular, we are mainly concerned about two cases. If $\mathcal{W} = 0$, i.e., if the logical topology remains connected after any single physical link failure, then the underlying mapping is called a *survivable mapping* (see Fig. 4.2a). Otherwise, if $\mathcal{W} > 0$, then the mapping is not survivable, as presented in Fig. 4.2b.

To summarize, given a fixed logical graph G^λ and physical graph G^ϕ , our goal is to find a survivable mapping M .

4.3.3 Related Work

The problem of survivable mapping is NP-complete [55], and has drawn much attention. It was first identified by Crochat et.al. [54], and named “design protection”. Since then, many algorithms solving this problem (with different variations) have been proposed. In general, the previous approaches

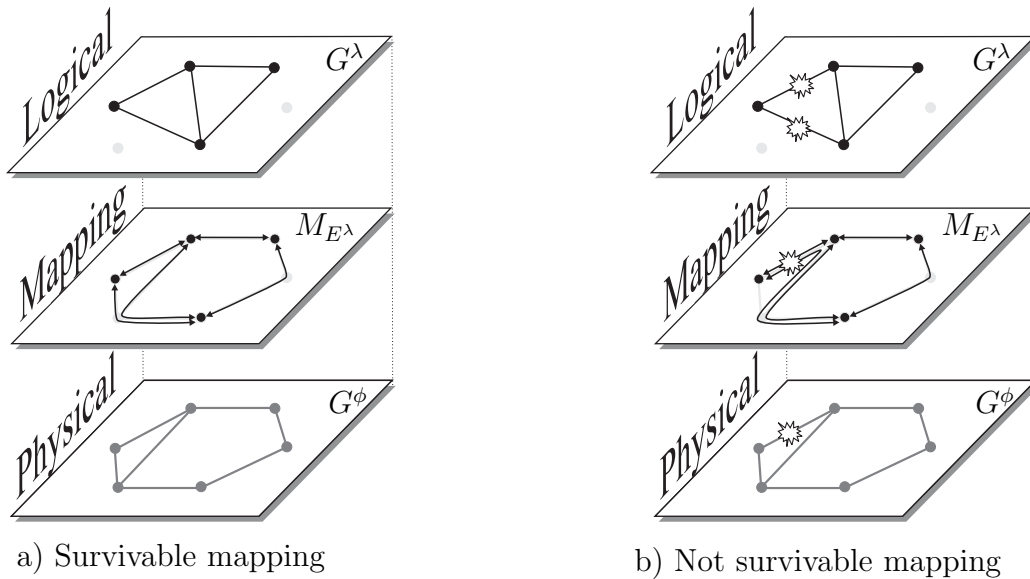


Figure 4.2: (a) Survivable mapping: the logical topology G^λ remains connected after any single physical link failure. (b) Not survivable mapping: G^λ gets partitioned after the single physical failure shown in the figure.

can be divided into two groups: (i) exact algorithms based on Integer Linear Programming (ILP) and (ii) pure heuristics.

ILP The ILP solutions can be found, for example, in [50,55,56]. The authors of [55] observe that a mapping is survivable if and only if no physical link is shared by all logical links belonging to a cut-set of the logical graph.¹ This observation is used in [55] to formulate an ILP model for the survivable mapping problem: For each logical link and for each cut-set of the logical graph, a constraint is added to the ILP. This leads to exact solutions, but also to excessive run-times ([57]) for networks of a non-trivially small size (few tens of nodes). To overcome this difficulty the authors of [55] propose two relaxations to their ILP, in which they include only cut-sets of small sizes. This considerably speeds up the algorithm, but can easily lead to suboptimal solutions. Facing the same time-complexity problem of ILP, the authors of [50] and [56] try a heuristic approach.

¹A *cut-set* of a network is defined by a cut of the network: A cut is a partition of the set of nodes V into two sets S and $V - S$, and the cut-set defined by this cut is the set of edges which have one endpoint in S and one in $V - S$.

Heuristics Despite many differences, the heuristics used to solve the survivable mapping problem share the same general methodology. They start with some initial mapping (e.g., shortest path) and try to improve it at subsequent iterations. Probably the most often used heuristic is *Tabu Search*. It is a version of a steepest descent search algorithm which stores a list (called a Tabu List) of recent moves in order to avoid them. This allows Tabu Search to escape the local minima. For more details refer to [58]. Tabu Search is used with success to solve the survivability problem in many settings, e.g., without capacity constraints [54], with capacity constraints [59,60] or additionally meeting maximum delay requirements [56]. Another general heuristic applied to solve the survivable mapping problem is Simulated Annealing in [52]. There is also a number of heuristics developed specifically to solve this problem, e.g., in [50] and [61]. The FastSurv algorithm introduced recently in [61], exploits the observation already mentioned in the ILP paragraph, which takes use of cut-sets in the logical topology. However, unlike in [55], the FastSurv algorithm systematically and indirectly learns about the importance of particular cut-sets and focuses only on the most relevant ones. This approach results in efficiency and scalability much better than those of other heuristics.

4.3.4 Our Contributions

We propose a new approach to the problem of survivable mapping, called ‘SMART’ (“Survivable Mapping Algorithm by Ring Trimming” [6,7]). SMART does not fall in any of the two groups above. It is based on a breakdown of the problem into a set of independent smaller problems that are easy to solve. Each of them is solved separately, and then the solutions are combined to obtain a survivable mapping of the entire topology. This novel approach not only leads to the fastest and scalable heuristic to date, but also exhibits a number of provable and useful properties. As a result, we propose *three important applications* of SMART, as follows.

Application 1: Formal verification of the existence of a survivable mapping. First, we are interested in the *existence* of a survivable mapping for a given pair of logical and physical topologies. There is some work on the topic in the literature, but it assumes *ring topologies* at the physical [62,63] or the logical [55,64] layer. We study the existence of a survivable mapping for general mesh topologies at both layers, which is foreseen to be the main future topology. To date, the only general method verifying the existence of a survivable mapping is an exhaustive search (or equivalent) run for the *entire* topology. Due to NP-completeness of the survivable mapping problem [55],

the exhaustive approach is not realizable in practice for the topologies larger than a few nodes. To bypass this difficulty, we introduce a new type of mapping that preserves the survivability of some subgraphs ('pieces') of the logical topology; we call it a *piecewise survivable mapping*. The formal analysis of the piecewise survivable mapping shows that a survivable mapping of the logical topology on the physical topology exists if and only if there exists a survivable mapping for a *contracted* logical topology, that is, a logical topology where a specified subset of edges is contracted (contraction of an edge amounts to removing it and merging its end-nodes). This new result substantially simplifies the verification of the existence of a survivable mapping. Of course, the problem remains NP-complete, but this simplification allows us, for the first time, to solve many instances of moderate and large topology size (say, 15 or more nodes), which makes it applicable in practice.

Application 2: A tool to trace and repair the vulnerable areas of the network. A second application of a piecewise survivable mapping is tracing the vulnerable areas in the network and pointing where new link(s) should be added to enable a survivable mapping. To the best of our knowledge, this is also a novel functionality.

Application 3: A fast heuristic Finally, SMART finds a survivable mapping much faster (often by orders of magnitude) than the ILP and heuristic techniques proposed to date.

4.3.5 We use the general two-layer model terminology

In Chapter 2 we defined the general components of the two-layer system. In Section 4.2 we described how the IP/WDM network fits in the two-layer model. In the remainder of this chapter we obtain results that, although inspired by IP/WDM, are rather general and not necessarily restricted to optical networks. Therefore, hereafter we intentionally drop the IP/WDM-specific terminology and use only the terms directly related to the two-layer model.

4.4 Additional Notation

In this section we introduce a number of definitions that are essential to the approach we propose.

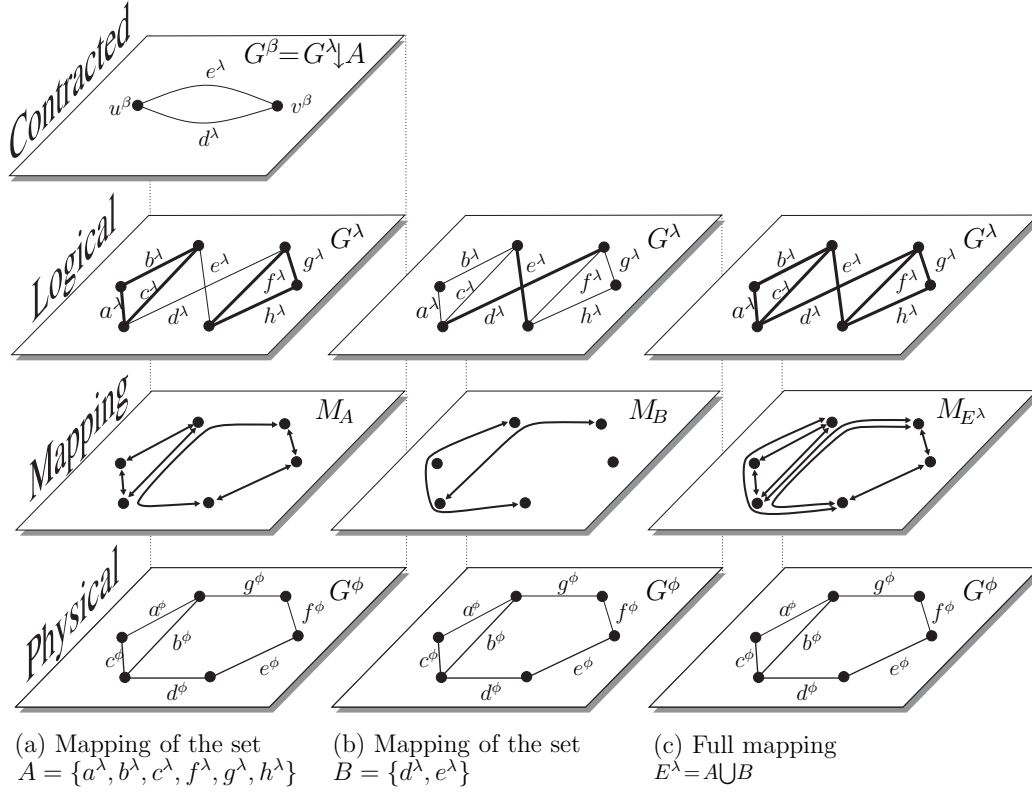


Figure 4.3: Three mapping examples. We have four layers, from bottom to top: the physical topology G^ϕ , the mapping M , the logical topology G^λ and the contracted logical topology G^β (only in (a)). In (a) the pairs $[G^\lambda_{\{a^\lambda, b^\lambda, c^\lambda\}}, M_A]$ and $[G^\lambda_{\{f^\lambda, g^\lambda, h^\lambda\}}, M_A]$ are survivable, and therefore the pair $[G^\lambda, M_A]$ is piecewise survivable. In (b) the mapping M_B maps edge-disjointly the set $B = \{d^\lambda, e^\lambda\}$ of two logical links. The contracted topology G^β in (a) is composed of these two links. Taking G^β and M_B together, we obtain the pair $[G^\beta, M_B]$, which is survivable. In (c) the pair $[G^\lambda, M_{E^\lambda}]$ is survivable, that is M_{E^λ} is a survivable mapping of the entire logical topology.

4.4.1 Combining two mappings

We allow putting a set of logical links $A_{sub} \subset A$ as an argument, which results in a set of paths $M_A(A_{sub}) \subset P^\phi$. Similarly, we also define a mapping of a logical path p^λ . If all logical edges in p^λ belong to $A \in E^\lambda$, then $M_A(p^\lambda)$ is the concatenation of mappings of all consecutive pairs of vertices in p^λ . Consequently, $M(p^\lambda)$ is also a path in G^ϕ , but some edges and nodes may be repeated.

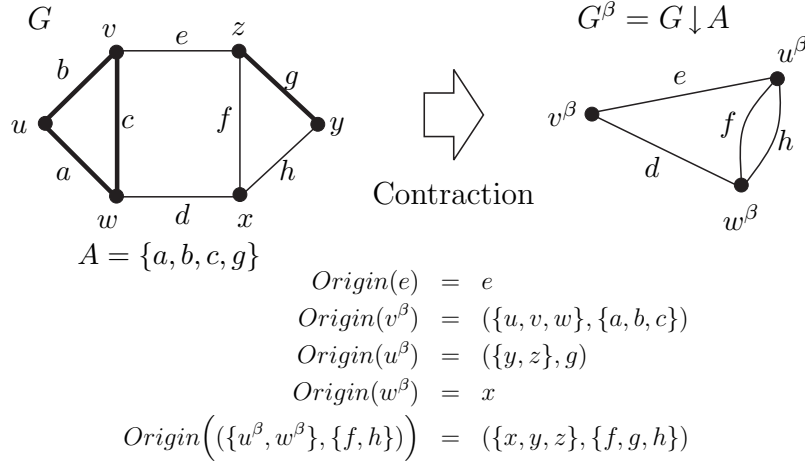


Figure 4.4: Contraction of a graph G on a set of edges $A = \{a, b, c, g\}$. The origins of some elements of $G^\beta = G \downarrow A$ are also shown (bottom).

Example 1 Fig. 4.3 illustrates the definitions given above. In Fig. 4.3a the mapping M_A is defined for the subset A of logical links (marked in bold in the logical topology). For example, we have $M_A(f^\lambda) = \langle d^\phi, b^\phi, g^\phi \rangle$, which means that the lightpath assigned for the logical edge f^λ consists of three physical links. Fig. 4.3b presents a mapping defined for the subset B , whereas the mapping M_{E^λ} in Fig. 4.3c is defined for all links of the logical topology $E^\lambda = A \cup B$.

We will often deal with mappings of different subsets of logical edges. Let $A_1, A_2 \subset E^\lambda$. For consistency, we always require that:

$$\text{for every } e^\lambda \in A_1 \cap A_2 : M_{A_1}(e^\lambda) = M_{A_2}(e^\lambda). \quad (4.1)$$

The mappings M_{A_1} and M_{A_2} can be merged, resulting in a mapping M_{A_3} defined as follows

$$A_3 = A_1 \cup A_2 \quad (4.2)$$

$$M_{A_3}(A_3) = M_{A_1}(A_1) \cup M_{A_2}(A_2). \quad (4.3)$$

For convenience of notation, we will write (4.2) and (4.3) as $M_{A_3} = M_{A_1} \cup M_{A_2}$.

4.4.2 Contraction and Origin

In the chapter we will often use the graph operator of *contraction*, which is illustrated in Fig. 4.4 and is defined as follows:

Definition 1 (Contraction [65]) Contracting an edge $e \in E$ of a graph $G = (V, E)$ consists in deleting that edge and merging its end-nodes into a single node. The result is called the contraction of a graph G on an edge e (or simply a contracted graph), and is denoted by $G^\beta = G \downarrow e$.

By extension, we also allow contracting a set of edges $A \subset E$, resulting in a contracted graph $G^\beta = G \downarrow A$, obtained by successively contracting the graph G on every edge of A . It is easy to show that the order in which the edges of A are taken to contraction, does not affect the final result.

Let $G = (V, E)$, $A \subset E$ and $G^\beta = (V^\beta, E^\beta) = G \downarrow A$. Note that by construction $E^\beta = E \setminus A$. Therefore each edge of G^β can be found in G , as depicted in Fig. 4.4. This is not always true for vertices. A vertex of V^β may either ‘originate’ from a single vertex in G (like w^β in Fig. 4.4), or from a connected subgraph of G (like v^β and u^β). We call this relation an *Origin*(\cdot).

Definition 2 (Origin) Let $G^\beta = G \downarrow A$. Now take a subgraph $G_{sub}^\beta \subseteq G^\beta$. We say that $G_{sub} = \text{Origin}(G_{sub}^\beta)$, if G_{sub} is the maximal subgraph of G that was transformed into G_{sub}^β by the contraction of A in G .

According to this definition, the result of the *Origin*(\cdot) function is the *maximal* subgraph transformed in its argument. For example, we could say that in Fig. 4.4, the vertex $z \in G$ was transformed into the vertex $u^\beta \in G^\beta$, however $z \neq \text{Origin}(u^\beta)$ because it is not the only element that was transformed into u^β by contraction. The maximal subgraph in this case is $(\{y, z\}, g) = \text{Origin}(u^\beta)$.

4.4.3 Survivability and Piecewise Survivability

Let M_{E^λ} be a mapping of the logical topology G^λ on the physical topology G^ϕ . Assume that a physical link e^ϕ fails. Each logical link in G^λ using e^ϕ in its mapping (lightpath) will then be cut. This may cause a disconnection of G^λ . If, after any single physical link failure, the graph G^λ remains connected, then the pair $[G^\lambda, M_{E^\lambda}]$ is declared *survivable*. We extend this property to a family of graphs constructed from the logical topology in the following definition:

Definition 3 (Survivability) Let $G^\lambda = (V, E^\lambda)$, $A \subset E^\lambda$ and $G^\beta = (V^\beta, E^\beta) = G^\lambda \downarrow A$. Take any connected subgraph $G_{sub}^\beta = (V_{sub}^\beta, B)$ of the contracted topology G^β , and let M_B be a mapping of the set B of logical links. The pair $[G_{sub}^\beta, M_B]$ is survivable if the failure of any single physical link e^ϕ does not disconnect the graph G_{sub}^β .

A direct consequence of Definition 3 is that if $[G_{sub}^\beta, M_B]$ is survivable, then $[G_{sub}^\beta, M_{B'}]$ is also survivable, for any $B \subset B' \subseteq E^\lambda$.

In Definition 3, G_{sub}^β represents a large family of graphs obtained from the logical topology. If $A = \emptyset$, then $G^\beta = G^\lambda$ and G_{sub}^β is any connected subgraph of G^λ (including G^λ itself). If $A \neq \emptyset$, then G_{sub}^β is any connected subgraph of $G^\lambda \downarrow A$. The different instances of G_{sub}^β and survivable pairs are given in Fig. 4.3 and described in the following three examples:

Example 2 *It is easy to check that in Fig. 4.3c the pair $[G^\lambda, M_{E^\lambda}]$ is survivable.*

Example 3 *In Fig. 4.3a, let $G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda$ be the subgraph of G^λ defined by the edges $a^\lambda, b^\lambda, c^\lambda$ and their end-vertices. The pair $[G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda, M_A]$ is survivable, because the failure of any physical link does not disconnect $G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda$. Similarly, the pair $[G_{\{f^\lambda, g^\lambda, h^\lambda\}}^\lambda, M_A]$ is also survivable.*

Example 4 *In Fig. 4.3a, the contracted topology G^β is the result of the contraction of the logical topology on the set A , i.e., $G^\beta = G^\lambda \downarrow A$. Take $G_{sub}^\beta = G^\beta$. It consists of two logical links, d^λ and e^λ . A possible mapping of the set $B = \{d^\lambda, e^\lambda\}$ is the mapping M_B shown in Fig 4.3b. Consider the pair $[G^\beta, M_B]$; it is survivable, because a single physical link failure cannot bring down both d^λ and e^λ at the same time, hence G^β remains connected.*

Definition 4 (Piecewise Survivability) *Let M_A be a mapping of a set $A \subset E^\lambda$ on the physical topology. The pair $[G^\lambda, M_A]$ is piecewise survivable if, for every vertex v^β of the contracted logical topology $G^\lambda \downarrow A$, the pair $[Origin(v^\beta), M_A]$ is survivable.*

In contrast to survivability, piecewise survivability is defined only for the entire logical topology G^λ . We will say that a mapping M_A is (piecewise) survivable, if the pair $[G^\lambda, M_A]$ is (piecewise) survivable (i.e., we take G^λ as the default topology).

Example 5 *In Fig. 4.3a, the pair $[G^\lambda, M_A]$ is piecewise survivable. To prove it, we have to show that for vertices u^β and v^β of $G^\lambda \downarrow A$, the pairs $[Origin(u^\beta), M_A]$ and $[Origin(v^\beta), M_A]$ are survivable. Here we have $Origin(u^\beta) = G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda$ and $Origin(v^\beta) = G_{\{f^\lambda, g^\lambda, h^\lambda\}}^\lambda$. We have shown in Example 3 that each of these two graphs forms a survivable pair with M_A .*

4.5 Fundamental properties of survivable and piecewise survivable mappings

In this section we prove three general properties of survivable and piecewise survivable mappings. They hold independently of what is fixed in the two-layer structure. We will often use these results in the following sections.

4.5.1 The Expansion of Survivability

Given a piecewise survivable mapping, the logical topology can be viewed as a set of survivable ‘pieces’. This is a general property of a piecewise survivable mapping. (For instance in Example 5, given the piecewise survivable mapping M_A , there are two survivable ‘pieces’ of G^λ : $G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda \subset G^\lambda$ and $G_{\{f^\lambda, g^\lambda, h^\lambda\}}^\lambda \subset G^\lambda$.) The following theorem enables us to merge some of these pieces, resulting in a single large survivable piece.

Theorem 1 (Expansion of Survivability)

Let M_A be a mapping of a set of logical edges $A \subset E^\lambda$ on the physical topology G^ϕ , such that the pair $[G^\lambda, M_A]$ is piecewise survivable. Let $G^\beta = G^\lambda \downarrow A$. Take any subgraph of G^β , call it $G_{sub}^\beta = (V_{sub}^\beta, B)$. Let M_B be a mapping of the set B of edges of G_{sub}^β on G^ϕ . If the pair $[G_{sub}^\beta, M_B]$ is survivable then the pair $[Origin(G_{sub}^\beta), M_A \cup M_B]$ is also survivable.

Proof 1 See Appendix.

The following example illustrates this theorem.

Example 6 In Example 5 we have shown that in Fig. 4.3a, the pair $[G^\lambda, M_A]$ is piecewise survivable. Take $G_{sub}^\beta = G^\beta = G^\lambda \downarrow A$ and take M_B as in Fig. 4.3b. From Example 4, we know that the pair $[G^\beta, M_B]$ is survivable. Now, by Theorem 1, the pair $[Origin(G^\beta), M_A \cup M_B] = [G^\lambda, M_A \cup M_B]$ is survivable. So beginning from the piecewise survivable mapping M_A and adding the mapping M_B , we merged the two survivable pieces $G_{\{a^\lambda, b^\lambda, c^\lambda\}}^\lambda$ and $G_{\{f^\lambda, g^\lambda, h^\lambda\}}^\lambda$ into a single, large, survivable piece. In this example the resulting survivable piece is the entire logical topology G^λ . The full mapping $M_A \cup M_B = M_{E^\lambda}$ is shown in Fig. 4.3c.

4.5.2 Invariance of Survivability Under Contraction

Theorem 2 (Invariance of Survivability Under Contraction)

Let $G_{sub}^\beta = (V_{sub}^\beta, B)$ be a subgraph of some contracted topology G^β . If M_B is a mapping such that the pair $[G_{sub}^\beta, M_B]$ is survivable, then for any set $A \subset B$ of logical links the pair $[G_{sub}^\beta \downarrow A, M_B]$ is also survivable.

Proof 2 See Appendix.

In other words, Theorem 2 says that if we can map in a survivable way some subgraph G_{sub}^β of the logical or contracted logical topology, then the subgraph obtained by contracting some additional set A of edges can always be mapped in a survivable way, whatever the choice of A .

Example 7 Take $G_{sub}^\beta = G^\lambda$ and $M_B = M_{E^\lambda}$ as in Fig. 4.3c. We know that the pair $[G^\lambda, M_{E^\lambda}]$ is survivable. Theorem 2 implies that for any set of logical edges $A \subset E^\lambda$ the pair $[G^\lambda \downarrow A, M_{E^\lambda}]$ is also survivable. In particular, for the set A as defined in Fig. 4.3a, $[G^\lambda \downarrow A, M_{E^\lambda}]$ is survivable, which was shown in Example 4 ($M_B \subset M_{E^\lambda}$).

Note that we do not impose any requirements (such as e.g., preserving piecewise survivability) on the contracted edges A . Moreover, we do not have any restrictions on what happens with the rest of the contracted topology, i.e., in $G^\beta \setminus G_{sub}^\beta$.

4.5.3 The Existence of a Survivable Mapping

In general, for a given pair of physical and logical topologies, it is very difficult to verify the existence of a survivable mapping. A heuristic approach, if it fails, does not give any answer. The ILP approach or an exhaustive search could provide us with the answer, but due to their high computational complexity their application is limited to the topologies of several nodes. The following theorem shows how this verification problem can be substantially reduced:

Theorem 3 (Existence of a survivable mapping)

Let M_A be a mapping of a set of logical edges $A \subset E^\lambda$, such that the pair $[G^\lambda, M_A]$ is piecewise survivable. A survivable mapping $M_{E^\lambda}^{surv}$ of G^λ on G^ϕ exists if and only if there exists a mapping $M_{E^\lambda \setminus A}^{surv}$ of the set of logical links $E^\lambda \setminus A$ on G^ϕ , such that the pair $[G^\lambda \downarrow A, M_{E^\lambda \setminus A}^{surv}]$ is survivable.

Proof 3 See Appendix. The proof uses Theorems 1 and 2.

The following example illustrates this theorem.

Example 8 In Fig. 4.3 delete edge b^ϕ from the physical topology G^ϕ . Now, for the logical topology G^λ and the physical topology $G^\phi \setminus \{b^\phi\}$, a survivable mapping does not exist. To prove it, note that we can still find a mapping M_A of G^λ on $G^\phi \setminus \{b^\phi\}$ that is piecewise survivable. For instance, we can take M_A defined as follows: $a^\lambda \mapsto (c^\phi)$, $b^\lambda \mapsto (a^\phi)$, $c^\lambda \mapsto (d^\phi, e^\phi, f^\phi, g^\phi)$, $f^\lambda \mapsto (d^\phi, c^\phi, a^\phi, g^\phi)$, $g^\lambda \mapsto (f^\phi)$ and $h^\lambda \mapsto (e^\phi)$. However, the remaining two logical links d^λ and e^λ cannot be mapped edge-disjointly on $G^\phi \setminus \{b^\phi\}$. Therefore no survivable mapping $M_{\{d^\lambda, e^\lambda\}}$ of the contracted logical topology $G^\lambda \downarrow A$ on $G^\phi \setminus \{b^\phi\}$ exists. Consequently, by Theorem 3 we know that no survivable mapping of G^λ on $G^\phi \setminus \{b^\phi\}$ exists, which was to be proved. Note that to prove it, we only considered the two-edge topology $G^\lambda \downarrow A$ instead of the entire G^λ , which greatly simplified the problem. Clearly, the larger the set A , the more we benefit from Theorem 3.

4.6 The SMART algorithm

In this section we present an algorithm that searches for a survivable mapping, which we call SMART. It maps the topology part by part, gradually converging to a final solution. By formal graph theoretic analysis, we prove that if SMART converges completely, a *survivable* mapping is found. Otherwise, when the algorithm terminates before its complete convergence, the returned mapping is *piecewise survivable* and no survivable solution exists.

4.6.1 The pseudo-code of SMART

SMART:

-
1. $G^\beta \leftarrow G^\lambda$, $M_A = \emptyset$, $A = \emptyset$
 2. Take some subgraph $G_{sub}^\beta = (V_{sub}^\beta, B)$ of G^β and find a mapping M_B , such that the pair $[G_{sub}^\beta, M_B]$ is survivable. **if** no such pair exists, **then return** M_A **and** $G^\beta = G^\lambda \downarrow A$, **end**.
 3. Update the mapping by merging M_A and M_B , i.e., $M_A \leftarrow M_A \cup M_B$.
 4. Contract G^β on B , i.e., $G^\beta \leftarrow G^\beta \downarrow B$.
 5. **if** G^β is a single node, **then return** M_A , **end**.
 6. **goto** Step 2.
-

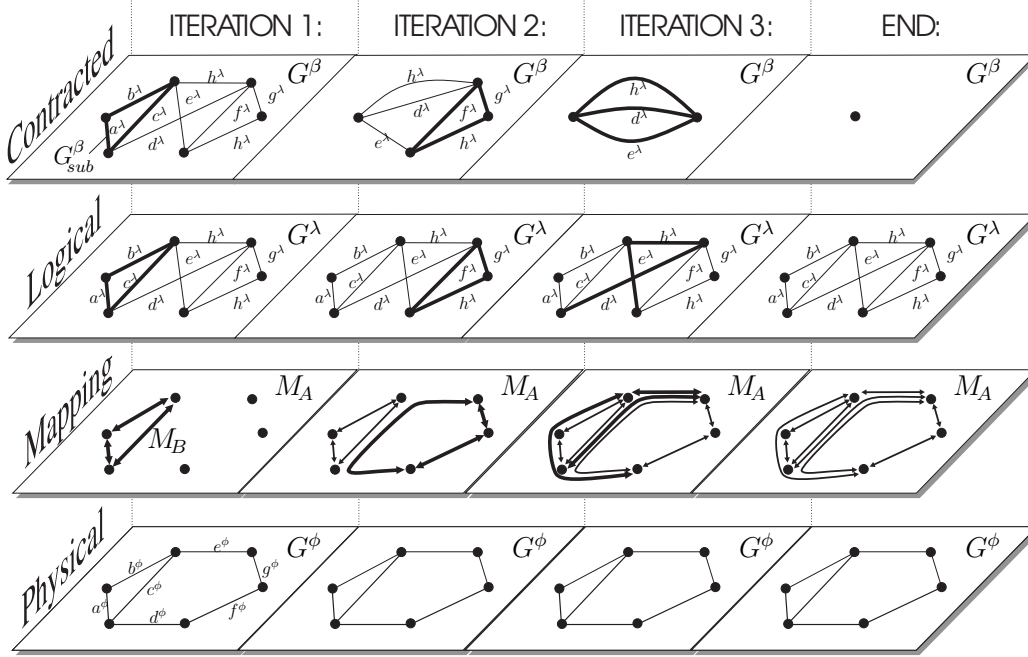


Figure 4.5: Illustration of the SMART algorithm (4.6.1). During a run of the SMART algorithm only the contracted topology and the mapping change from one iteration to the next one. The logical and physical topologies are included only for the context.

4.6.2 SMART illustration

An illustration of a run of SMART is presented in Fig. 4.5. Initially, the mapping M_A is empty, and the contracted topology G^β is identical with the logical topology G^λ . In Step 2 of Iteration 1, SMART picks in G^β a subgraph G_{sub}^β - the triangle set in bold. The logical edges composing it are $B = \{a^\lambda, b^\lambda, c^\lambda\}$. Next, SMART finds a mapping M_B of edges in B such that the pair $[G_{sub}^\beta, M_B]$ is survivable; M_B is set in bold at the mapping layer. In Step 3 this mapping is added to M_A . In Step 4 we contract G_{sub}^β in G^λ , which results in a smaller contracted topology that is used as a starting point of Iteration 2. In this example, after three iterations the graph G^β converges to a single node and SMART terminates in Step 5.

Note that at Iteration 1 and 2 the subgraph G_{sub}^β is a cycle. It requires the mapping M_B to be link-disjoint. In contrast, at Iteration 3 the subgraph G_{sub}^β has a different structure for which a non-link-disjoint mapping M_B is enough to form a survivable pair $[G_{sub}^\beta, M_B]$.

4.6.3 The Correctness of the SMART Algorithm

We declare that *SMART converges* if the contracted topology G^β converges to a *single* node. We prove later in Corollary 1, that the mapping M_A returned in Step 5 is then a survivable solution.

In contrast, we say that *SMART does not converge* if SMART terminates before G^β converges to a single node, i.e., in Step 2. We prove below in Theorem 4 that the mapping M_A returned in Step 2 is piecewise survivable. Moreover, we show in Corollary 1 that in this case a survivable solution does not exist. The graph $G^\beta = G^\lambda \downarrow A$ (also returned in Step 2) is called the *remaining contracted logical topology* as it consists of unmapped logical links $E^\lambda \setminus A$.

Theorem 4 (SMART’s piecewise survivability)

After each iteration of the SMART algorithm, the pair $[G^\lambda, M_A]$ is piecewise survivable.

Proof 4 *See Appendix.*

Theorem 4 leads us to the following important property of the SMART procedure:

Corollary 1 (SMART’s Convergence)

The SMART algorithm returns a single node contracted topology G^β , if and only if there exists a survivable mapping of the logical graph G^λ on the physical graph G^ϕ . In this case the returned mapping M_A is survivable.

Proof 5 *See Appendix.*

G^β may converge to a single node topology with *self-loops*; they form a set of remaining unmapped logical links $E^\lambda \setminus A$. However, this does not affect the result, because the links of $E^\lambda \setminus A$ may be mapped in any way (e.g. shortest path) to obtain a full survivable mapping M_{E^λ} .

4.6.4 The Order of a Sequence of Subgraphs

Recall that in Step 2 of the SMART algorithm we take some subgraph $G_{sub}^\beta = (V_{sub}^\beta, B)$ of the contracted topology G^β . We do not specify which subgraph to take; if there are more candidates G_{sub}^β that meet the condition given in Step 2 (which is usually the case), we are free to choose any of them. This raises a natural question:² How does the choice of G_{sub}^β affect the convergence

²I was asked this question several times when presenting SMART.

of the SMART algorithm? In the following theorem we show that this choice does *not* affect the outcome of the SMART algorithm.

Theorem 5 (SMART’s Uniqueness)

The contracted topology G_{min}^β (excluding self-loops) returned by SMART is unique.

Proof 6 *See Appendix.*

A direct consequence of Theorem 5 is that the order in which we take the subgraphs G_{sub}^β in the SMART algorithm does not affect the final result.

4.7 Implementation - SMART-H

In the previous section we have proposed a general procedure, called SMART, for which we have proved a number of important properties. In practice, however, an exact implementation of SMART (as described by the pseudo-code in Section 4.6.1) is not feasible, because Step 2 is in general an NP-complete problem. For this reason, for large topologies we use a heuristic to solve Step 2. We will refer to the SMART procedure that takes use of any heuristic in Step 2 as “SMART-H.” In this section we first discuss which theoretical results carry over from SMART to SMART-H. Next, we give a simple and effective example of a heuristic for solving Step 2.

4.7.1 Which theoretical results hold for SMART-H?

With a heuristic used to solve Step 2, SMART-H can terminate in Step 2 not only if a pair $[G_{sub}^\beta, M_B]$ does not exist, but also if our heuristic was not able to find one. However, most of the theoretical results obtained for SMART in Section 4.6 carry over to SMART-H and can be successfully applied as we show in Sections 4.8 and 4.9. In particular:

- Theorem 4 holds.
- Corollary 1 holds only in one direction: If SMART-H converges, then the returned mapping is survivable (see Proof of Corollary 1). Otherwise, the returned mapping is piecewise survivable (by Theorem 4), but a survivable solution might still exist. However, the piecewise survivability of the returned mapping can be effectively exploited to verify the existence of a survivable solution, as we demonstrate in Section 4.8.1.

- Theorem 5 does no longer hold. In practice, however, it is highly probable that if our heuristic for Step 2 can find a survivable mapping of some G_{sub}^β then it will also find a survivable mapping of its contracted version $G_{sub}^\beta \downarrow A$. (For instance, it is harder to map disjointly all edges of a cycle, than only a subset of them.) With this property, distinct runs of SMART-H converge to the same contracted topology. We have found an excellent confirmation of this claim in simulations. We have tested many pairs of physical and logical topologies for which a survivable mapping does not exist (so G_{min}^β is always larger than a one-node topology). To ensure a variety of sequences of cycles considered in Step 2 of SMART-H, we randomly permute every time the list of candidates. Even with this approach, for a given pair of topologies, more than 99% of distinct runs of SMART-H resulted in the same contracted topology.

4.7.2 *DisjointMap* - a heuristic for Step 2 of SMART-H

In our implementation of Step 2 of SMART-H we take the graph G_{sub}^β in the form of a *cycle*. Thus we will systematically contract cycles (or ‘rings’) found in the contracted logical topology, which explains the name of the algorithm (“Survivable Mapping Algorithm by Ring Trimming”). G_{sub}^β in the form of a cycle requires the mapping M_B (Step 2) to be *edge-disjoint*. (Otherwise, if the same physical link e^ϕ is used by two or more logical links in G_{sub}^β , a failure of e^ϕ will bring these links down, disconnecting the cycle G_{sub}^β .) Since finding it is equivalent to the NP-complete edge-disjoint paths problem [66], we applied a simple heuristic that we call *DisjointMap*, as follows.

Let $G^\phi = (V, E^\phi)$ be the physical graph, $G^\lambda = (V, E^\lambda)$, and C be a cycle in the contracted graph. Note that the cycle C consists of logical edges only, i.e., $C \subset E^\lambda$. The *DisjointMap* heuristic attempts to map the set of logical edges C on the physical graph in an edge disjoint way. Its pseudocode is as follows:

DisjointMap:

1. **for** every $e^\phi \in E^\phi$ **do** $\text{weight}(e^\phi) \leftarrow I_{small}$
 2. $M \leftarrow \emptyset$
 3. **for** every $e^\lambda \in C$ **do**
 $M(e^\lambda) \leftarrow$ shortest path in G^ϕ from $e_{(1)}^\lambda$ to $e_{(2)}^\lambda$
for every $e^\phi \in M(e^\lambda)$ **do**
 $\text{weight}(e^\phi) \leftarrow \text{weight}(e^\phi) + I_{big}$.
 4. **if** no physical edge is used more than once in M **then**
return . (M is a disjoint mapping)
 5. **for** every occurrence of any physical edge $e^\phi \in M$ **do**
 $\text{weight}(e^\phi) \leftarrow \text{weight}(e^\phi) - I_{big}$ (to restore the state from before step 2)
 6. **for** every $e^\phi \in E^\phi$ that is used more than once in M **do**
 $\text{weight}(e^\phi) \leftarrow \text{weight}(e^\phi) + I_{small}$.
 7. **if** number of iterations > 10 **then**
return . (no mapping found)
 8. **goto** 2
-

In this algorithm, $e_{(1)}^\lambda$ and $e_{(2)}^\lambda$ are the end-nodes of the edge e^λ . The choice of the parameters I_{small} and I_{big} affects the convergence; we used $I_{small} = 1$ and $I_{big} = 1000$.

It is worth noting, that even with a perfect disjoint-map algorithm (such as exhaustive search), its failure for every existing cycle does not imply the non-existence of a survivable mapping. We show an appropriate counterexample in Fig. 4.6. However, in practice this ‘cycle-based’ approach proved to be very efficient.

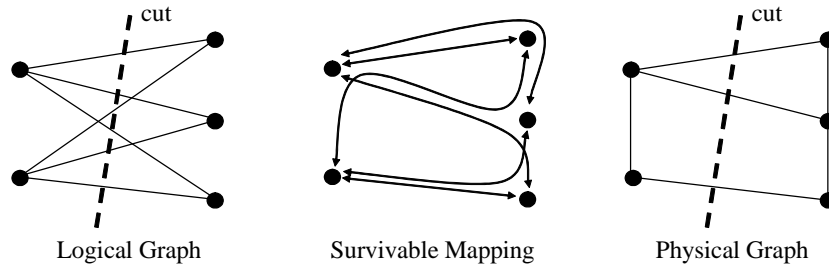


Figure 4.6: An example of topology, for which a survivable mapping exists although it is not possible to disjointly map any of the cycles found in logical topology. The only cycles one can find in the logical topology are three cycles of length four. Since the logical topology is a bipartite graph, each cycle traverses *four* times the cut indicated on the picture. The same cut traverses *three* physical links in the physical topology, which implies that at least one of those physical links must be used twice when mapping a cycle. So it is impossible to disjointly map any cycle; at the same time a survivable mapping exists as shown in the picture in the middle.

4.8 SMART-H Applications

We can apply the SMART-H algorithm in a number of ways. The general scheme can be found in Fig. 4.7. The option we choose depends on the nature of the results we want to obtain. Specifically we can distinguish three types of applications, as follows.

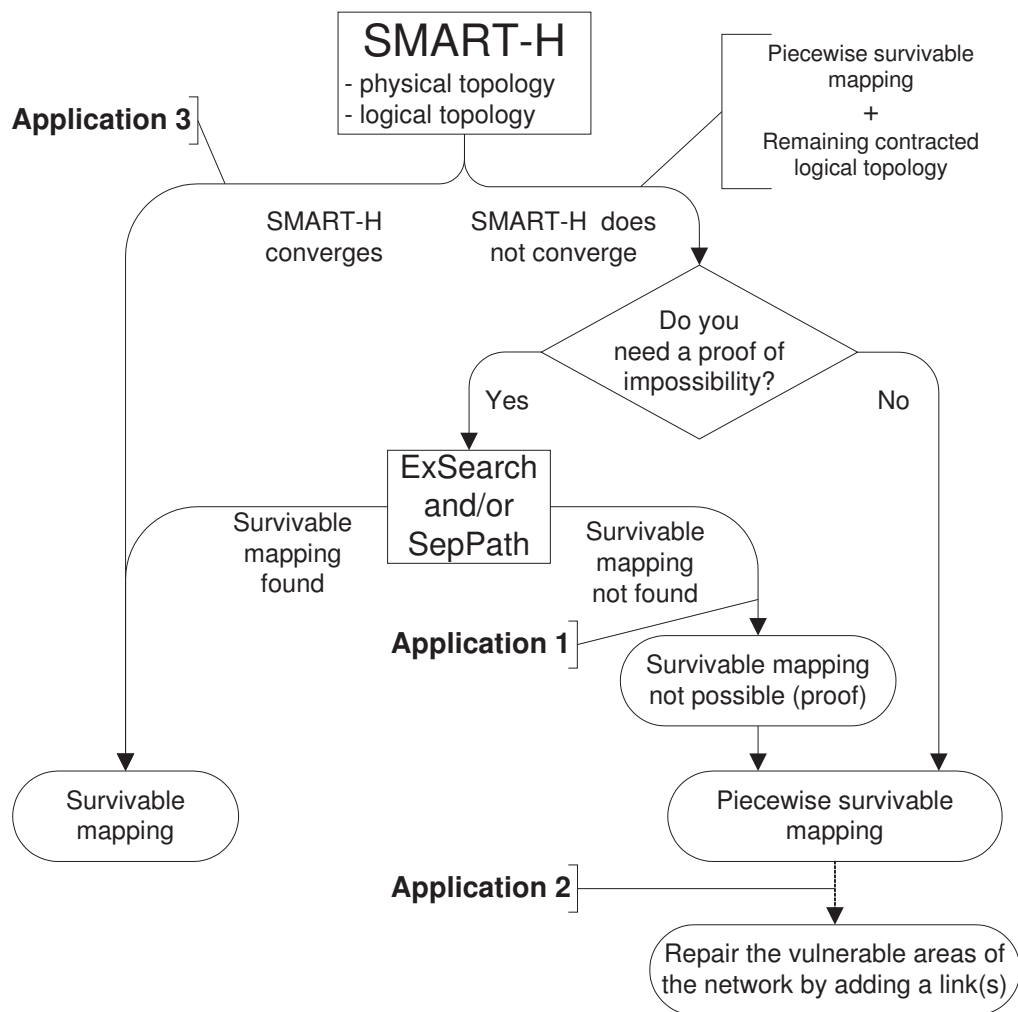


Figure 4.7: Applications of the SMART-H algorithm.

4.8.1 Application1: Formal Verification of the Existence of a Survivable Mapping (ExSearch and SepPath)

Run SMART-H to map a logical topology G^λ on the physical topology G^ϕ . If SMART-H converges, a survivable mapping exists and is returned (see the comment on Corollary 1 in 4.7.1). If SMART-H does not converge, it returns a mapping M_A and a remaining contracted logical topology $G^\lambda \downarrow A$. What makes SMART-H very different from other heuristics is that, by Theorem 4, the returned mapping M_A is piecewise survivable. This allows us to apply Theorem 3, which reduces the task of verifying the existence of a survivable mapping for the entire G^λ , to the same verification for $G^\lambda \downarrow A$. This property is a key feature of SMART-H: if there is a survivable mapping of G^λ on G^ϕ , then SMART-H *will never miss it*, because the set of the remaining logical links $E^\lambda \setminus A$ can be still mapped in a way that preserves the survivability of $G^\lambda \downarrow A$ (and hence of G^λ). In other words, *no backtracking is needed* in order to assess if there exists a survivable mapping for G^λ ; we obtain exactly the same answer by studying $G^\lambda \downarrow A$. Although this verification is NP-complete for both G^λ and $G^\lambda \downarrow A$, in practice the size of $G^\lambda \downarrow A$ is often very small, which makes the verification feasible. We use two methods to verify the existence of a survivable mapping for $G^\lambda \downarrow A$, *ExSearch* and *SepPath*, as follows.

ExSearch uses exhaustive search to find a survivable mapping of the contracted logical topology $G^\lambda \downarrow A$. In order to further reduce the cost of this search we applied Theorem 1 from [55]. It binds the cut-sets in the logical topology G^λ with the mapping M and survivability, which allows us for substantial pruning during the exhaustive search.

SepPath, or ‘Separated Path check’, is defined as follows. If the contracted logical topology $G^\lambda \downarrow A$ contains a path p^β such that all nodes on p^β but the first and the last ones, are of degree two, then clearly all the logical links in p^β must be mapped edge-disjointly to enable survivability. Therefore the failure of an exhaustive search for an edge-disjoint mapping of p^β will prove impossibility. (Otherwise, nothing can be concluded.) Compared to an unrestricted exhaustive search, the exhaustive search respecting the edge-disjointness constraint is relatively easy, although still NP-complete. For this reason SepPath is better suited to larger topologies than ExSearch.

The implementations that first use SMART-H, and then apply ExSearch or SepPath to the returned contracted topology $G^\lambda \downarrow A$, will be called SMART-H + ExSearch and SMART-H + SepPath, respectively. We test the efficiency of these approaches in Section 4.9.2; we apply them to verify the existence of a survivable mapping for various topologies in Section 4.9.3.

4.8.2 Application 2: A Tool Tracing and Repairing the Vulnerable Areas of the Network

We have developed two methods to verify the existence of a survivable mapping: ExSearch and SepPath. Once we know that a particular pair of physical and logical topologies cannot be mapped in a survivable way, a natural question is to modify the topologies to enable such a mapping. Where should a new link be added? The SMART-H algorithm helps us in answering this question. Run SMART-H and wait until it terminates. The remaining contracted logical topology $G^\lambda \downarrow A$ and the piecewise-survivable mapping M_A are returned. Choose at random two nodes u^β, v^β in $G^\lambda \downarrow A$ and pick any two nodes u, v in G^λ , such that $u \in Origin(u^\beta)$ and $v \in Origin(v^\beta)$. Now connect u and v with an additional logical/physical link (remember that we assume identical vertices at both layers). If this link already exists, repeat the procedure.

The simulation results in Section 4.9.4 discuss the efficiency of this approach.

4.8.3 Application 3: A Fast Heuristic

With SMART-H, a survivable mapping is found orders of magnitude more rapidly and usually more often than with other approaches proposed to date. We gave an overview of these techniques in Section 4.3.3, and we compare them with SMART-H in simulation results in Section 4.9.5.

4.9 Simulation Results

In this section we evaluate by simulations the three applications of SMART-H.

4.9.1 Physical and Logical Topologies

In the simulations we use various topologies. A relatively small physical topology is NSFNET (14 vertices, 21 edges) presented in Fig. 4.8a. To imitate larger real-life physical topologies, we also generate square lattices in which a fraction f of edges is deleted, as shown in Fig. 4.8b; we call them f -lattices. The parameter f is often fixed to $f = 0.3$, which resulted in an f -lattice with an average vertex degree slightly smaller than that of NSFNET. As the IP graph is less regular (for instance, there is no reason why it should be planar), the logical topologies are 2-edge-connected random graphs of various average vertex degree. (Clearly, 2-edge-connectivity of both physical

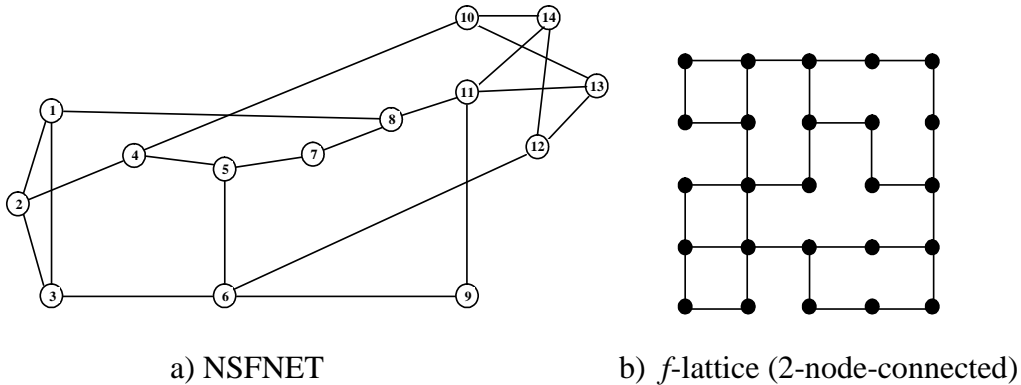


Figure 4.8: Physical topologies used in simulations. (a) NSFNET; (b) f -lattice constructed from full square lattice by deleting fraction f of links, while preserving 2-node-connectivity (here $f \simeq 0.25$).

and logical topologies is a necessary condition for the existence of a survivable mapping.)

4.9.2 ExSearch and SepPath Efficiency, and ‘Unknown Area’

In Section 4.8.1 we defined two methods of verification of the existence of a survivable mapping, ExSearch and SepPath. In this section we examine the benefits of these approaches.

The physical topology is an f -lattice with the parameter $f = 0.3$. The logical topology is a random graph with average vertex degree $\langle k^\lambda \rangle = 4$. For each number of nodes N , we generate a number of physical/logical topology pairs, and keep the first 1000 for which SMART-H *does not converge*. In Fig. 4.9a, we present the cumulative distribution function (CDF) of the number of logical links in the remaining contracted logical topology $G^\lambda \downarrow A$ returned by the algorithm. We can see that, if SMART-H does not converge, the size of $G^\lambda \downarrow A$ is usually relatively small. For instance, for $N = 36$, SMART-H leaves six or fewer logical links out of the total number of 72, in about 80% of cases. Moreover, this property seems to depend only slightly on the topology size.

Now we apply EsSearch and SepPath to the contracted $G^\lambda \downarrow A$ returned by SMART-H. The distribution of run-times of SMART-H + ExSearch is plotted in Fig. 4.9d. For $N = 16$, about 90% of topologies need less than 0.001 sec to run SMART-H + ExSearch.³ Only very few need more than

³We implemented all the algorithms in C++ and ran them on a Pentium 4 machine.

0.1 sec. For comparison purposes, we also ran a complete exhaustive search ExSearch without prior contraction by SMART-H for $N = 16$. We observe the difference in run-times of at least 7 orders of magnitude. Most of the runs of the fully exhaustive search last more than 10000 seconds (~ 3 hours), the maximal time allowed in the simulations. This limits the application of the full exhaustive search to the topologies of a very small size.

1 minute stopping time Fig. 4.9d also exemplifies the tradeoff we faced in simulations. On one hand, the SMART-H + ExSearch runs quickly for the majority of the topologies, but on the other hand, the remaining few topologies take orders of magnitude more time. We observed the same phenomenon when applying the SMART-H + SepPath verification method. Therefore we have decided to use a strict, *one minute stopping time*. If neither SMART-H + ExSearch nor SMART-H + SepPath finishes the computation within 60 seconds, the question of the existence of a survivable mapping is left unanswered. As the result, the figures in the following sections display two curves: the lower one is the percentage of survivable mappings found within 1 minute, the upper one is the percentage of logical topologies proved to be unmappable in a survivable way within 1 minute. The curves are separated by an ‘unknown area’ set in gray.

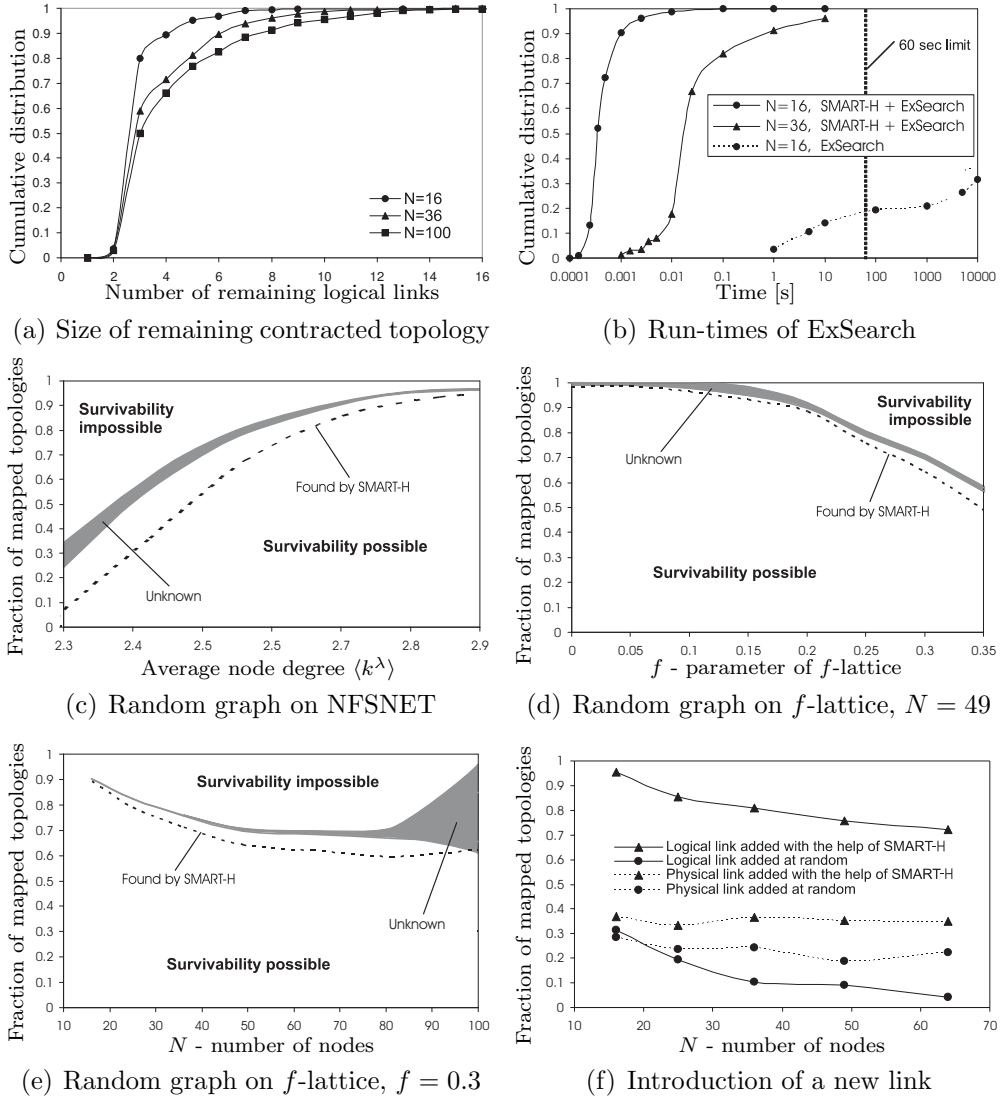


Figure 4.9: Survivability under various scenarios. Parameters: N —number of nodes; f —parameter of the f -lattice physical topology; $\langle k^\lambda \rangle$ —average node degree of the logical topology. (a) CDF of the number of logical links in the remaining contracted logical topology. $f = 0.3$, $N = 16 \dots 100$; (b) CDF of ExSearch times with and without prior contraction by SMART-H. $f = 0.3$, $N = 16 \dots 100$. (c) Random graph logical topologies mapped on NSFNET. $N = 14$, $\langle k^\lambda \rangle = 2.3 \dots 2.9$; (d) Random graph logical topologies mapped on f -lattices. $N = 49$, $f = 0 \dots 0.35$, $\langle k^\lambda \rangle = 4$; (e) Random graph logical topologies mapped on f -lattices. $N = 16 \dots 100$, $f = 0.3$, $\langle k^\lambda \rangle = 4$; (f) Enabling survivability by introducing an additional link. Random graph logical topologies mapped on f -lattices. $N = 16 \dots 100$, $f = 0.3$, $\langle k^\lambda \rangle = 4$.

4.9.3 Results for Application 1

It is interesting to see what fraction of randomly chosen topologies can/cannot be mapped in a survivable way. To the best of our knowledge, this is the first time these results can be obtained in a reasonable time for moderate and large topologies.

For a particular pair of physical and logical topologies, we first apply the SMART-H algorithm. If SMART-H does not converge, we try ExSearch and SepPath to verify the existence of a survivable solution. Their run-times are restricted to the ‘one minute bound’, as explained in Section 4.9.2.

In Fig. 4.9c we present the results of the mapping of random graph logical topologies on NSFNET. We vary the average vertex degree $\langle k^\lambda \rangle$ of the logical graph; for each value of $\langle k^\lambda \rangle$ we generate 1000 topologies. Observe that the results strongly depend on $\langle k^\lambda \rangle$.

In order to examine a larger spectrum of physical topologies and topology sizes, in Figs. 4.9de we map a random graph logical topology on the f -lattice physical topology. This time we fix the average vertex degree of the logical topology $\langle k^\lambda \rangle = 4$ and we vary the parameter f of the physical topology (Fig. 4.9d) or the number of nodes N (Fig. 4.9e). We generated 1000 topologies for each parameter. Fig. 4.9d shows that the fraction of topologies mappable in a survivable way decreases with growing f . This was expected, because it is more difficult to map the logical topology on a sparser physical graph. In Fig. 4.9e, the ‘unknown area’ quickly widens for $N > 80$ because of the ‘one minute bound’.

The dashed curves in Figs. 4.9cde show the fraction of topologies mapped in a survivable way by SMART-H alone, without being followed by ExSearch or SepPath. The distances between these curves and the mapping-impossible areas are relatively small, which confirms the high efficiency of SMART-H as a heuristic.

4.9.4 Results for Application 2

Another property of the SMART-H algorithm is the ability to trace and repair the vulnerable areas of the network. In particular, in Section 4.8.2 we described a way to introduce an additional logical link to enable a survivable mapping. In this section we verify the efficiency of that approach.

We map random graph logical topologies on f -lattices and vary N . For each N , we generate 1000 pairs of physical and logical topologies, such that for each pair separately, a survivable mapping does not exist. For each topology pair, we add one logical or physical link with the help of SMART-H, as described in Section 4.8.2. Next, the existence of a survivable mapping is

verified again, for this extended pair of topologies. For comparison purposes we also simulate a completely random placement of an additional link.

The results are shown in Fig. 4.9f. For better readability, we do not include the ‘unknown area’, which lies above each curve. The application of SMART-H enables a very efficient placement of an additional *logical* link, which helps in 70% to 95% of cases (depending on N). In contrast, the completely random placement helps far less, and only for small topologies - for larger N its efficiency becomes insignificant. This is because only new logical links connecting *different* nodes in $G^\lambda \downarrow A$ (i.e., different survivable pieces in G^λ) may help; the larger the topology, the lower the probability of achieving it with a completely random placement.

The efficiency of the placement of a new *physical* link has a more random nature. Again, the SMART-H approach helps, but its effect is not as significant nor dependent on N , as in the case of logical links. This is because the introduction of a new physical link *within the same* survivable piece may also help. For instance, in Example 8, for the logical topology G^λ and the physical topology $G^\phi \setminus \{b^\phi\}$, the piecewise survivable mapping M_A consists of two survivable pieces $\{a^\lambda, b^\lambda, c^\lambda\}$ and $\{f^\lambda, g^\lambda, h^\lambda\}$. We have shown that for this pair of topologies a survivable mapping does not exist. But it is enough to add the physical link b^ϕ to make a survivable mapping possible, as shown in Fig. 4.3c. Note that the link b^ϕ lies within the survivable piece $\{a^\lambda, b^\lambda, c^\lambda\}$.

4.9.5 Results for Application 3

Many approaches to the survivable mapping problem can be found in the literature [54–56,60,67]. In contrast to our basic technique, many of them take some additional constraints into account, such as capacity, delays, number and localization of wavelength converters. For the comparison purposes we have chosen three proposals, that focus exclusively on the survivability issue. The first one is based on Integer Linear Programming (ILP). The other two, Tabu Search and Simple Layout Algorithm, are heuristics.

ILP approach

In [55] the authors specify the necessary and sufficient conditions for a mapping to be survivable. These conditions are injected into the Integer Linear Programming (ILP) formulation, that is used to find a survivable mapping. Then a simple relaxation (ILP-Relax) for the ILP is introduced, which substantially reduces the processing time.

We ran the SMART algorithm for exactly the same topologies as in [55], namely NSFNET as the physical topology and the same 300 random graphs

of degree $\bar{d} = 3, 4$ and 5 as those in [55] for the logical topologies. A survivable mapping was found in *all runs* when using ILP, ILP-Relax and SMART approaches. Therefore it is interesting to compare the *run-times* of the algorithms. The machines were not the same, yet comparable (Sun Sparc Ultra-10 vs. Pentium 500). However, we have to stress that SMART was implemented in pure C++ whereas ILP required a dedicated program (CPLEX) that could significantly affect the results.

The run-times from [55] are reprinted in Table 4.1; the last column shows the results of the SMART algorithm. The SMART algorithm is several orders of magnitude faster than pure ILP, and about 3 orders of magnitude faster than the relaxed version of ILP. Note that the degree of the logical topology practically does not affect the run-time of SMART. This is because most of the processing time is consumed in the search of a survivable mapping (convergence of the contracted logical topology to a single node); the remaining logical links are mapped by the shortest path Dijkstra algorithm, which takes negligible time.

Average degree \bar{d}	ILP	ILP-Relax	SMART-H
3	8.3 sec	1.3 sec	0.0028 sec
4	2 min 53 sec	1.5 sec	0.0028 sec
5	19 min 17 sec	2.0 sec	0.0029 sec

Table 4.1: Run-times of ILP and SMART-H

Tabu Search and large topologies

One of the most efficient and widely used techniques to solve a survivable mapping problem is *Tabu Search*. Our implementation of Tabu Search follows the one in [54]; we will refer to it as *Tabu97*. Since Tabu Search turned out to be substantially faster than the ILP approach (described in previous section), we carried out the simulations for relatively large graphs and studied Tabu Search and SMART scalability.

The *physical topology* is an f -lattice (Fig. 4.8b) with the fraction of deleted edges f ranging from 0 to 0.35. The maximal value 0.35 was chosen in such a way that even the smallest topologies could be 2-edge-connected.⁴ The *logical topology* was a 2-edge-connected random graph of the average vertex degree $\bar{d} = 4$. In Fig. 4.10 we present the results obtained in simulations

⁴A full lattice of 16 nodes has 24 edges. So for $f = 0.35$, the corresponding f -lattice will have $24(1-0.35) \simeq 16$ edges, which is the smallest value still enabling 2-edge-connectivity (a cycle topology).

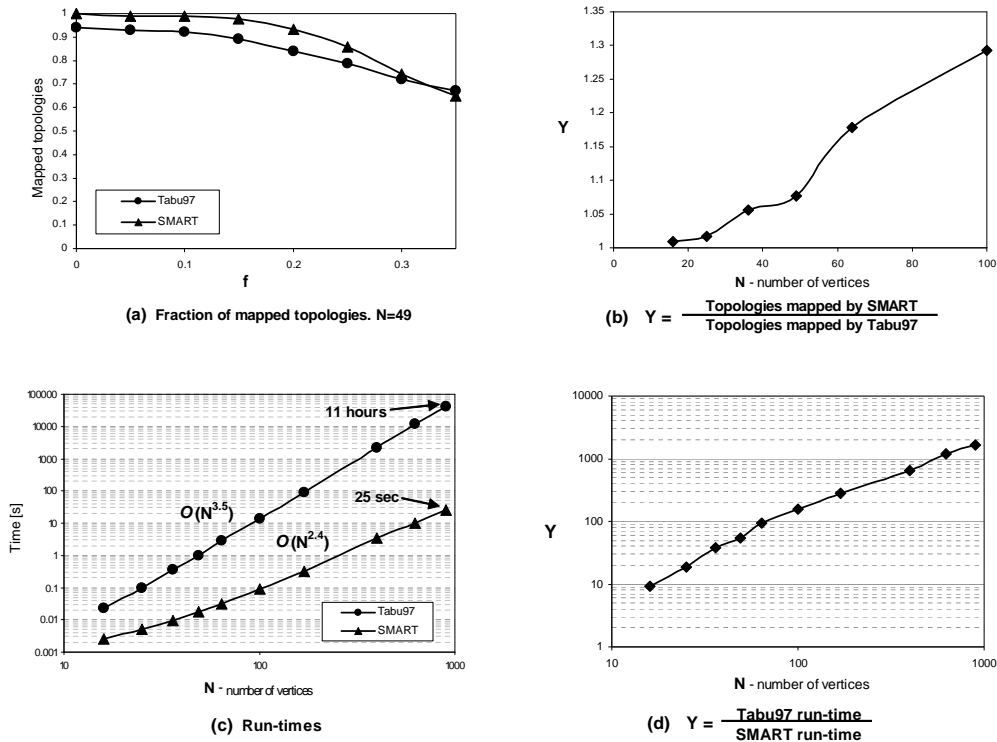


Figure 4.10: SMART-H vs. Tabu97. Random graph logical topology of average node degree $\bar{d} = 4$ mapped on f -lattice physical topology of N nodes, $f=0\dots0.35$, $N=16\dots900$. A pair of topologies is declared to be ‘mapped,’ if it is mapped in a survivable way.

on a Pentium 4 machine. We investigate the topologies with a number of vertices ranging from 16 to 900. Figs. 4.10a,b are related to the *efficiency* of algorithms, i.e., their ability to find a survivable mapping. In Fig. 4.10a the fraction of successfully mapped topologies is drawn against the fraction of deleted edges f for a constant number of nodes $N = 49$. We observe that the fraction of mapped topologies is substantially higher for SMART than for Tabu97. Fig. 4.10a also confirms that, in general, it is more difficult to map the same logical topology on a sparser physical topology.

Fig. 4.10b depicts the dependence of the efficiency on the size of topologies. The advantage of SMART grows with N ; for $N = 16$ it is negligible but for $N = 100$ it already reaches 30%. Again, this is because SMART divides the whole task into tiny subtasks. For a small N there is not much to divide and SMART cannot take advantage of this property.

It is interesting to investigate the run-times of the algorithms. Since Tabu97 was integrated with our implementation of SMART (both in C++,

using the same structures and functions), it is reasonable to compare their real processing times. They are given in Fig. 4.10c in log-log scale. The curves are almost linear, hence the observed complexities of the algorithms are polynomial, with $O(N^{3.5})$ for Tabu97 and $O(N^{2.4})$ for SMART. Both values fit in the theoretical maximal bounds, which are $O(N^4)$ [54] and $O(N^3)$, respectively. Note that Tabu97 took about *11 hours* when solving 900 node problem, which is a lot more than the *25 seconds* measured for SMART. Fig. 4.10d is a direct comparison of run-times of the two algorithms. SMART converged orders of magnitude faster than Tabu97; again the difference strongly depends on N .

Simple Layout Algorithm

The Simple Layout Algorithm by Sasaki et al. [67], similarly to SMART, breaks down the survivable mapping problem into a set of small and easy to solve subproblems. Therefore we expected it to be as fast as SMART. We implemented the version of the Simple Layout Algorithm with multiple link computation, *DEGR* node ordering and *EL-M* link cost (the detailed description can be found in [67]). The authors reported this set of parameters to be the most efficient. The physical and logical topologies were the same as described in the previous subsection (Tabu Search). The run-times of the Simple Layout Algorithm were about three times shorter than those of SMART. However, as illustrated in Fig. 4.11, the fraction of topologies mapped (in a survivable way) by Simple Layout Algorithm is dramatically smaller than the fraction mapped by SMART. Although for $N = 16$ the results are comparable, the $N = 64$ already yields a forty times difference. The reason for the poor efficiency of the Simple Layout Algorithm and its strong dependence on the size of topologies is the following. By construction, the Simple Layout Algorithm prevents only *single* nodes from being separated in the case of a fiber failure. Since in small graphs (authors used six-node topologies for simulations) a separation of a single node is the most common type of loss of connectivity, the Simple Layout Algorithm approach is efficient. But for large graphs there are substantially more possibilities for the separation of larger (than a single node) subgraphs. The Simple Layout Algorithm does not take them into account, which results in a dramatic fall in its efficiency. This is yet another evidence that the survivability is a complex problem and has to be carefully addressed.

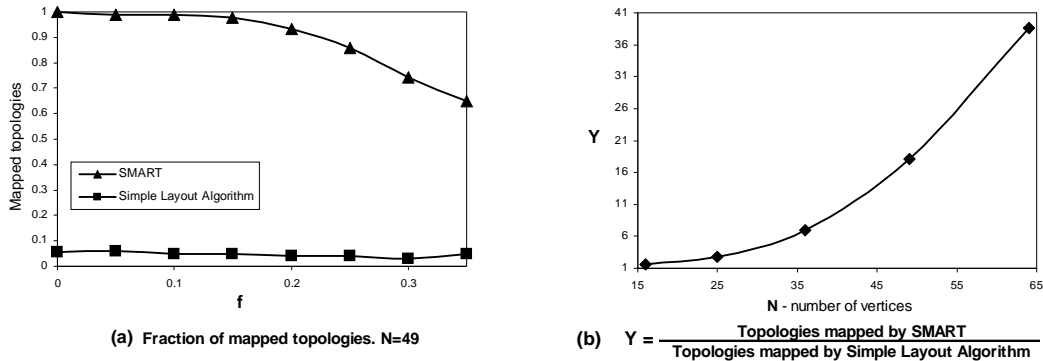


Figure 4.11: SMART vs. Simple Layout Algorithm. A pair of topologies is declared to be ‘mapped,’ if it is mapped in a survivable way.

4.10 Extension 1: Span failures

So far we have only considered single physical link failures. In the following three sections we briefly describe the extensions of SMART-H to deal with failure scenarios more sophisticated than a single physical link failure. The main idea is always the same - at each iteration we find a survivable mapping of some subgraph of the contracted logical topology and then contract this subgraph into one node. For more detailed information, please refer to [6,8,9].

We begin with *span failures* described in 4.2.2. So far, span failures were addressed mainly by physical layer protection [48,68]. In contrast, we use the IP restoration approach [8].

4.10.1 Changes: SMART-Span

A mapping is declared *span-survivable* if it preserves the connectivity of the logical graph after any single cut of a fiber or of a multi-link span. The SMART approach requires only a minor upgrade of the DisjointMap function to be adapted to span-survivability. Assume for instance that in the physical topology in Fig. 4.5 the fibers a^ϕ and c^ϕ are laid partially in the same span. Then, the mapping M_B shown at Iteration 1 is not survivable and must be rejected by the SMART-Span algorithm. However, assigning $M_B(c^\lambda) = \{e^\phi, g^\phi, f^\phi, d^\phi\}$ instead of $M_B(c^\lambda) = \{c^\phi\}$ solves the problem. A modification of SMART taking into account span failures will be referred to as *SMART-Span*.

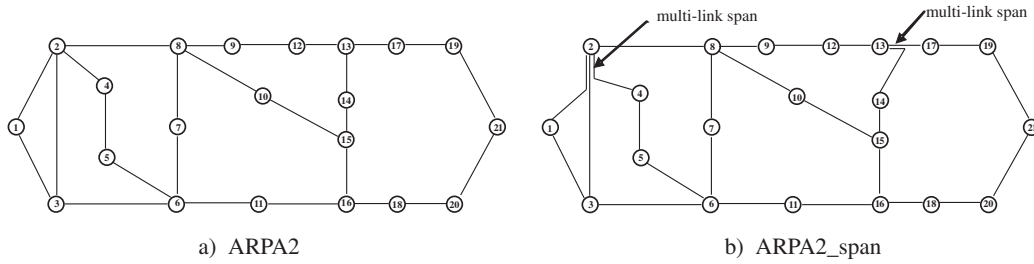


Figure 4.12: Physical topologies used for simulations of span failures: a) ARPA2; b) ARPA2 with two multi-link spans (spans with multiple physical links).

4.10.2 Results

For the illustration of a SMART-Span version of our algorithm we modified the ARPA2 network by assuming two multi-link spans as depicted in Fig. 4.12b. The logical topologies were 2-edge-connected random graphs of average node degree $\bar{d} = 3 \dots 6$. We generated 1000 logical topologies for each \bar{d} .

As this is the first time the span-survivability is addressed by the IP restoration approach, we have no benchmark against which we can compare our algorithm. Therefore the only comparison is made with the standard version of SMART.⁵ The results are presented in Table 4.2. The Shortest Path algorithm and SMART (with no span protection) are added for reference. The results show that even very few multi-link spans may result in a loss of survivability when the mapping does not take them into account. However, SMART-Span enables an efficient protection. The run-times of SMART and SMART-Span (not shown here) are comparable.

Average degree \bar{d}	Shortest Path	SMART-H	SMART-Span
3	999	681	183
4	994	373	64
5	965	177	8
6	867	129	1

Table 4.2: Number of topologies *failed* to be mapped in a span-survivable way (out of 1000).

⁵This is also the case for the other types of failures addressed in the following two sections.

4.11 Extension 2: Node failures

A single *node failure* is another typical problem in IP/WDM networks, as described in 4.2.2 and in [69]. We have addressed it by a heuristic in [8] and by formal analysis in [6].

4.11.1 Changes: SMART-Node

Clearly, after a failure of a vertex $v \in V$, the logical topology is disconnected - at least the vertex v is always separated. The best we can do is to keep connected the remaining part of the logical topology. Therefore a mapping is declared *node-survivable* if, after any single failure of vertex $v \in V$, the logical topology $G^\lambda \setminus \{v\}$ remains connected.

Node-survivability is a significantly more difficult problem than survivability. However, the SMART approach efficiently solves this problem as well. We need only the DisjointMap function to search for *node-disjoint* mappings instead of link-disjoint ones. Then no single node failure can disconnect the mapped ring. We will refer to this version of our algorithm as *SMART-Node*.

4.11.2 Results

SMART-Node was tested in the same setting as SMART-Span in previous section, except that the physical topology was the ARPA2 network with no modifications, as in Fig. 4.12a. The results of the simulations are presented in Table 4.3. SMART-Node performs fairly well and its run-time (not shown here) is only slightly higher than that of ‘pure’ SMART.

Average degree \bar{d}	Shortest Path	SMART-H	SMART-Node
3	999	903	428
4	995	717	204
5	963	410	23
6	859	228	3

Table 4.3: Number of topologies *failed* to be mapped in a node-survivable way (out of 1000).

4.12 Extension 3: Double-link failures

Finally, we consider the double-link failures. As we argued in 4.2.2, they are less frequent than single link failures, but still possible. In the literature,

double-link failures were addressed mainly at the physical layer [70–72]. In [8, 9] we propose and analyze an IP restoration approach, based on the following extension of SMART.

4.12.1 Changes: SMART-2Link

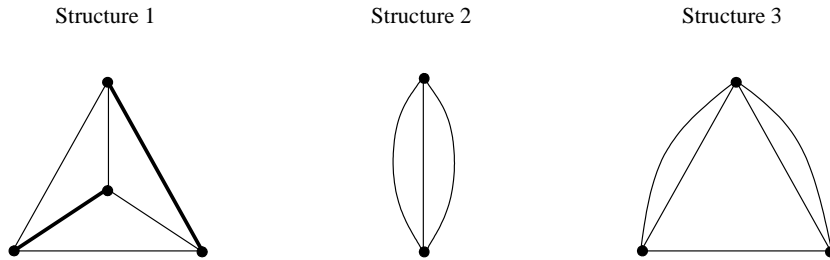


Figure 4.13: Subgraphs of the contracted logical topology attempted to be mapped to obtain 2Link-survivability.

If the logical topology remains connected after any two single link failures, then the mapping is declared *2Link-survivable*.

To obtain a 2Link-survivable mapping we use the basic idea of SMART: at each iteration we find a 2Link-survivable mapping of some subgraph of the contracted logical topology G^β and then contract this subgraph into one node. However, the subgraph we find will differ. Before, in the case of single-link failures, the simplest subgraph we could search for was a cycle; the disjoint mapping of this cycle ensured survivability. In the case of double-link failures, the subgraphs are more complex as they must be at least 3-edge-connected. Fig. 4.13 presents the three structures we used. Note that only the first one is a simple graph, i.e., it has no multi-edges. In fact it is the smallest possible 3-edge-connected simple graph. Since the logical topology is also a simple graph, only the Structure 1 may be found at the first iteration of the algorithm. However, at subsequent iterations we work on the *contracted* logical topology, that may have multi-edges and self-loops. Then Structures 2 and 3 will become useful. The extension of our algorithm searching for a 2Link-survivable mapping is called *SMART-2Link*.

4.12.2 Results

Since the necessary condition for 2Link-survivability is the 3-edge connectiveness of the physical and logical topologies, we used NSFNET3EC presented in Fig. 4.14b as the physical topology and 3-edge-connected random graph

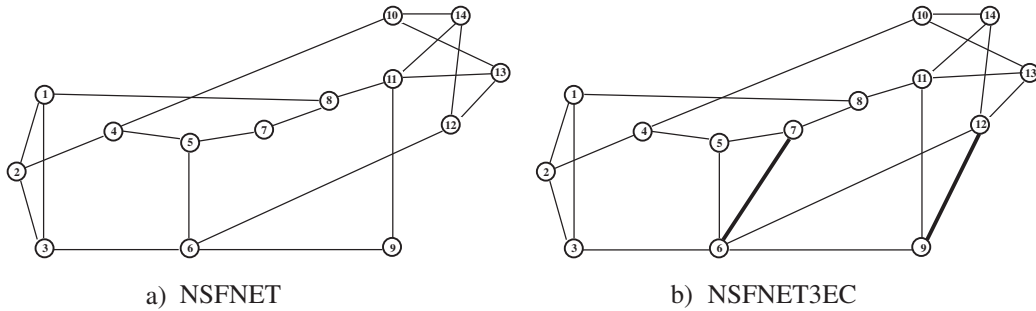


Figure 4.14: Physical topologies used for simulations of double-links failures: a) NSFNET; b) NSFNET with two additional links to obtain 3-edge-connectedness (3EC).

of average vertex degree $\bar{d} = 5 \dots 7$ as the logical topology. Note that the degree is larger than in previous examples, because 2Link-survivability naturally requires the topologies to be connected more strongly.

Table 4.4 presents the results. Clearly ‘pure’ SMART is completely inefficient when dealing with double-link failures, whereas, SMART-2Link performs a lot better. The run-time of SMART-2Link (not shown here) was three times longer than that of SMART.

Average degree \bar{d}	Shortest Path	SMART-H	SMART-DF
5	1000	998	422
6	992	971	36
7	950	924	3

Table 4.4: Number of topologies *failed* to be mapped in a 2Link-survivable way (out of 1000).

4.13 Extension 4: Capacity constraints

The SMART approach described in this chapter is a powerful tool for studying the survivability of two-layer systems (such as IP-over-WDM) from a *topological* perspective. Therefore, as in the approaches in [54,55], we have assumed infinite capacities (number of wavelengths) on each physical fiber. This has pros and cons. On the one hand, this makes ‘negative results’ more general: if we prove that a survivable mapping does not exist for a particular pair of physical and logical topologies with infinite physical capacities, then this proof holds for any combination of finite capacities. On the other hand,

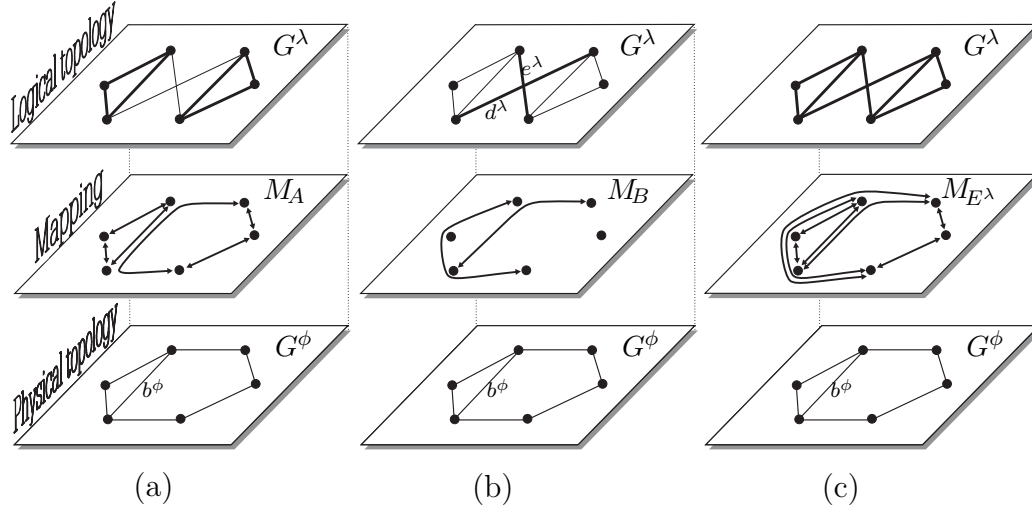


Figure 4.15: An illustration of invalidness of Theorem 3 [Existence of a survivable mapping] in the presence of capacities - a counterexample. Assume that all physical links have very big (or infinite) capacities except the link b^ϕ that has got a capacity equal to 2. The mapping M_A presented in (a) is piecewise-survivable and respects the capacity constraints. M_A uses b^ϕ two times, which means that no more paths can be routed on b^ϕ . But it is easy to see that every survivable mapping M_B of the logical links d^λ and e^λ must use b^ϕ ; an example of M_B is shown in (b). If Theorem 3 held in the presence of capacities, we could conclude that there does not exist a survivable mapping of G^ϕ on G^λ respecting the capacity constraints. However, such a mapping exists and is shown in (c).

a ‘positive result’ (i.e., a survivable mapping) found for infinite capacities is not necessarily applicable to a scenario with given capacity constraints.

In this section we show that SMART, although primarily devised to tackle purely topological aspects of the survivability problem, can also incorporate some additional real-life constraints, which makes our approach an interesting alternative for the heuristics available in the literature.

4.13.1 Which theoretical results hold?

We first review the theorems given in this chapter. Assume that the capacity constraints are respected at every moment. This means, for example, that in Step 2 of the SMART algorithm the pair $[G_{sub}^\beta, M_B]$ must be not only survivable, but that the resulting mapping $M_A \cup M_B$ (Step 3) should not exceed the capacity of any physical link.

With this assumption it is easy to see that Theorems 1 and 4 hold. In contrast, Corollary 1 holds only in one direction, i.e., if the contracted topol-

ogy G^β converges to a single node then the underlying mapping is survivable and capacity constraints are not violated.

Unfortunately, Theorems 2, 3, and 5 do not hold, because their proofs implicitly require unlimited capacities of physical links. To better understand this, we present a counterexample for Theorem 3 in Fig. 4.15.

As a result, the SMART algorithm cannot be used to verify the *existence* of a survivable mapping respecting capacity constraints. Thus the functionality of SMART is reduced to that of a pure heuristic. In the remainder of this section we evaluate the effectiveness and speed of this heuristic.

4.13.2 SMART-C

In the implementation of the SMART algorithm with the capacity constraints (called “SMART-C”) we have exploited the following idea. First, we construct a “light” survivable mapping that uses relatively few logical links. This is achieved by accepting in Step 2 only the mappings that are not significantly longer than their shortest path counterpart. Next, we map the remaining logical links trying to satisfy the capacity constraints, instead of applying the shortest path mapping suggested in Section 4.6.

4.13.3 Results

As a benchmark, we take one of the most efficient and widely used heuristic to solve a survivable mapping problem, Tabu Search [54,56,59,60]. We have followed the implementation given in [59]; it is a version of [54] that takes the capacity constraints into account. We refer to this algorithm as Tabu98.

We compare SMART-C with Tabu98 in Fig. 4.16. On one hand, Tabu98 always finds slightly better solutions than SMART-C - on average Tabu98 needs several percent smaller physical link capacities to succeed (see Fig. 4.16b). On the other hand, the comparison of the run-times of the examined algorithms (see Fig. 4.16a) reveals a big difference that grows with the network size to several orders of magnitude.

These results are not surprising. At every iteration, Tabu98 examines the mapping of the entire logical topology, whereas SMART-C processes only a small portion of it. This gives Tabu98 a global view and results in a slightly better routing of lightpaths, but at the very significant cost of speed and scalability. Therefore, with finite link capacities, the two approaches are complementary.

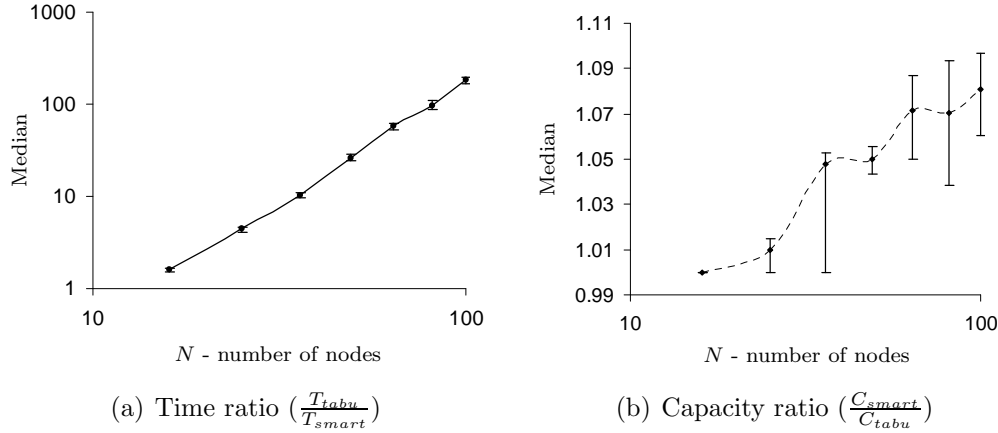


Figure 4.16: Survivability with capacity constraints: SMART-C vs. Tabu98. Two-edge-connected random logical topologies of average node degree $\langle k^\lambda \rangle = 4$ are mapped on the f -lattice physical topology, with $f = 0.3$. The size of topologies ranges from $N = 16$ to 100. For each value of N we generate 500 topology pairs and run SMART-C and Tabu98 until they succeed (i.e., find a survivable mapping that meets the capacity constraints) or the maximum number of iterations is reached. For the plots we took only the cases where both SMART-C and Tabu98 succeeded (Tabu98 succeeded less often than SMART-C). All fibers in physical topology have the same capacity, chosen separately for every pair of topologies. For every pair of topologies we compare SMART-C and Tabu98 with respect to two metrics: (a) (log-log) Comparison of run-times T_{smart} of SMART-C with run-times T_{tabu} of Tabu98. We fix the capacities of the physical links to the lowest value C_{smart} for which SMART-C succeeds. Tabu98 is run using the same capacity C_{smart} . (b) (log-lin) Comparison of minimal capacities. We compare the value C_{smart} with the minimal value C_{tabu} of capacities for which Tabu98 succeeds. In (b) the run-times are ignored. In both figures, all points are the medians over all topology pairs taken into account; the confidence interval for medians are computed at 0.95 confidence level.

4.14 Conclusion

In this chapter we studied the IP restoration in IP/WDM networks, as an example of two-layer systems. In this setting, the physical and the logical graphs are considered fixed. The goal is to find a survivable mapping, i.e., a mapping that keeps the logical graph connected in the presence of the most common physical failures.

In order to achieve this, we defined a *piecewise survivable mapping* which preserves the survivability of some subgraphs of the logical topology. The formal analysis of the piecewise survivable mapping enabled us to specify the

Functionality	SMART	ILP	Tabu
fast and scalable	✓✓	×	×
capacity and other constraints	✓	✓	✓
verification of a solution existence	✓	✓	×
node failures	✓	?	?
span failures	✓	?	?
multiple failures	✓	?	?
tracing and repairing the vulnerable areas	✓	×	×

Table 4.5: Comparison of efficiency and functionalities of SMART, FastSurv, ILP and Tabu97. The question mark “?” means that the option might be possible to realize, but, to the best of our knowledge, nobody did it to date.

necessary and sufficient conditions for the existence of a survivable mapping. This has led us to the SMART algorithm that is guaranteed to converge to a survivable mapping if and only if it exists. As one iteration of SMART is an NP-complete problem, we adapted it by using a heuristic, which results in the SMART-H algorithm. Most of the theoretical results obtained for SMART carry over to SMART-H, which makes the latter a practical tool that substantially simplifies the verification of the existence of a survivable mapping. A second application of SMART-H is tracing vulnerable areas in the network and pointing where new link(s) should be added to enable a survivable mapping. Finally, SMART-H can serve as an efficient and scalable heuristic that searches for a survivable mapping.

Many features and applications of SMART were not available under previous approaches. We have also extended SMART to a number of common types of failures that were not addressed to date by IP restoration techniques. We summarize it in Table 4.5.

To conclude, the formal analysis of the piecewise survivable mapping gives us a powerful tool to designing, diagnosing and upgrading the topologies in IP/WDM networks.

APPENDIX

4.A Proofs

In this section we prove all Theorems introduced before. For this purpose we use the following definition of survivability, equivalent to Definition 3.

Definition 5 (Survivability) Let $G^\lambda = (V, E^\lambda)$, $A \subset E^\lambda$ and $G^\beta = (V^\beta, E^\beta) = G^\lambda \downarrow A$. Take any connected subgraph $G_{sub}^\beta = (V_{sub}^\beta, B)$, $B \subseteq E^\beta$, of the contracted topology G^β , and let M_B be a mapping of the set B of logical links. The pair $[G_{sub}^\beta, M_B]$ is survivable if for any physical link e^ϕ and for any two vertices $u, v \in V_{sub}^\beta$, there exists a path $p_{u,v}^\beta$ in G_{sub}^β between vertices u and v , such that $e^\phi \notin M_B(p_{u,v}^\beta)$.

(Note that every path in the contracted topology, e.g., $p_{u,v}^\beta$, actually consists of logical links.)

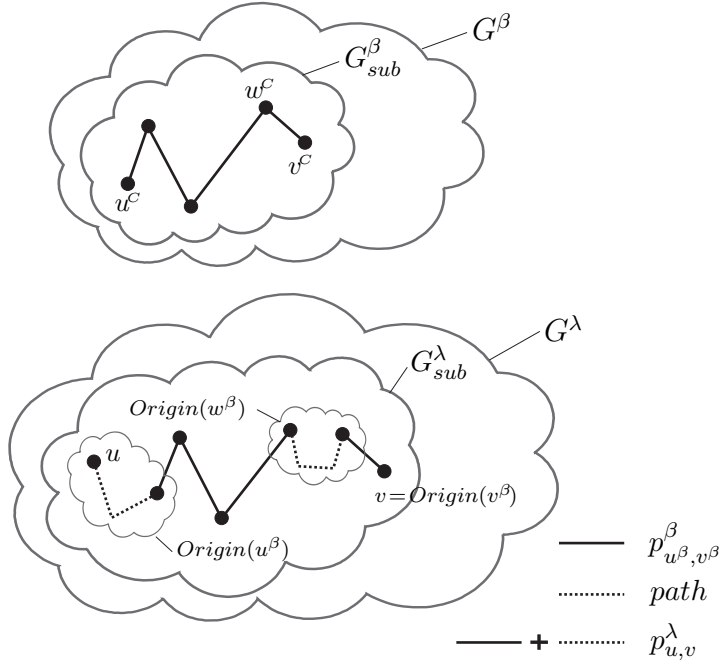


Figure 4.17: Illustration of proof of Theorem 1. A first portion of the path $p_{u,v}^\lambda$ is the path $p_{u^\beta, v^\beta}^\beta$ found in G_{sub}^β . Next it is completed, where necessary, with the patches found in origins of the nodes of $p_{u^\beta, v^\beta}^\beta$.

4.A.1 Proof of Theorem 1

(Please refer to Fig. 4.17.)

First note that since $G^\beta = G^\lambda \downarrow A$, no logical edge from the set A can be found in G^β , which implies that $A \cap B = \emptyset$. Therefore the operation $M_A \cup M_B$ is always well defined, as in (4.2) and (4.3).

Let $M_{A \cup B} = M_A \cup M_B$ and $G_{sub}^\lambda = Origin(G_{sub}^\beta)$. We have to prove that the

pair $[G_{sub}^\lambda, M_{A \cup B}]$ is survivable. Take any single physical link e^ϕ and two vertices $u, v \in G_{sub}^\lambda$. According to Definition 5 we have to show that there exists a path $p_{u,v}^\lambda$ in G_{sub}^λ such that $e^\phi \notin M_{A \cup B}(p_{u,v}^\lambda)$. The path $p_{u,v}^\lambda$ is constructed in two steps, (i) and (ii).

(i) A first portion of $p_{u,v}^\lambda$ is found in the contracted graph G^β (recall that G^β consists of logical edges), as follows. Call $u^\beta, v^\beta \in V_{sub}^\beta$ the vertices in $G_{sub}^\beta = (V_{sub}^\beta, B)$ whose origins contain u and v , respectively, i.e., such that $u \in Origin(u^\beta)$ and $v \in Origin(v^\beta)$. Find a path $p_{u^\beta, v^\beta}^\beta$ in G_{sub}^β , such that $e^\phi \notin M_B(p_{u^\beta, v^\beta}^\beta)$. This is always possible since the pair $[G_{sub}^\beta, M_B]$ is survivable. We take $p_{u^\beta, v^\beta}^\beta$ as the first portion of $p_{u,v}^\lambda$.

(ii) We now turn our attention to the origins of vertices in the path $p_{u^\beta, v^\beta}^\beta$. Take any two consecutive edges a^λ and b^λ of $p_{u^\beta, v^\beta}^\beta$, and let w^β be their common end-node in G_{sub}^β . If $Origin(w^\beta)$ is not a single node in G_{sub}^λ , then a^λ and b^λ might not have a common end-node in G_{sub}^λ . However, by piecewise survivability of $[G^\lambda, M_A]$, the pair $[Origin(w^\beta), M_A]$ is survivable. Therefore, if we denote respectively by $v_a, v_b \in Origin(w^\beta)$ the end-nodes of a^λ and b^λ , that belong to $Origin(w^\beta)$, we can find a logical path p_{v_a, v_b}^λ in $Origin(w^\beta)$ connecting v_a and v_b , such that $e^\phi \notin M_A(p_{v_a, v_b}^\lambda)$. We call this path a patch of w^β and denote it by $patch(w^\beta)$. If for a given w^β , the edges a^λ and b^λ have a common end-node v^λ in G_{sub}^λ then $patch(w^\beta) = v^\lambda$.

For every vertex $w^\beta \in p_{u^\beta, v^\beta}^\beta$, find $patch(w^\beta)$. If $w^\beta = u^\beta$ then $patch(u^\beta)$ will connect the logical vertex u with an end-node of the first logical edge in $p_{u^\beta, v^\beta}^\beta$, instead of connecting two end-nodes. The same holds for $w^\beta = v^\beta$.

To summarize, in step (i) we have found the path $p_{u^\beta, v^\beta}^\beta$ in the contracted subgraph G_{sub}^β . Next, in step (ii), we have constructed a set of patches for each vertex of this path. Now we combine steps (i) and (ii) to obtain the full path $p_{u,v}^\lambda$:

$$p_{u,v}^\lambda = p_{u^\beta, v^\beta}^\beta \cup \left\{ \bigcup_{w^\beta \in p_{u^\beta, v^\beta}^\beta} patch(w^\beta) \right\}. \quad (4.4)$$

The logical path $p_{u,v}^\lambda$ connects the vertices u and v and has been constructed in such a way, that

$$e^\phi \notin M_B(p_{u^\beta, v^\beta}^\beta) \quad (4.5)$$

$$e^\phi \notin M_A(patch(w^\beta)) \quad \text{for every } w^\beta \in p_{u^\beta, v^\beta}^\beta. \quad (4.6)$$

Since $M_A \cup M_B = M_{A \cup B}$ and $A \cap B = \emptyset$, we can rewrite (4.5) and (4.6) as

$$e^\phi \notin M_{A \cup B}(p_{u^\beta, v^\beta}^\beta) \quad (4.7)$$

$$e^\phi \notin M_{A \cup B}(\text{patch}(w^\beta)) \quad \text{for every } w^\beta \in p_{u^\beta, v^\beta}^\beta. \quad (4.8)$$

Combining (4.4), (4.7) and (4.8) yields finally that $e^\phi \notin M_{A \cup B}(p_{u, v}^\beta)$, which proves the claim. ■

4.A.2 Proof of Theorem 2

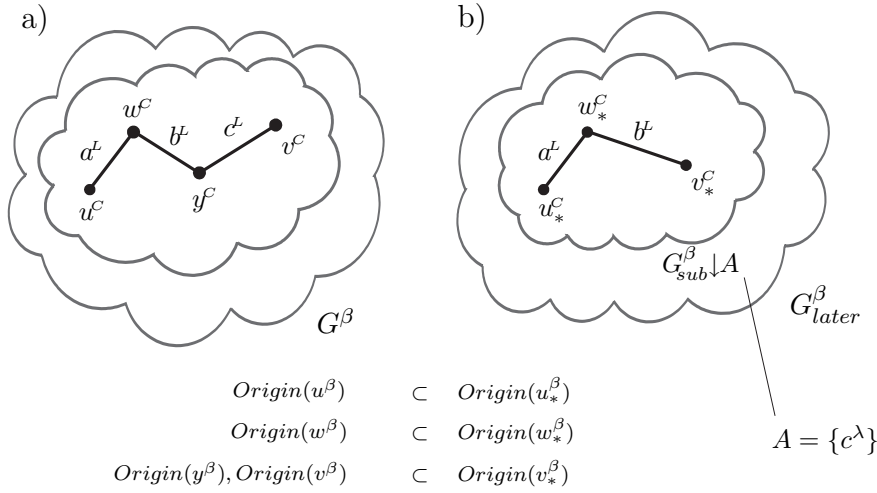


Figure 4.18: Illustration of the proof of Theorem 2. (a) The original subgraph G_{sub}^β and a path $p_{u^\beta, v^\beta}^\beta$ that avoids e^ϕ in its mapping. (b) The subgraph G_{sub}^β contracted on the set $A = \{c^\lambda\}$ of logical edges; the resulting subgraph is denoted by $G_{sub}^\beta \downarrow A$. The path $p_{u_*^\beta, v_*^\beta}^\beta$ originates from $p_{u^\beta, v^\beta}^\beta$, hence it also avoids e^ϕ in its mapping.

(Please refer to Fig. 4.18)

Take any single physical link $e^\phi \in E^\phi$ and two vertices $u_*^\beta, v_*^\beta \in G_{sub}^\beta \downarrow A$. According to Definition 5 we have to show that there exists a path $p_{u_*^\beta, v_*^\beta}^\beta$ in $G_{sub}^\beta \downarrow A$ such that $e^\phi \notin M_B(p_{u_*^\beta, v_*^\beta}^\beta)$.

First, find in G_{sub}^β two vertices $u^\beta, v^\beta \in V_{sub}^\beta$, such that

$$Origin(u^\beta) \subseteq Origin(u_*^\beta), \text{ and} \quad (4.9)$$

$$Origin(v^\beta) \subseteq Origin(v_*^\beta). \quad (4.10)$$

Note that since $G_{sub}^\beta \downarrow A$ is created by contracting some edges in G_{sub}^β , vertices u^β and v^β always exist (they are not necessarily unique). Since the pair $[G_{sub}^\beta, M_B]$ is survivable, there exists a path $p_{u^\beta, v^\beta}^\beta$ in G_{sub}^β such that $e^\phi \notin M_B(p_{u^\beta, v^\beta}^\beta)$. Define a sequence of logical edges p_*^β by contracting in $p_{u^\beta, v^\beta}^\beta$ all edges that exist also in A , i.e.,

$$p_*^\beta = p_{u^\beta, v^\beta}^\beta \downarrow (A \cap p_{u^\beta, v^\beta}^\beta). \quad (4.11)$$

Since $p_{u^\beta, v^\beta}^\beta$ is a path in G_{sub}^β , and since the contraction an edge merges its two end-nodes and thus preserves its continuity, p_*^β is a path in $G_{sub}^\beta \downarrow A$. Moreover, relations (4.9,4.10) imply that the path p_*^β connects u_*^β and v_*^β in $G_{sub}^\beta \downarrow A$. Finally, $e^\phi \notin M_B(p_{u^\beta, v^\beta}^\beta)$ and (4.11) yields that $e^\phi \notin M_B(p_*^\beta)$. Therefore p_*^β is the path $p_{u_*^\beta, v_*^\beta}^\beta$ that we are searching for. ■

4.A.3 Proof of Theorem 3

\Leftarrow We know that the pair $[G^\lambda, M_A]$ is piecewise survivable. Suppose that there exists a mapping $M_{E^\lambda \setminus A}^{surv}$, such that the pair $[G^\lambda \downarrow A, M_{E^\lambda \setminus A}^{surv}]$ is survivable. Then, by Theorem 1, the pair $[Origin(G^\lambda \downarrow A), M_A \cup M_{E^\lambda \setminus A}^{surv}] = [G^\lambda, M_A \cup M_{E^\lambda \setminus A}^{surv}]$ is also survivable. So the mapping $M_{E^\lambda}^{surv} = M_A \cup M_{E^\lambda \setminus A}^{surv}$ is a survivable mapping of G^λ on G^ϕ .

\Rightarrow Assume that a survivable mapping of G^λ on G^ϕ exists, call it $M_{E^\lambda}^{surv}$. Now, by taking $G_{sub}^\beta := G^\lambda$ and $M_B := M_{E^\lambda}^{surv}$, Theorem 2 yields that $[G^\lambda \downarrow A, M_{E^\lambda \setminus A}^{surv}]$ is survivable. Consequently, the pair $[G^\lambda \downarrow A, M_{E^\lambda \setminus A}^{surv}]$ is also survivable. ■

4.A.4 Proof of Theorem 4

(By induction)

INITIALIZATION:

Initially $G^\beta = G^\lambda$. Therefore the origin of any vertex $v^\beta \in V^\beta$ is a single node in G^λ , and it cannot be disconnected. Hence for every $v^\beta \in V^\beta$, the pair $[Origin(v^\beta), M_A]$ is survivable and consequently the pair $[G^\lambda, M_A]$ is piecewise survivable.

INDUCTION:

Assume that after some iteration the pair $[G^\lambda, M_A]$ is piecewise survivable. We have to prove that after the next iteration of the algorithm, the updated

mapping \widehat{M}_A will still form a piecewise survivable pair $[G^\lambda, \widehat{M}_A]$.

One iteration of the SMART algorithm consists of Steps 2, 3 and 4, which we recall here:

2. Find $G_{sub}^\beta = (V_{sub}^\beta, B)$ and M_B , such that the pair $[G_{sub}^\beta, M_B]$ is survivable.
3. $\widehat{M}_A := M_A \cup M_B$
4. $\widehat{G}^C := G^\beta \downarrow B$

(For clarity we indicated the updated M_A and G^β by a hat: ‘ $\widehat{}$ ’)

The updated contracted topology $\widehat{G}^\beta = (\widehat{V}^\beta, \widehat{E}^\beta)$ was created from G^β by replacing $G_{sub}^\beta = (V_{sub}^\beta, B)$ by a single node, which we call \widehat{v}_{sub}^β ; the remaining nodes stayed unchanged. So $\widehat{V}^\beta = \{\widehat{v}_{sub}^\beta\} \cup V^\beta \setminus V_{sub}^\beta$. Take any $\widehat{v}^\beta \in \widehat{V}^\beta$; we have two possibilities:

(i) $\widehat{v}^\beta = \widehat{v}_{sub}^\beta$: Since $G_{sub}^\beta = (V_{sub}^\beta, B)$ was contracted into \widehat{v}_{sub}^β , their origins coincide: $Origin(G_{sub}^\beta) = Origin(\widehat{v}_{sub}^\beta)$. Since $\widehat{M}_A = M_A \cup M_B$, the pair $[Origin(\widehat{v}_{sub}^\beta), \widehat{M}_A] = [Origin(G_{sub}^\beta), M_A \cup M_B]$ is survivable by Theorem 1.

(ii) $\widehat{v}^\beta \neq \widehat{v}_{sub}^\beta$: In this case $\widehat{v}^\beta \in V^\beta \setminus V_{sub}^\beta$, so $\widehat{v}^\beta = v^\beta$. By piecewise survivability of the pair $[G^\lambda, M_A]$, the pair $[Origin(v^\beta = \widehat{v}^\beta), M_A]$ is survivable.

Since $\widehat{M}_A = M_A \cup M_B$, the pair $[Origin(\widehat{v}^\beta), \widehat{M}_A]$ is survivable as well.

Combining (i) and (ii), we have proven that for every $\widehat{v}^\beta \in \widehat{V}^\beta$, the pair $[Origin(\widehat{v}^\beta), \widehat{M}_A]$ is survivable. So, by Definition 4, the pair $[G^\lambda, \widehat{M}_A]$ is piecewise survivable. ■

4.A.5 Proof of Corollary 1

\Rightarrow We have to show that if there is only one vertex in G^β then $[G^\lambda, M_A]$ is survivable.

We have two observations: (i) By Theorem 4, the pair $[G^\lambda, M_A]$ is piecewise survivable. This means that for every vertex $v^\beta \in G^\beta$ the pair $[Origin(v^\beta), M_A]$ is survivable. (ii) There is only one vertex in G^β (i.e., $G^\beta = \{v^\beta\}$), and therefore $Origin(v^\beta) = G^\lambda$. Combining (i) and (ii), we have that $[G^\lambda, M_A]$ is survivable.

\Leftarrow We have to show that if the contracted topology G^β has more than one node then a survivable mapping of G^λ on G^ϕ does not exist.

By Theorem 4, the pair $[G^\lambda, M_A]$ is piecewise survivable. Since the algorithm has terminated before converging to a single node (i.e., in Step 2), there exists no pair $[G_{sub}^\beta, M_B]$ that is survivable. In particular, if we take $G_{sub}^\beta = G^\beta = G^\lambda \downarrow A$, there exists no pair $[G^\lambda \downarrow A, M_*]$ that is survivable. Now, by Theorem 3 there exists no survivable mapping of G^λ on G^ϕ . ■

4.A.6 Proof of Theorem 5

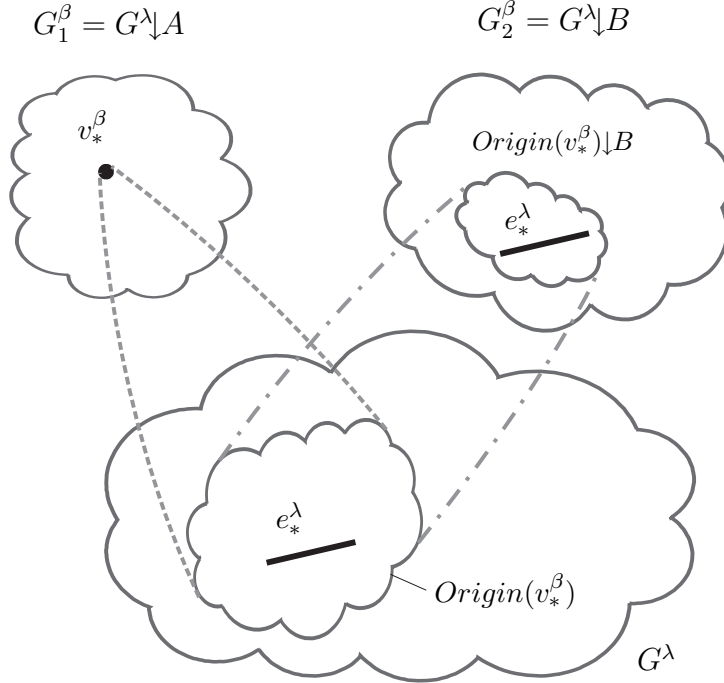


Figure 4.19: Illustration of proof of Theorem 5. We start with an edge e_*^λ that is in G_2^β , but not in G_1^β . Next, we choose a vertex $v_*^\beta \in G_1^\beta$ such that $e_*^\lambda \in \text{Origin}(v_*^\beta)$. In the topology G_2^β , $\text{Origin}(v_*^\beta)$ is contracted to $\text{Origin}(v_*^\beta) \downarrow B$ that contains at least e_*^λ . This nonempty subgraph $\text{Origin}(v_*^\beta) \downarrow B$ can be mapped in a survivable way using the mapping M_A , which leads to contradiction.

(By contradiction, Please refer to Fig. 4.19.)

Let us assume that two different runs of SMART converge to two different contracted topologies $G_1^\beta = G^\lambda \downarrow A$ and $G_2^\beta = G^\lambda \downarrow B$, and the mappings M_A and M_B , respectively. The SMART algorithm terminated in Step 2, which implies that no subgraph G_{sub1}^β of G_1^β can be mapped in a survivable way; similarly, no subgraph G_{sub2}^β of G_2^β can be mapped in a survivable way. Assume, without loss of generality, that there exists an edge e_*^λ such that $e_*^\lambda \in G_2^\beta$ and $e_*^\lambda \notin G_1^\beta$. (If such an edge does not exist, an edge satisfying a converse condition must exist, because $G_1^\beta \neq G_2^\beta$.) Since $e_*^\lambda \notin G_1^\beta$, there exists $v_*^\beta \in G_1^\beta$ such that $e_*^\lambda \in \text{Origin}(v_*^\beta)$. By Theorem 4, the pair $[G_1^\beta, M_A]$ is piecewise survivable, which implies that $[\text{Origin}(v_*^\beta), M_A]$ is survivable. Now, by Theorem 2, the pair $[\text{Origin}(v_*^\beta) \downarrow B, M_A]$ is also survivable. By construction the subgraph $\text{Origin}(v_*^\beta) \downarrow B$ contains at least the edge e_*^λ . Therefore, there exists

a non-empty subgraph $G_{sub}^\beta = Origin(v_*^\beta) \downarrow B$ of G_2^β that can be mapped in a survivable way (using the mapping M_A), which is impossible because no subgraph G_{sub2}^β of G_2^β can be mapped in a survivable way.

■

Chapter 5

Maximal Path Diversity in Overlay/IP Networks

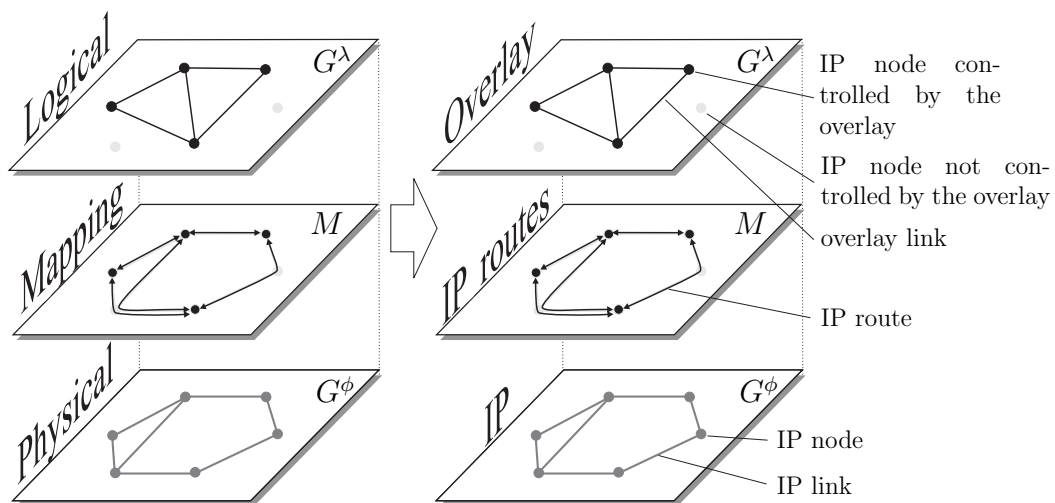


Figure 5.1: An illustration of two layers in the Overlay/IP setting.

In Chapter 4, we address the problem of appropriate design (i.e., robust to typical failures) of two-layer systems, in the context of IP/WDM networks. In this chapter we tackle the same problem, but in the context of overlay multicast networks.

5.1 Introduction

An overlay network is a virtual network of nodes and logical links that is built on top of an existing network with the purpose of implementing a network

service that is not available in the existing network [73]. Every logical link corresponds to a physical path in the underlying network.

More particularly, in this chapter we focus on application-layer overlay networks that are built on top of the Internet. We refer to this scenario as ‘Overlay/IP’ and illustrate it in Fig. 5.1.

Overlay/IP systems have numerous applications. Probably the best known example is the P2P (peer-to-peer) *file sharing systems* where the peers are organized in a (structured or random) overlay network, which enables them to efficiently find other peers with the desired content.

Another good example of overlay application is *multicast*, i.e., one-to-many or many-to-many communication. In this case the nodes are organized in a data distribution overlay network (typically a tree) and cooperate by forwarding the traffic to other nodes. Many applications, e.g., audio/video conferencing or online gaming, additionally impose strict requirements on the resulting delays. This, in turn, makes the system very vulnerable to overlay node and IP link failures. The goal of this chapter is to construct an overlay multicast system that is robust to such failures.

In this chapter we first present in Sections 5.2-5.4 a high-level introduction to the Overlay/IP setting and the problems that we address. In particular, in Section 5.2 we describe the two layers, mapping, types of failures and failure protection mechanisms in the Overlay/IP context. Next, in Section 5.3 we specify the particular type of overlay applications that we focus on. Finally, in Section 5.4 we overview our objectives, contributions and related work.

In contrast, the remaining sections rigorously address a specific problem. First, we formulate precisely the problem in Section 5.5. Next, in Section 5.6 we present a simple topology-based version of this problem, and prove that under some conditions the two formulations are equivalent. In Sections 5.7-5.9, we discuss the techniques, objective functions and heuristics that introduce path diversity and redundancy in the system, leading to good solutions of our problem. We evaluate our approach on real-life Internet topologies in Section 5.10. Finally, in Section 5.11 we conclude the chapter.

5.2 Overlay as a Two-Layer System

In this section we describe how the Overlay/IP system corresponds to the general two-layer setting introduced in Chapter 2. We consider all aspects, from the specification of the two layers and their mapping, to potential failures and typical ways of handling them.

5.2.1 The Physical Layer, Logical Layer and the Mapping in Overlay/IP Networks

Physical Layer G^ϕ

The *physical layer* $G^\phi = (V^\phi, E^\phi)$ consists of IP nodes and IP links. It is given and we have no control over it.

Logical Layer G^λ

The *logical layer* $G^\lambda = (V^\lambda, E^\lambda)$ represents the overlay system. The logical nodes $V^\lambda \subseteq V^\phi$ are those IP hosts that run the overlay application. The logical links are virtual, established at the level of the application. Therefore we are free to construct and change the logical layer.

Mapping M

The IP layer provides a routing service, meaning that it can transport an IP packet from any IP node to any other IP node. Therefore, the mapping M is defined for any pair $\langle v^\lambda, u^\lambda \rangle$ of logical nodes, and $M(\langle v^\lambda, u^\lambda \rangle) = p_{v^\lambda, u^\lambda}^\phi$ is the IP path followed by the IP packets. It can be discovered by running the traceroute tool. We have no control over M and assume it as fixed.¹

5.2.2 Failures in Overlay/IP Networks

There are two major types of failures that may be encountered in the Overlay/IP setting: overlay node failures and IP link losses.

Overlay Node Failures

In many overlay systems nodes participate on voluntary basis and may leave (e.g., by closing the application) at any time. This changes the structure of the overlay network, which may even result in its partition. Therefore, the main concern in many applications are *overlay node failures* due to the node departures, which is also called ‘node churn’.

IP Link Losses

Another potential source of failures are the packet losses at the IP layer. This is usually caused by congestions at the routers, resulting in drops of individual IP packets.

¹In reality the routes may change or alternate due to routing updates or load balancing.

5.2.3 Failure Protection Mechanisms in Overlay/IP Networks

The techniques used to increase the system's robustness strongly depend on the application of the overlay. In this section we briefly overview the general approaches. We make it more specific in the following sections, where we work on a concrete type of application.

Reactive

A simple and resource-efficient way of addressing the failures in the Overlay/IP setting is by using a reactive approach. For example, when an overlay node leaves, its absence is usually detected by missing keep-alive messages. This, in turn, triggers some (usually local) connectivity restoration mechanisms that may establish new overlay links between the remaining peers. Similarly, the IP link losses can be addressed by using some ARQ (Automatic Repeat-reQuest) techniques, as implemented in TCP, for example.

Proactive

The reactive solutions based on detection and restoration inherently introduce additional delays in the transmission, which may be unacceptable for low-delay (e.g., interactive) applications. In such cases the failures must be taken care of separately. This may be achieved by introducing some sort of *redundancy* in the system, e.g., through forward error correction coding or by adding some redundant active links in the topology of the overlay.

5.3 Application-Level Multicast (ALM)

Our main point of interest is the Application-Level Multicast (ALM). Its goal is to enable a transmission from one or more sources to many destinations. Indeed, for typical data streams (e.g., audio, video) the uplink bandwidth of the source is often not sufficient to serve directly all destinations. In ALM this problem is solved by streaming from the source to a limited number of peers that in turn forward the traffic to other peers, and so on. This results in a data distribution overlay network, typically a tree rooted at the source.

ALM is a feasible alternative for IP Multicast that is a technique not yet (and might never be) widely deployed in the Internet [74]. Therefore, it has been a hot topic in the last decade, which has resulted in numerous protocols and systems, e.g., Chaining [75], Narada [74], Nice [76], HMTP [77],

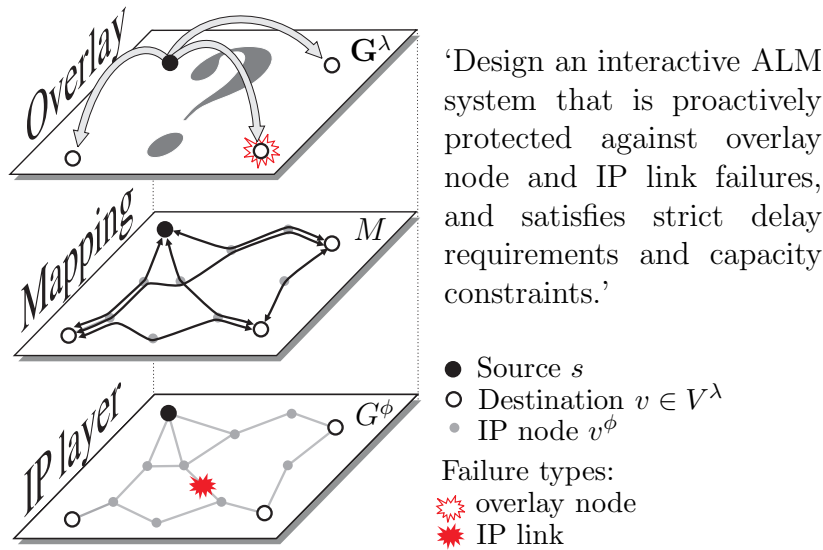


Figure 5.2: General problem illustration in Interactive Application-Level Multicast.

SplitStream [78], ZigZag [79], OMNI [80], CoopNet [81,82], Promise [83], CoolStreaming [84], Active [85], AnySee [86], Prime [87], PPLive [88].

These solutions can be roughly divided into two groups: regular ALMs and interactive ALMs. We describe them below.

5.3.1 Regular ALM

Regular ALM, or P2P streaming, focuses on the delivery of a high-quality data stream (e.g., a TV channel) from a single source to a large number of destinations. As this typically involves a one-way communication, the acceptable time lag is in the order of seconds or even minutes [87,88]. Consequently, when a node leaves the system, all nodes connected to it have enough time to find another peer with the context, without delaying the stream playback.

In other words, regular ALM can be effectively protected by *reactive* failure recovery techniques [79,89]. Moreover, some systems rely on pull-based techniques also during the regular dissemination of the stream [84,87,88], which is a scalable and resource-efficient approach [90].

5.3.2 Interactive ALM

In contrast, *interactive ALM* addresses applications such as tele- and video-conference [85,89] or network multiplayer games [91,92]. These settings have

specific features, requirements and design challenges [89] very different from those of regular ALM, as follows.

First, interactive ALM has very *strict delay requirements*, typically not exceeding a few hundreds of milliseconds. Therefore, it cannot rely on pull-based data distribution mechanisms, as they result in unacceptable delays. Instead, the stream should be quickly *pushed* through an organized structure such as a delay-optimized tree [80,89,93,94] for an overview).

Second, even if the total number of participants of a teleconference is large, usually only a small group of them interact directly [85,89,95] Similarly, in network multiplayer games, the players can be organized into small groups of interacting users who are close to each other in the virtual world [91]. Therefore, interactive applications are typically of a *limited size*, ranging from a few to several tens of nodes.

Third, reactive failure recovery schemes usually take too much time to meet the target delay requirements of interactive applications [96]. Instead, we should use *proactive failure protection techniques* that guarantee timely data delivery in the presence of most common failures.

Finally, the main concern in regular ALM are *overlay node failures* due to the node departures (i.e., ‘node churn’). In interactive ALM, however, this problem is naturally limited, due to the small number of peers and their commitment to the application. In contrast, what can become a serious impediment are *packet losses at the IP layer*. Indeed, a lost IP packet can be easily and automatically retransmitted at the cost of larger delays allowed in regular ALM, but it must be taken care of separately under the low-latency constraint.

5.3.3 Design Goals in Interactive ALM

To conclude, an interactive ALM system should:

1. push the data stream through a delay-constrained structure,
2. proactively protect the system against overlay node failures, and
3. proactively protect the system against IP link losses.

We illustrate these three design goals in Fig. 5.2.

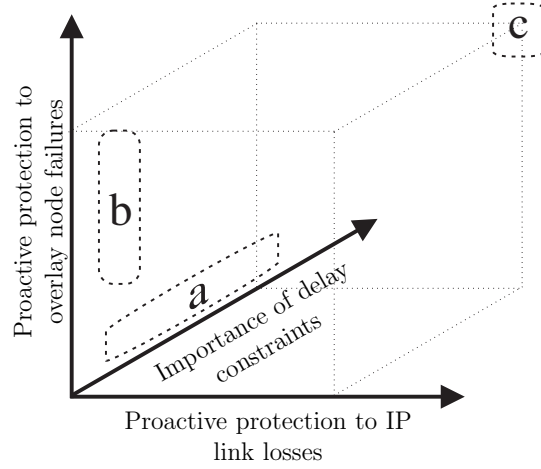


Figure 5.3: Existing ALM approaches (a,b) and the goal of this work (c) with respect to three factors critical in interactive ALM.

5.4 Our Objectives and Achievements

5.4.1 We consider the Interactive ALM

Note that in contrast to regular ALM, interactive ALM directly involves both the IP layer and the overlay. This coincides well with the main topic of this thesis (i.e., the existence and interactions between two layers), and therefore *the main focus of this chapter is the interactive ALM*.

5.4.2 Vulnerability \mathcal{W}

In Chapter 2.4 we introduced the general notion of Vulnerability \mathcal{W} . In this chapter we specify \mathcal{W} as the expected fraction of packets not delivered on time at the destinations under the presence of overlay node failures and IP link losses. Such a definition captures well the design goals (1-3) in Section 5.3.3.

To summarize, given G^ϕ and M , our goal is to design the logical layer G^λ in such a way that \mathcal{W} is minimized.

5.4.3 Related work

The problem of constructing an interactive ALM system has been only partially addressed to date. In Fig. 5.3 we position the existing work and our approach with respect to the three design goals (1-3) listed in Section 5.3.3.

For example, minimizing delays is the main goal in [80,85,89,93]. As

these solutions typically use failure protection that is reactive, they fall in the category (a) in Fig. 5.3.

Another group of approaches, indicated by (b) in Fig. 5.3, is the ALM systems that exploit the *overlay path diversity* combined with some redundancy codings [78,82,97]. This results in a good protection to overlay node failures and some (limited) protection to IP losses, but the strict delay constraint is not among the primary concerns.

Our objective is to satisfy simultaneously all three goals (1-3), which places us in the region (c) in Fig. 5.3. In order to achieve this, we leverage not only on the overlay path diversity, but also on the *IP path diversity*. Indeed, the existence of multiple IP paths was successfully exploited to fight IP losses in the context of the unicast delay-constrained connection [98–100].

Taking the topology of the underlying IP layer into account proved to be effective also in related scenarios, such as placing new overlay nodes within an ISP [101], choosing a good overlay backup path [102–104], streaming from several sources (peers) to one receiver [83], or overlay routing [105,106].

5.4.4 Our Contributions

We contribute to the state of the art in several aspects:

First, we define a metric called Vulnerability \mathcal{W} , based purely on the *topology* of the ALM system. We prove that, under some assumptions, \mathcal{W} is equivalent to the average packet loss rate observed at the destinations. This crucial observation allows us to focus directly on minimizing \mathcal{W} , which significantly simplifies the problem.

Second, we note that \mathcal{W} drops with the amount of *IP-level and overlay-level path diversity* available in the system. Therefore, we consider typical techniques to create a number of alternative paths in ALM, such as adding redundant cross-links, or using a set of multiple distribution trees. We propose a framework that accommodates and generalizes these approaches.

Third, within this framework we optimize the structure of ALM. In simulations on real Internet topologies, we find that maximizing the overlay-path diversity (that roughly represents the state of the art) may result in a poor IP-path diversity. Therefore, we propose to include the IP topology in the objective function and to maximize both the overlay-level and IP-level path diversity at the same time. As a result, we proactively protect the ALM system against overlay node failures *and* IP losses, reducing its Vulnerability \mathcal{W} (and thus the effective loss rate) by typically 30%-70%. Moreover, we develop a set of lower-bounds on \mathcal{W} and show that our approach is nearly optimal. Finally, we study factors that naturally limit the available IP-path diversity, such as the system size and maximal allowed redundancy.

ALM	Application-Layer Multicast
s, V^λ	source node, destination nodes
$G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$	directed overlay graph with all possible overlay edges E^λ
$\gamma_{max} \geq 0$	maximal redundancy
$t_{max}(v)$	maximal allowed delivery time
$b_{max}(v)$	maximal uplink bandwidth ratio
$r(v)$	source packet loss rate observed at node v
\bar{r}	$r(v)$ averaged over all destinations
$C^\lambda(v), C^\phi(v)$	set of critical overlay nodes / IP links of node v
$\mathcal{W}^\lambda, \mathcal{W}^\phi$	logical and physical vulnerability
$\mathcal{W}(\eta) = \eta\mathcal{W}^\lambda + \bar{\eta}\mathcal{W}^\phi$	total vulnerability
\mathbb{P}, \mathbb{E}	probability, expected value

Table 5.1: Additional/specific notation used in this chapter.

Finally, it should be stressed that our solution respects the strict delay constraints and uses only the available resources (i.e., nodes participating in the application, and no ‘exterior’ relay nodes) and protocols (i.e., no source routing).

5.5 Additional notation and problem formulation

We assume that there is one stream source, e.g., a lecturer. The other nodes, called destinations, should get the stream reliably and within the delay deadline. The existence of one main source of stream is typical of many interactive applications such as a teleconference [89]. This scenario can also be considered as a basic building block of a reliable all-to-all multicast system.

5.5.1 Overlay layer and capacity constraints

Let s be the traffic *source* node and let V^λ be a set of *destination nodes*, illustrated in Fig. 5.2. The source s generates *source packets* destined to all nodes in V^λ . The source s can send the packets to any destination node(s) that may forward them to any other destination node(s); each such communication link is interpreted as a directed overlay edge $e^\lambda \in E^\lambda$. Denote by $G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$ the directed graph with all possible overlay edges E^λ , $|E^\lambda| = |V^\lambda| + |V^\lambda|(|V^\lambda| - 1) = |V^\lambda|^2$. (Note that compared to our basic definition of G^λ in Section 2.1, we count the source s separately in the set of nodes of G^λ , i.e., we use $V^\lambda \cup \{s\}$ instead of V^λ . Moreover, we exclude from the logical edges E^λ the edges incoming to s , as they do not make any sense in the studies ALM application.)

In practice, we cannot usually use all the edges in E^λ at the same time because of the following capacity constraints. First, every node v has limited downlink and uplink capacities. The latter is a critical factor in the construction of ALM [107]. Therefore, we denote by $b_{max}(v)$ the maximal outgoing bandwidth of node v normalized by the data stream rate. For example, $b_{max}(v) = 2.5$ when the uplink capacity of v is 1Mbit/s and the stream rate is 0.4Mbit/s.² We refer to this system-specific feature as *local capacity constraint*.

Second, *global capacity constraint* γ_{max} is the maximal amount of redundancy we are allowed to introduce in the entire system. For example, $\gamma_{max} = 0.2$ means that we can add up to 20% of redundant traffic on top of the primary traffic (equal to $|V^\lambda| \cdot stream_rate$). This constraint is set (or relaxed) by the application. Fixing γ_{max} allows us to make a fair comparison between various protection schemes.

5.5.2 IP layer

The overlay is built on top of the underlying IP layer (see Fig. 5.2). As described in Section 5.2, every overlay edge e^λ is mapped on the physical graph G^ϕ as a physical path $M(e^\lambda) = \langle e_1^\phi, e_2^\phi, \dots \rangle$. Recall also, that, we have no control over the IP layer topology G^ϕ and over the mapping M , which we therefore consider as fixed and given. In contrast, we have full control over the overlay layer.

5.5.3 Packet losses

We consider overlay node failures and IP link failures. An *overlay node failure* is usually a departure of a node $v \in V^\lambda$ from the application, with no prior notification. Clearly all packets that were scheduled to be forwarded by v are lost until the system adapts to the new situation.

An *IP link failure* is usually caused by traffic congestion, which results in the loss of one or a burst of packets [108]. As these problems are typically short-lived, they are often classified as ‘soft failures’, to stress the difference from ‘hard failures’ (e.g., a cut of the IP link); we use the single-word term ‘failure’ for simplicity.

²We assume here a Constant Bit Rate (CBR) stream; non-CBR streams are not considered here.

5.5.4 Maximal allowed time $t_{max}(v)$ and average loss rate \bar{r}

We say that a source packet i is *delivered* at a destination v when i (or its copy) reaches v or i is reconstructed at v from other packets that were successfully delivered. Let $t_{max}(v)$ be the maximal delay acceptable (by the application) between the creation of a source packet and its delivery at node $v \in V^\lambda$. This delay is composed purely of the propagation times at the IP layer - the processing delays at the overlay nodes are ignored as they are very small (less than 1ms) for non-overloaded nodes and when high priority forwarding is used [109]. $t_{max}(v)$ may be the same for all nodes, or not. Denote by $r(v)$ the source packet loss rate observed at node v , i.e., the probability that a source packet is *not* delivered at v within time $t_{max}(v)$. Finally, $\bar{r} = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} r(v)$ is the source packet loss rate averaged over all traffic destinations.

5.5.5 General problem formulation: Problem P1

Now we can state the general problem as follows:

P1: *Given capacity and delay constraints $\{b_{max}(v), \gamma, t_{max}(v)\}$ for $v \in V \cup \{s\}$, design an ALM system that minimizes the average source packet loss rate \bar{r} under the presence of overlay node failures and IP link losses.*

In other words, we maximize the robustness of ALM to overlay node and IP link failures, while satisfying strict delay requirements and capacity constraints (see Fig. 5.2).

5.6 Topology-based formulation: Problem P2

Our primary goal is to solve P1. However, to evaluate the average source packet loss rate \bar{r} , we have to specify not only the capacity and delay constraints, but also the packet loss model together with all loss parameters for every overlay node and IP link. This quickly gets very complicated and dependent on a myriad of heterogeneous elements.

For this reason, in this section we define a problem P2 that is based purely on the *topological* aspects of the system. P2 unifies all these elements under one homogenous framework, making it much easier to handle. Moreover, we formally link the two problems - we show that, under some conditions, P1 and P2 are equivalent.

5.6.1 Critical components $C^\lambda(v)$ and $C^\phi(v)$

Let us begin with the following definition:

Definition 6 (Critical components $C^\lambda(v)$ and $C^\phi(v)$) *We say that a network component $x \in E^\phi \cup V^\lambda \setminus \{v\}$ is critical for an overlay node $v \in V^\lambda$, if during a single and permanent failure of x , the source packets are not delivered at v within time $t_{max}(v)$. Denote by $C^\lambda(v) \subset V^\lambda \setminus \{v\}$ the set of all critical overlay nodes of v , and by $C^\phi(v) \subseteq E^\phi$ the set of all critical IP links of v .*

Consider the example in Fig. 5.4a. The sets of critical nodes of v and u are, respectively, $C^\lambda(v) = \{u\}$ and $C^\lambda(u) = \emptyset$, respectively. (Note that by definition, source s is not included in these sets.) Similarly, the sets of critical IP links of these nodes are $C^\phi(v) = \{a^\phi, b^\phi, c^\phi, d^\phi, e^\phi\}$ and $C^\phi(u) = \{a^\phi, b^\phi, c^\phi\}$, respectively. However, adding one more active link to the overlay topology as shown in Fig. 5.4b greatly changes the situation of node v . Indeed, now no single overlay node failure or physical link failure can disconnect v from s and thus $C^\lambda(v) = C^\phi(v) = \emptyset$.

5.6.2 Vulnerability \mathcal{W} , \mathcal{W}^λ and \mathcal{W}^ϕ

We can now define the metric that captures the system vulnerability to single physical and overlay failures.

Definition 7 (Vulnerability \mathcal{W}^λ , \mathcal{W}^ϕ and \mathcal{W}) *We define the logical Vulnerability \mathcal{W}^λ (to overlay node failures) and the physical Vulnerability \mathcal{W}^ϕ (to IP link failures) as*

$$\mathcal{W}^\lambda = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} |C^\lambda(v)|, \quad \mathcal{W}^\phi = \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} |C^\phi(v)|.$$

We average these two vulnerabilities in a single metric \mathcal{W} ,

$$\mathcal{W}(\eta) = \eta \cdot \mathcal{W}^\lambda + (1 - \eta) \cdot \mathcal{W}^\phi, \quad (5.1)$$

for some averaging coefficient $0 \leq \eta \leq 1$ that weights the relative contributions of \mathcal{W}^λ and \mathcal{W}^ϕ .

Vulnerabilities \mathcal{W}^λ and \mathcal{W}^ϕ are determined by the topology of the ALM. For example, in Fig. 5.4a, we have $\mathcal{W}^\lambda = 1/|V^\lambda|$ (contributed by node v) and $\mathcal{W}^\phi = (3 + 3 + 5)/|V^\lambda| = 11/|V^\lambda|$. Adding one active overlay link (as in Fig. 5.4b) creates an alternative path between s and v , which decreases

vulnerabilities to $\mathcal{W}^\lambda=0$ and $\mathcal{W}^\phi=6/|V^\lambda|$, respectively. This also hints that \mathcal{W}^λ and \mathcal{W}^ϕ are not independent.

Vulnerability metrics capture the *path diversity* in the system, which can be interpreted as follows:

$$\begin{aligned} \text{Small } \mathcal{W}^\lambda &\Leftrightarrow \text{High overlay-path diversity.} \\ \text{Small } \mathcal{W}^\phi &\Leftrightarrow \text{High IP-path diversity.} \end{aligned}$$

5.6.3 Problem P2

Vulnerability $\mathcal{W}(\eta)$ is designed to jointly capture the IP losses and overlay node failures, by weighting \mathcal{W}^λ and \mathcal{W}^ϕ with a parameter η . Therefore, we define the Problem P2 as follows:

P2: *Given capacity and delay constraints $\{b_{\max}(v), \gamma, t_{\max}(v)\}$ for $v \in V \cup \{s\}$, design an ALM system that minimizes Vulnerability $\mathcal{W}(\eta)$ for a given parameter η .*

5.6.4 Equivalence of P1 and P2

In order to link P1 and P2, we make two assumptions on the loss model, A1 and A2. They can be considered as first order approximations of the reality. First, as we have no prior knowledge of the failure probabilities of particular components, we assume that

A1: *Every overlay node $v \in V^\lambda$ fails with the same probability $0 \leq p^\lambda < 1$, independently of other elements. Every IP link $e^\phi \in E^\phi$ fails with the same probability $0 \leq p^\phi < 1$, independently of other network components.*

Second, we assume that

A2: *Failures are persistent, i.e., a source packet i and all its duplicates and derivatives observe the same state of every network component.*

This is a natural assumption for overlay node failures, as they are typically long-lasting (e.g., leaving the system). Moreover, A2 roughly captures the bursty nature of losses at IP links in today's Internet [108]. Indeed, A2 assumes ideal, 'all or none' type of bursts. This simplifies the aspects of time variability and allows us to focus explicitly on achieving high path diversity, which is our main goal.

Now we can state the following theorem:

Theorem 6 (Equivalence of P1 and P2)

Given Assumptions A1-A2, we have that

$$\bar{r} = \alpha \cdot \mathcal{W}(\eta) + O((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2), \quad (5.2)$$

for a system-specific constant

$$\alpha = (p^\lambda + p^\phi - 2p^\lambda p^\phi) \cdot (1 - p^\lambda)^{|V^\lambda|-1} (1 - p^\phi)^{|E^\phi|-1}$$

and the parameter η equal to

$$\eta = \frac{p^\lambda(1-p^\phi)}{p^\lambda + p^\phi - 2p^\lambda p^\phi}. \quad (5.3)$$

Proof: See Appendix.

As the overall failure probability in the Internet is very low (implying small p^ϕ) and the interactive ALMs are usually small, stable systems (implying small $|E^\phi|$, $|E^\lambda|$ and p^λ), the term $O((p^\phi|E^\phi|)^2 + (p^\lambda|V^\lambda|)^2)$ is negligible in practice. Indeed, this term captures the probability of having multiple failures; if we additionally assume that at most one failure occurs at a time, then (5.2) boils down to precisely $\bar{r} = \alpha' \cdot \mathcal{W}$.

An immediate corollary of Theorem 6 is that, given A1-A2, minimizing the average loss rate \bar{r} is (almost) equivalent to minimizing Vulnerability $\mathcal{W}(\eta)$ with η set as shown in (5.3). Therefore, from now on we focus explicitly on Vulnerability and on solving P2 rather than P1.

5.7 Protection techniques

So far we have defined a meaningful metric $\mathcal{W}(\eta)$ that we want to minimize, but we have not said how to achieve this in practice. In this section we discuss four protection techniques that allow us to introduce the path diversity and redundancy in the system, and, as a result, to optimize $\mathcal{W}(\eta)$. Please refer to Fig. 5.4 for examples of each technique.

5.7.1 SingleTree (Fig. 5.4a)

The first technique is only used as a reference point with strictly *no* redundancy ($\gamma = 0$), i.e., a directed tree $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$, $|E_1^\lambda| = |V^\lambda|$, rooted at the source s . This straightforward ALM solution was used e.g., in [74,76,77,79,80,85,89,93]. Check [94] for a review and comparison. Most of these papers propose some heuristics to minimize the average or maximal end-to-end propagation time, as this problem is in general NP-complete [80,93]. We refer to this basic technique as *SingleTree*.

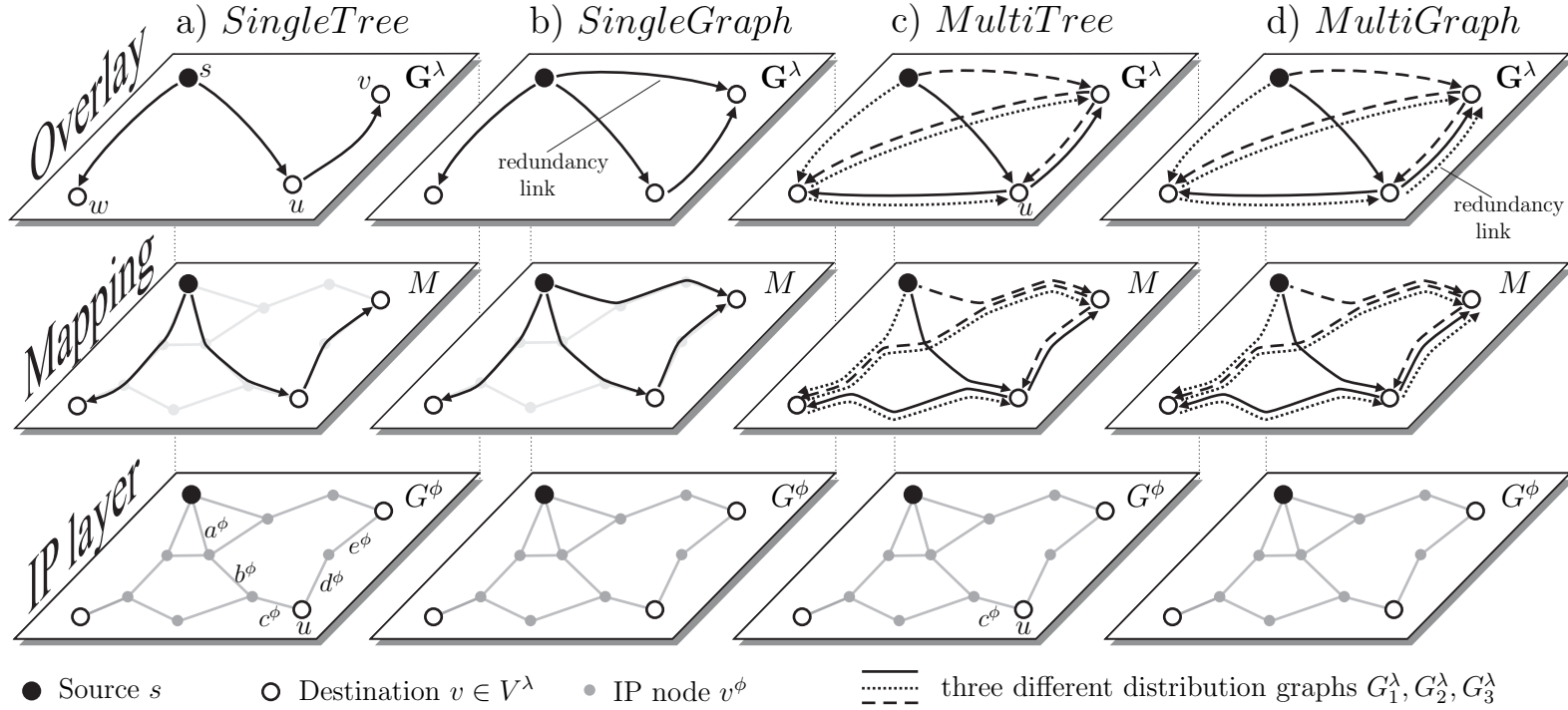


Figure 5.4: An illustration of four protection techniques in a system with one source and $|V^\lambda| = 3$ destinations. To simplify the presentation we set $t_{max} = \infty$. **(a) *SingleTree***: $\gamma = 0$ (no redundancy, reference point) and $\mathcal{W}^\lambda = \frac{1}{|V^\lambda|}$ and $\mathcal{W}^\phi = \frac{11}{|V^\lambda|}$. **(b) *SingleGraph***: $\gamma = (4 - 3)/3 = 1/3$ ($R = 1$ redundancy link), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{6}{|V^\lambda|}$. **(c) *MultiTree*** with FEC(3,2): $\gamma = (3 - 2)/2 = 1/2$ (only FEC redundancy), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{5}{|V^\lambda|}$. **(d) *MultiGraph*** with FEC(3,2): $\gamma = 1/2 + 1/9$ (FEC redundancy plus $R = 1$ redundancy link in one distribution graph), $\mathcal{W}^\lambda = 0$ and $\mathcal{W}^\phi = \frac{3}{|V^\lambda|}$.

5.7.2 SingleGraph (Fig. 5.4b)

The obvious problem with *SingleTree* is that it does not provide any proactive protection - any single failure always affects all nodes located downstream of the failing element. One way of increasing its resilience is to use a distribution graph $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$ with *more edges than necessary*, i.e., with $|E_1^\lambda| > |V^\lambda|$. All edges E_1^λ actively forward all the packets resulting in a number of alternative paths leading to destination nodes. The redundancy of this system is $\gamma = (|E_1^\lambda| - |V^\lambda|)/|V^\lambda| = R/|V^\lambda|$, where $R = |E_1^\lambda| - |V^\lambda|$ is the number of redundancy edges, i.e., the maximal number of edges without which the necessary traffic can be still delivered.

How can we construct a good graph G_1^λ ? One method is an incremental design starting from a *SingleTree* topology, to which a number of ‘cross-edges’ are added, as e.g., in PRM [97], TMesh [110], HBM [111] and in [112]. Instead, we propose to construct G_1^λ starting from scratch, which clearly gives us more freedom and potentially leads to a better performance. We call this general approach *SingleGraph*.

5.7.3 MultiTree (Fig. 5.4c)

Another technique to introduce redundancy in the system is coding. In particular, we consider a non-systematic Forward Error Correction $FEC(n, k)$ scheme where each source packet is encoded as one FEC block of n packets, called *FEC* packets. The total size of the FEC block is n/k times larger than the original source packet, so the introduced redundancy is $\gamma = (n - k)/k$. If k or more FEC packets are received within time $t_{max}(v)$ at destination v , then the original source packet is reconstructed and delivered; otherwise it is lost. In order to benefit from the available path diversity, we construct a set $\{G_1^\lambda, \dots, G_n^\lambda\}$ of n trees, and disseminate the i th FEC packet over the i th tree G_i^λ . We refer to this technique as *MultiTree*.

Multiple trees were first used together in ALM systems such as Coop-Net [81,82] and SplitStream [78]. These systems use Multiple Description Coding (MDC) instead of FEC. But they share the same general principle - exploiting path diversity by disseminating encoded packets over different trees.

5.7.4 MultiGraph (Fig. 5.4d)

Finally, we propose to combine *MultiTree* and *SingleGraph* by making at least one of the distribution graphs denser than a tree, i.e., $n > 1$ and

$|E_i^\lambda| > |V^\lambda|$ for at least one $i, 1 \leq i \leq n$.³ Under this technique, called *MultiGraph*, the number of edges needed to carry the necessary traffic (with no redundancy) is equal to $k \cdot |V^\lambda|$ (k trees). So every edge in the system carries fraction $1/(k \cdot |V^\lambda|)$ of the necessary traffic. Thus the system redundancy γ is

$$\gamma = \frac{\sum_{i=1}^n |E_i^\lambda|}{k \cdot |V^\lambda|} - 1 = \frac{n - k}{k} + \frac{R}{k \cdot |V^\lambda|},$$

where R is the total number of redundancy edges in all distribution graphs, $R = \sum_{i=1}^n (|E_i^\lambda| - |V^\lambda|)$.

To the best of our knowledge, no technique equivalent to *MultiGraph* has been studied to date. In the Appendix we demonstrate on a concrete example the following:

Observation 1 (Each protection technique may be best) *Consider a source s , a set V^λ of destinations and some fixed redundancy $\gamma_{max} > 0$. Depending on the underlying IP topology, each of the three protection techniques (*SingleGraph*, *MultiTree* and *MultiGraph*) may alone lead to the smallest vulnerability \mathcal{W}^ϕ .*

5.7.5 A common framework \mathbf{G}^λ

Let us now introduce a common notation that accommodates all the above redundancy techniques. By convention, we say that our system always uses FEC(n, k) with $n \geq k \geq 1$. Denote by $\mathbf{G}^\lambda = \{G_1^\lambda, \dots, G_n^\lambda\}$ a set of n dissemination overlay graphs $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda \subseteq E^\lambda)$, $|E_i^\lambda| \geq |V^\lambda|$ for every $1 \leq i \leq n$. We show in Table 5.2 that all four protection techniques described above are some special cases of the general framework \mathbf{G}^λ .

Feasibility. We say that the set \mathbf{G}^λ is *feasible* if both local and global capacity constraints are satisfied, and if in every graph G_i^λ , every node $v \in V^\lambda$ is reachable from s within time $t_{max}(v)$. Naturally, for every protection technique we require that it yields a feasible topology \mathbf{G}^λ .

Critical components. In order to evaluate Vulnerability $\mathcal{W}(\eta)$, we need a way to assess if a network component is critical in the system (see Def. 6,7). In our framework \mathbf{G}^λ , it is simple and well defined. To achieve this, we will say that a network component x (overlay node or IP link) is critical for a node $v \in V^\lambda$ in a separate distribution graph $G_i^\lambda \in \mathbf{G}^\lambda$ if every path from s to v in G_i^λ shorter than $t_{max}(v)$ traverses x . Now, a network component x is critical for $v \in V^\lambda$ in the entire system \mathbf{G}^λ if x is critical for

³Note that according to this definition neither *SingleGraph* nor *MultiTree* is a *MultiGraph*.

Technique	n and k	number of edges	redundancy γ
<i>SingleTree</i>	$n=k=1$	$ E_1^\lambda = V^\lambda $	0
<i>SingleGraph</i>	$n=k=1$	$ E_1^\lambda > V^\lambda $	$\frac{ E_1^\lambda - V^\lambda }{ V^\lambda }$
<i>MultiTree</i>	$n > k \geq 1$	$ E_i^\lambda = V^\lambda $ for all $1 \leq i \leq n$	$\frac{n-k}{k}$
<i>MultiGraph</i>	$n > k \geq 1$	$ E_i^\lambda > V^\lambda $ for at least one i	$\frac{\sum_{i=1}^n E_i^\lambda }{k \cdot V^\lambda } - 1$

Table 5.2: Four protection techniques within the \mathbf{G}^λ framework.

v in more than $n-k$ distribution graphs G_i^λ . Indeed, during the failure of such a component x , more than $n-k$ FEC packets do not reach v making the FEC recovery impossible.

For example, in Fig. 5.4c, the IP link c^ϕ is critical for node u in two distribution graphs: G_1^λ (plain lines) and G_2^λ (dotted lines). As $2 > n-k = 1$, we know that c^ϕ is critical for u in the entire system \mathbf{G}^λ , i.e., $c^\phi \in C^\phi(u)$.

5.8 Objective functions

Recall that our goal is solving the problem P2, i.e., finding a feasible topology \mathbf{G}^λ that minimizes the vulnerability $\mathcal{W}(\eta)$. To the best of our knowledge, this general problem has not been addressed to date. Therefore, there is no prior work that we can directly compare with. However, many works address simplified versions of P2, which amount to replace $\mathcal{W}(\eta)$ by another function X . We distinguish three general categories of these objective functions: RAND, 1-LAYER and 2-LAYER, as follows.

5.8.1 RAND (reference point)

Under RAND, we select the topology of \mathbf{G}^λ at random, only guaranteeing its feasibility. Other than that, no effort is made to minimize $\mathcal{W}(\eta)$. This simple approach was used together with *SingleGraph* e.g., in PRM [97] where the additional cross-links were randomly added to the distribution tree. In the context of *MultiTree* the examples are CoopNet [81], and the RMF scheme in [113]. RAND also serves as a reference point for other solutions.

5.8.2 1-LAYER (state of the art)

In contrast, 1-LAYER optimizes the overlay topology \mathbf{G}^λ , but *ignores the knowledge of the underlying IP layer*. Therefore, under 1-LAYER we directly minimize the logical vulnerability \mathcal{W}^λ only. This should result in some level

of IP path diversity too, but we have no direct control over the physical vulnerability \mathcal{W}^ϕ . The 1-LAYER objective function captures many existing solutions. Under *SingleGraph*, maximizing the overlay path diversity is one of the main goals in [111,112,114]. Similarly, for *MultiTree* a set of node-disjoint trees is constructed in SplitStream [78], and in [82,113]. Thus 1-LAYER corresponds to the state of the art, with the addition of a strict delay constraint that we consider here, contrary to the works mentioned above.

5.8.3 2-LAYER (our proposal)

Finally, under 2-LAYER we propose to take *both* the overlay and the IP layer into account. This additional information allows us to directly minimize the entire Vulnerability $\mathcal{W}(\eta)$ for any $0 \leq \eta \leq 1$. To the best of our knowledge no ALM technique that falls in this category has been proposed to date. One of the goals of this chapter is to study the gain of 2-LAYER over 1-LAYER.

To conclude, within a given protection technique (*SingleGraph*, *MultiTree*, *MultiGraph*), we minimize

$$X = \begin{cases} 0 & \text{under RAND} \\ \mathcal{W}^\lambda & \text{under 1-LAYER} \\ \mathcal{W} & \text{under 2-LAYER} \end{cases}, \quad (5.4)$$

subject to the feasibility of \mathbf{G}^λ .

5.9 Constructing a good topology \mathbf{G}^λ

In the previous two sections we specified our protection techniques and objective functions. The last step is to find the actual topology \mathbf{G}^λ that minimizes the objective function (5.4) under one of the protection techniques described in Section 5.7. Unfortunately, finding the optimal \mathbf{G}^λ is an NP-complete problem. Indeed, already finding a feasible topology \mathbf{G}^λ with $n=k=1$, $\gamma=0$ and fixed $b_{max}(v)$ for some t_{max} is equivalent to the ‘minimum maximum-latency degree-bounded directed spanning tree problem’ [80,115] shown to be NP-complete.

The problem complexity makes the computation of the optimal solutions untractable for more than a few nodes. Therefore, we use a *simulated annealing* heuristic to optimize the topology of \mathbf{G}^λ . The *input data* are the full graph $G^\lambda = (V^\lambda \cup \{s\}, E^\lambda)$ with propagation delays for every link, the mapping M , FEC parameters n, k , local and global capacity constraints and

maximal delays $t_{max}(v)$. As the *initial topology* of \mathbf{G}^λ we take a set of n different connected random graphs with the total number of edges equal to

$$\sum_{i=1}^n |E_i^\lambda| = \lfloor |V^\lambda| \cdot (1 + \gamma_{max} - \frac{n-k}{k}) \rfloor. \quad (5.5)$$

This guarantees that the global capacity constraint γ_{max} is satisfied and maximally exploited. At every iteration we add, delete or rewire one or more edges, always preserving (5.5). In particular, we allow for the following topology change operations on \mathbf{G}^λ :

1. ‘Rewire source’: in one of the distribution graphs G_i^λ change a randomly chosen overlay edge $e^\lambda = (v_1, w)$ into $e^\lambda \leftarrow (v_2, w)$, where v_2 is picked at random.
2. ‘Rewire target’: like ‘Rewire Source’, but changes $e^\lambda = (v, w_1)$ into $e^\lambda \leftarrow (v, w_2)$.
3. ‘Rewire pair’: change a randomly chosen overlay pair of edges $e_1^\lambda = (v_1, w_1)$ and $e_2^\lambda = (v_2, w_2)$ into $e_1^\lambda \leftarrow (v_1, w_2)$ and $e_2^\lambda \leftarrow (v_2, w_1)$.

The heuristic minimizes the value of X (defined in (5.4)) and returns the topology \mathbf{G}^λ corresponding to the smallest found X .

In the Appendix we describe two techniques we used to significantly speed-up the heuristic. First, we propose a Bellman-Ford-based algorithm that efficiently finds the critical components and calculates Vulnerability \mathcal{W} . Second, we give a useful theorem that allows us to evaluate the effect of a topology change without actually recalculating \mathcal{W} .

5.9.1 Lower-bounds on \mathcal{W}^λ and \mathcal{W}^ϕ

In order to evaluate the performance of this heuristic and get some insight into the problem, we derive a number of lower-bounds on vulnerabilities \mathcal{W}^λ and \mathcal{W}^ϕ . The first one is called *CompleteGraph*; it is a simple, general and rather conservative lower-bound on \mathcal{W}^ϕ . The remaining bounds are specifically dedicated to given protection techniques. In particular, we derive *SingleGraph/SingleTree* lower-bounds on \mathcal{W}^λ and \mathcal{W}^ϕ , and a *MultiTree* lower-bound on \mathcal{W}^ϕ .

As these lower bounds are tedious to derive, we moved their presentation to the Appendix. However, they lead us to important observations that we describe in the next section.

5.10 Simulation results

5.10.1 Data sets

We collected the all-to-all traceroutes between 800 nodes participating in the DIMES project [32]. These nodes are private users scattered around the world, who voluntarily run the DIMES measurement software in the background. In order to obtain a dense and representative data set, we kept only those nodes that were active throughout the entire experiment and have different geographical locations. We also discarded the unsuccessful traceroutes (that did not reach the destination or have some unknown hops) and excluded the access links from the remaining traceroutes. The resulting data set consists of 107 nodes and about 90% of all 107×106 possible traceroutes.

5.10.2 Setting t_{max}

The minimal propagation delays that the system can achieve strongly depend on the choice of source s and destinations V^λ . Therefore, fixing t_{max} across all simulation runs would be difficult to interpret. Instead, we decided to choose t_{max} for every set $V^\lambda \cup \{s\}$ separately, by allowing for slightly more time than strictly necessary. In particular, we use the CPT centralized heuristic [93] to find the (approximated) propagation delay t_{min} to the farthest destination in the delay optimized *SingleTree*. If not stated otherwise, we set $t_{max}(v) = t_{min} + 50ms$ for every destination v .

5.10.3 Vulnerability $\mathcal{W}(\eta)$ and topology \mathbf{G}^λ as a function of η .

Our first goal is to understand how different values of the weighting parameter η affect the optimized topology \mathbf{G}^λ . This study does not concern RAND and 1-LAYER, because, according to (5.4), these two objective functions are independent of η . In contrast, under 2-LAYER, the objective is to minimize the vulnerability $\mathcal{W}(\eta) = \eta\mathcal{W}^\lambda + (1-\eta)\mathcal{W}^\phi$, which clearly depends on η .

In Fig. 5.5 we study $\mathcal{W}(\eta)$ minimized by our heuristic, as a function of η . We also present the vulnerabilities \mathcal{W}^ϕ and \mathcal{W}^λ of the resulting topology \mathbf{G}^λ , which leads us to the first conclusion C1:

C1: *For a wide range $0 < \eta < 1$ (but not for $\eta = 0$ nor $\eta = 1$) under 2-LAYER, both \mathcal{W}^λ and \mathcal{W}^ϕ are minimized at the same time.*

To support C1, we consider three cases: $\eta = 0$, $\eta = 1$ and $0 < \eta < 1$. For $\eta = 0$, the equation (5.1) boils down to $\mathcal{W} = \mathcal{W}^\phi$, meaning that we maximize the IP path diversity only. Naturally, this results in the smallest value of

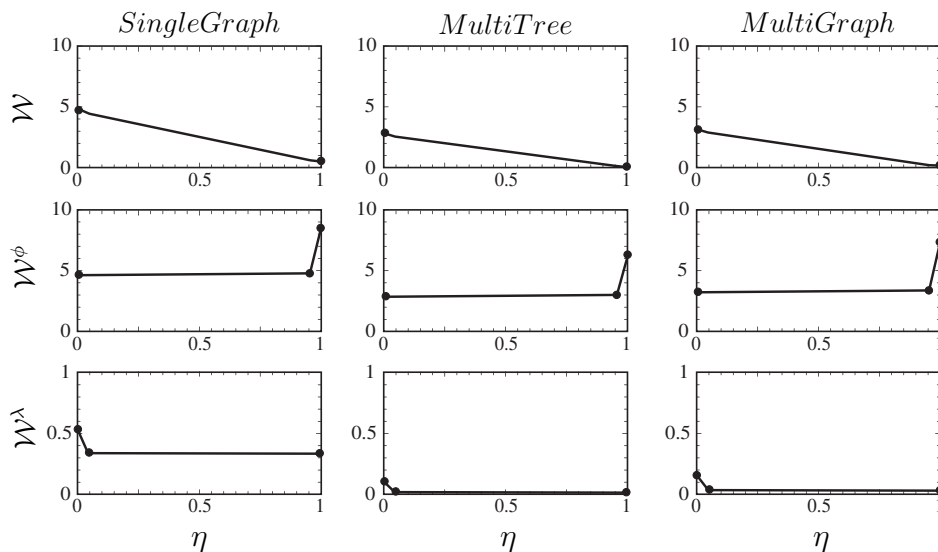


Figure 5.5: Vulnerability $\mathcal{W}(\eta)$ under 2-LAYER, as a function of the weighting parameter η (top). Below we show physical Vulnerability \mathcal{W}^ϕ (middle) and overlay Vulnerability \mathcal{W}^λ (bottom) of the resulting topology \mathbf{G}^λ . We simulated $\eta \in \{0, 0.05, 0.95, 1\}$, averaged over 1000 random subsets of the DIMES data set with $|V^\lambda|=10$, $\gamma_{max}=0.5$ and $b_{max}(v)=2$.

the physical vulnerability \mathcal{W}^ϕ . However, the overlay path diversity is not optimized: the overlay vulnerability \mathcal{W}^λ is significantly larger for $\eta=0$ than for $\eta>0$. Analogously, for $\eta=1$ we have $\mathcal{W}=\mathcal{W}^\lambda$, which results in minimal \mathcal{W}^λ , but clearly suboptimal \mathcal{W}^ϕ .

In contrast, for a wide range $0 < \eta < 1$, we observe a plateau of minimal values of both \mathcal{W}^λ and \mathcal{W}^ϕ . This means that the algorithm usually finds a topology \mathbf{G}^λ that minimizes both \mathcal{W}^λ and \mathcal{W}^ϕ at the same time, and thus suits any choice of η .

This is important, because in practice, we always have some non-zero probability of a node failure $p^\lambda > 0$ or IP loss $p^\phi > 0$, which, combined with (5.3), places us in the regime $0 < \eta < 1$. Moreover, the exact value of η (which might be difficult to estimate in practice) does not affect the resulting topology \mathbf{G}^λ . For these reasons, *in the remainder of this chapter we present results obtained for $\eta=0.5$, which represents well the entire range $0 < \eta < 1$.*

5.10.4 Detailed results for $0 < \eta < 1$

We present two types of results. First, in Fig. 5.6, under each of the four protection techniques we compare the vulnerabilities yielded by the heuristic, lower-bounds, and *the optimal topology \mathbf{G}^λ* . We find the optimal topology

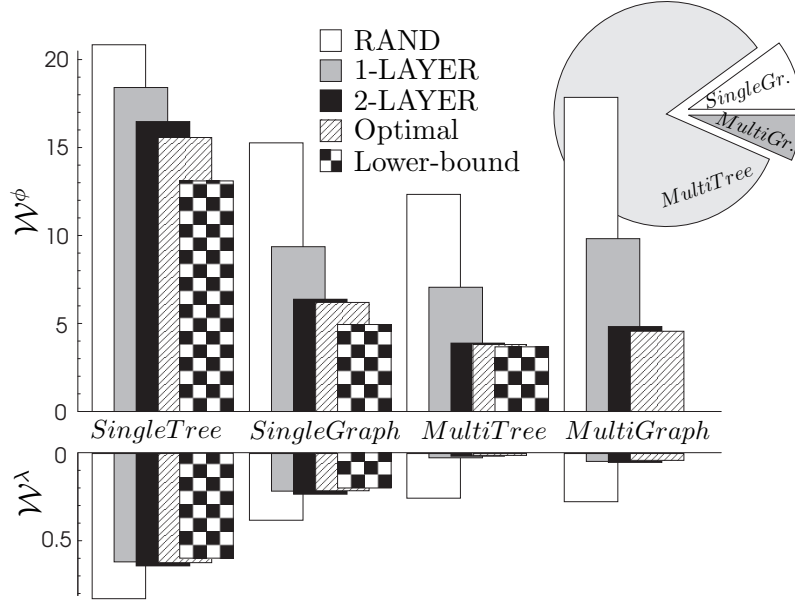


Figure 5.6: Vulnerability \mathcal{W}^ϕ (top) and \mathcal{W}^λ (bottom) under each of the four protection techniques, in a small system with $0 < \eta < 1$. We compare the optimal topology (‘Optimal’) with heuristic-based solutions (RAND, 1- and 2-LAYER) and the lower-bounds. We randomly choose $|V^\lambda \cup \{s\}| = 6$ nodes in the DIMES data set, and set $\gamma = 0.5$, $b_{max} = 2$. We use FEC(3,2) in *MultiTree* and FEC(4,3) in *MultiGraph*. The results are averaged over 200 different sets of nodes. The pie-chart shows how often a given protection technique results in the smallest \mathcal{W}^ϕ (out of all techniques). We discard here the cases where more than one technique leads to minimal \mathcal{W}^ϕ .

by running an exhaustive search with some obvious and more sophisticated pruning, which allowed us to fully study the overlays of up to 6 nodes in a reasonable time. Next, in Fig. 5.7 we present the heuristic results obtained for larger systems. We draw the following conclusions C2-C6.

C2: *1-LAYER outperforms RAND.*

Recall that 1-LAYER directly minimizes the overlay vulnerability \mathcal{W}^λ , whereas RAND guarantees the feasibility of \mathbf{G}^λ only. Therefore, it is not surprising that 1-LAYER results in values of \mathcal{W}^λ significantly smaller than RAND. In particular, under *MultiTree* we achieve $\mathcal{W}^\lambda \simeq 0$. Interestingly, the physical vulnerability \mathcal{W}^ϕ also benefits under 1-LAYER, resulting in a decrease of 10%-50% (depending on the protection technique).

C3: *2-LAYER outperforms 1-LAYER.*

More importantly, when moving from 1-LAYER to 2-LAYER we observe a further substantial decrease in \mathcal{W}^ϕ , ranging from 30% to 70%. This is

especially well pronounced in larger systems (see Fig. 5.7). In contrast, \mathcal{W}^λ remains practically unchanged. This is not surprising, because minimizing \mathcal{W}^λ is exactly the objective of 1-LAYER.

C4: *MultiTree is in general the best protection technique.*

Consider first the overlay vulnerability \mathcal{W}^λ . We observe that *MultiTree* and *MultiGraph* usually lead to $\mathcal{W}^\lambda = 0$ (with a slight advantage of *MultiTree*). Indeed, to achieve this, it is enough to guarantee that every destination serves as a relay in at most $n-k$ distribution graphs.

Moreover, according to the pie-chart in Fig. 5.6, *MultiTree* usually achieves the smallest physical vulnerability \mathcal{W}^ϕ out of all three protection techniques. To conclude, 2-LAYER *MultiTree* is a good and practical choice for protecting an interactive ALM.

C5: *Nearly optimal results can be achieved already by a simple heuristic.*

Recall that in Fig. 5.6 we show (among others) the optimal results. Interestingly, for all protection techniques the results obtained by our heuristic under 2-LAYER are nearly optimal. Similarly, in Fig. 5.7, most 2-LAYER results closely approach the corresponding lower-bounds. This means that we do not need any sophisticated and dedicated techniques to find a very good topology \mathbf{G}^λ . On the contrary, a simple general-purpose heuristic (simulated annealing in our case) is sufficient.

C6: *The IP-path diversity is limited by the system size $|V^\lambda|$ and redundancy γ_{max} .*

It is relatively easy to achieve a very high overlay-level path diversity. In particular, *MultiTree* usually reaches a zero logical vulnerability $\mathcal{W}^\lambda = 0$ (see the discussion in C4).

In contrast, there exist a number of factors that prevent us from achieving a high IP-level path diversity, i.e., that lower-bound the physical vulnerability \mathcal{W}^ϕ . First, \mathcal{W}^ϕ decreases with increasing system size $|V^\lambda|$. This is clearly visible in Fig. 5.8. We can also observe this tendency in more constrained settings - e.g., in Fig. 5.7 the curves decline with growing $|V^\lambda|$ (except *SingleGraph*). This phenomenon can be easily explained. With one source and one destination ($|V^\lambda| = 1$) we have no choice but to send the traffic directly to this destination, resulting in virtually no path diversity. But with growing $|V^\lambda|$, we obtain more and more means to exploit the underlying IP path diversity.

Second, the effective IP-level path diversity is limited by the maximal redundancy γ_{max} . In Fig. 5.8 we show *MultiTree* lower-bounds for three different values of γ_{max} . For example, for $\gamma = 0.2$ we have $\mathcal{W}^\phi \simeq 4$ even for large $|V^\lambda|$, which is much more than roughly $\mathcal{W}^\phi \simeq 0$ for the same setting with $\gamma = \infty$.

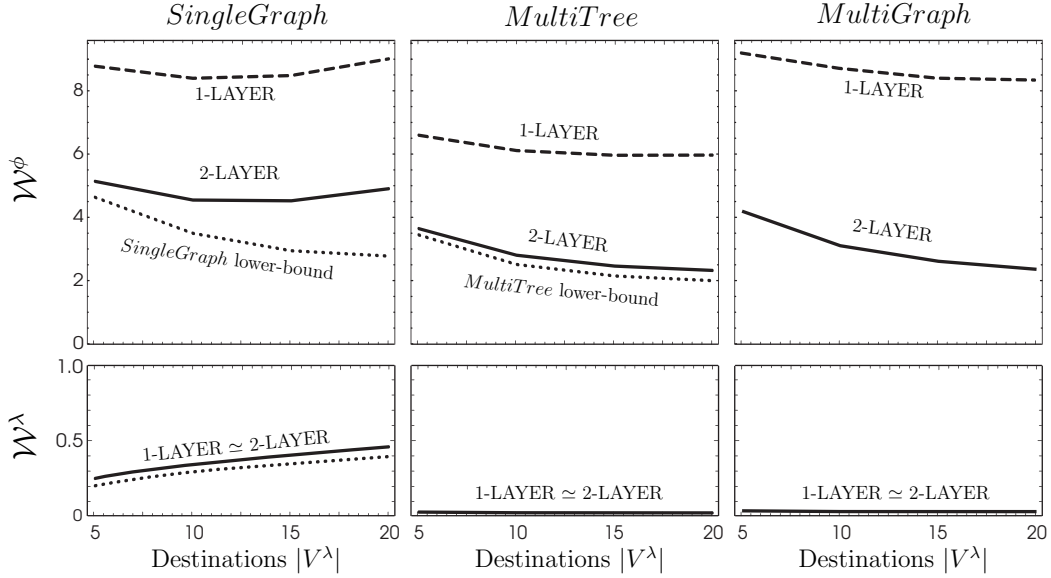


Figure 5.7: Vulnerability \mathcal{W}^ϕ and \mathcal{W}^λ in different settings for varying number of destination nodes $|V^\lambda| = 5 \dots 20$. We fix $\gamma_{max} = 0.5$, $b_{max} = 2$ and the regime $0 < \eta < 1$. We use three protection techniques: *SingleGraph*, *MultiTree* with FEC(3, 2), and *MultiGraph* with FEC(4, 3). For clarity, we do not show the *SingleTree* results, as they do not fit in the current scale. For the same reasons we do not show the RAND results for any of the techniques. The results are averaged over 1000 different sets of $|V^\lambda \cup \{s\}|$ nodes chosen at random from the DIMES data set.

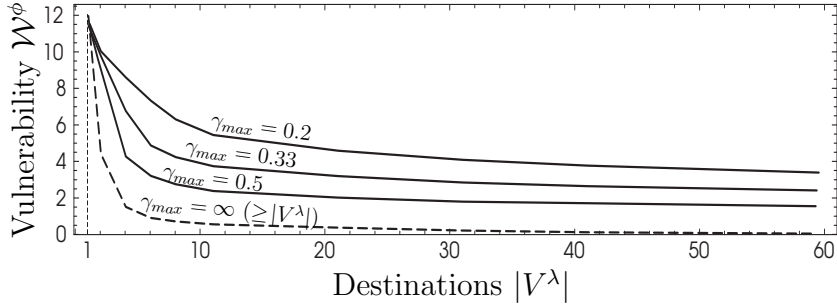


Figure 5.8: Minimal Vulnerability \mathcal{W}^ϕ as a function of system size $|V^\lambda|$, assuming $b_{max} = \infty (\geq |V^\lambda|)$ and $t_{max} = \infty$. The dashed curve is the *CompleteGraph* lower bound, i.e., a fully connected *SingleGraph* where all destinations actively forward the source packets to all other destinations. The remaining (plain) curves are *MultiTree* lower bounds for three different values of γ_{max} . The results are averaged over 1000 sets of overlay nodes.

5.11 Conclusion

In this chapter we studied the interactive single-source application-level multicast, as an example of a two-layer system. We found, that given the strin-

gent delay requirements and capacity constraints of the interactive ALM, it is (usually) still possible to create a system with a high path diversity at *both* the overlay and the IP layer. This results in a good protection against overlay node failures *and* IP losses. However, such a system should be carefully designed. It is not sufficient to maximize the overlay path diversity, as it does not automatically optimize the IP layer. Therefore, we should take into account not only the overlay-level information, but also the actual mapping of the overlay links on the IP layer. As a result, for real Internet topologies we reduce the system's physical Vulnerability \mathcal{W}^ϕ by typically 30%-70%, while keeping the logical Vulnerability \mathcal{W}^λ unchanged. Moreover, with the help of a set of lower-bounds, we showed that our results are nearly optimal. In a more general context, we have demonstrated that system size and maximal allowed redundancy are two important factors that naturally limit the available IP-path diversity. Last but not least, we have introduced a novel topology-based metric of Vulnerability \mathcal{W} , that not only captures the system's path diversity in a simple and meaningful way, but also can be formally linked with the average packet loss rate observed at the destinations.

APPENDIX

5.A Proofs

5.A.1 Proof of Theorem 6 [Equivalence of P1 and P2]

Denote by $\mathbb{P}(Y)$ the probability that all components (IP links and overlay nodes) in a set $Y \subseteq V^\lambda \cup E^\phi$ fail, and no other component fails. Recall that each overlay node fails independently with probability p^λ and each IP link fails independently with probability p^ϕ . Thus the probability of no failure is

$$\mathbb{P}(Y=\emptyset) = (1-p^\lambda)^{|V^\lambda|}(1-p^\phi)^{|E^\phi|} = (1-p^\lambda)(1-p^\phi)\beta,$$

where $\beta = (1-p^\lambda)^{|V^\lambda|-1}(1-p^\phi)^{|E^\phi|-1}$. The probabilities of a single failure of an overlay node v' and of an IP link e^ϕ are

$$\mathbb{P}(\{v'\}) = p^\lambda(1-p^\phi)\beta \quad \text{and} \quad \mathbb{P}(\{e^\phi\}) = p^\lambda(1-p^\lambda)\beta.$$

Hence the probability $\mathbb{P}(|Y|=1)$ of exactly one failure is

$$\mathbb{P}(|Y|=1) = \beta \left(|V^\lambda| p^\lambda (1-p^\phi) + |E^\phi| p^\phi (1-p^\lambda) \right).$$

Let $F_v(Y)$ be the indicator function returning 1 when a source packet is *not* delivered at node v within $t_{max}(v)$ due to the failure of Y , and 0 otherwise.

For instance, if exactly one IP link e^ϕ fails then $F_v(\{e^\phi\}) = 1_{\{e^\phi \in C^\phi(v)\}}$. Now we can write

$$\begin{aligned}
r(v) &= \mathbb{P}(\text{packet is not delivered at } v \text{ within } t_{max}(v)) = \\
&= \sum_{\text{all } Y \subseteq V^\lambda \cup E^\phi} F_v(Y) \mathbb{P}(Y) = \sum_{\{Y: |Y|=1\}} F_v(Y) \mathbb{P}(Y) + R_v = \\
&= \sum_{v' \in V^\lambda} F_v(\{v'\}) \mathbb{P}(\{v'\}) + \sum_{e^\phi \in E^\phi} F_v(\{e^\phi\}) \mathbb{P}(\{e^\phi\}) + R_v = \\
&= p^\lambda (1-p^\phi) \beta \sum_{v' \in V^\lambda} F_v(\{v'\}) + p^\phi (1-p^\lambda) \beta \sum_{e^\phi \in E^\phi} F_v(\{e^\phi\}) + R_v = \\
&= p^\lambda (1-p^\phi) \beta \cdot |C^\lambda(v)| + p^\phi (1-p^\lambda) \beta \cdot |C^\phi(v)| + R_v = \\
&= \alpha \left(\eta \cdot |C^\lambda(v)| + (1-\eta) \cdot |C^\phi(v)| \right) + R_v, \tag{5.6}
\end{aligned}$$

where $\alpha = (p^\lambda + p^\phi - 2p^\lambda p^\phi) \beta$ and $\eta = \frac{p^\lambda (1-p^\phi)}{p^\lambda + p^\phi - 2p^\lambda p^\phi}$.

The remainder R_v in (5.6) can be upper-bounded as follows:

$$\begin{aligned}
R_v &= \sum_{\{Y: |Y|>1\}} F_v(Y) \mathbb{P}(Y) \leq \sum_{\{Y: |Y|>1\}} \mathbb{P}(Y) = 1 - \mathbb{P}(|Y| \leq 1) = \\
&= 1 - \beta \left((1-p^\lambda)(1-p^\phi) + |V^\lambda| p^\lambda (1-p^\phi) + |E^\phi| p^\phi (1-p^\lambda) \right) = \\
&= 1 - \beta \left(1 + p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| + |E^\phi| - 1) \right) \leq \\
&\leq 1 - \beta \left(1 + p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| - 1) (|E^\phi| - 1) \right) = \\
&= 1 - \beta (1+Z) \leq 1 - (1-Z)(1+Z) = Z^2 = O\left((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2 \right), \tag{5.7}
\end{aligned}$$

where

$$Z = p^\phi (|E^\phi| - 1) + p^\lambda (|V^\lambda| - 1) - p^\phi p^\lambda (|V^\lambda| - 1) (|E^\phi| - 1),$$

and where we used $(1-p)^x \geq 1 - px$ to lower-bound β by

$$\beta \geq (1 - p^\lambda (|V^\lambda| - 1)) (1 - p^\phi (|E^\phi| - 1)) = 1 - Z.$$

Finally, combining (5.6) and (5.7) we obtain

$$\begin{aligned}
\bar{r} &= \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} r(v) = \\
&= \frac{\alpha}{|V^\lambda|} \left(\eta \sum_{v \in V^\lambda} |C^\lambda(v)| + (1-\eta) \sum_{v \in V^\lambda} |C^\phi(v)| \right) + \frac{1}{|V^\lambda|} \sum_{v \in V^\lambda} R_v = \\
&= \alpha \cdot \mathcal{W} + O\left((p^\phi |E^\phi|)^2 + (p^\lambda |V^\lambda|)^2 \right). \quad \blacksquare
\end{aligned}$$

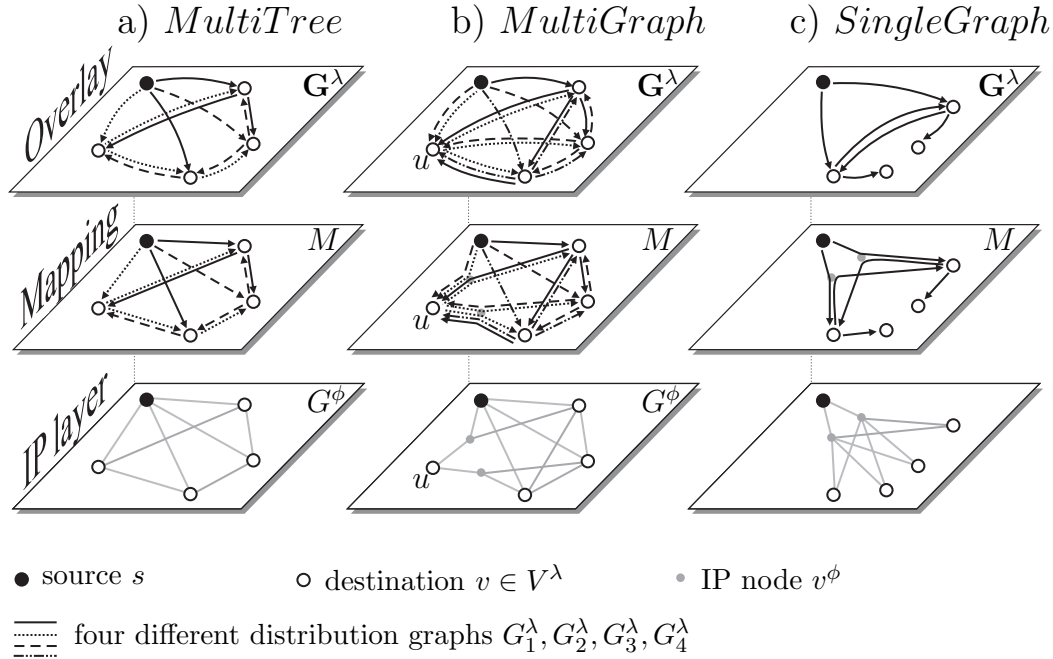


Figure 5.9: The absolute and relative effectiveness of the *SingleGraph*, *MultiTree* and *MultiGraph* strongly depends on the underlying IP topology. In this toy example we have one source s and $|V^\lambda| = 4$ destinations. For each of the three possible underlying IP topologies we find the optimal protection technique and topology \mathbf{G}^λ , assuming $\gamma = 0.5$, no local capacity constraints and $t_{max} = \infty$. These techniques are *MultiTree*, *MultiGraph* and *SingleGraph* for (a), (b) and (c), respectively.

5.A.2 Proof of Observation 1

First note that $\gamma = 0.5$ and $|V^\lambda| = 4$ allow us to use (i) *SingleGraph* with $|E_1^\lambda| = 6$ edges (i.e., 2 redundancy edges), (ii) *MultiTree* with FEC(3,2) or (iii) *MultiGraph* with FEC(4,3) and 2 redundancy edges. Values of n higher than 4 do not make sense because $|V^\lambda| = 4$.

Consider now the setting presented in Fig. 5.9. We will show that the three possible underlying IP topologies in (a), (b) and (c), the unique optimal protection techniques are *MultiTree*, *MultiGraph* and *SingleGraph*, respectively

For the IP topology as shown in Fig. 5.9a, *MultiTree* with FEC(3,2) and topology \mathbf{G}^λ as shown achieves $\mathcal{W}^\phi = 0$. In contrast, *SingleGraph* with two redundancy edges must leave two destinations unprotected (with in-degree equal to 1) resulting in $\mathcal{W}^\phi > 0$. Finally, under *MultiGraph* with FEC(4,3) every overlay node contributes to \mathcal{W}^ϕ with $\mathcal{W}^\phi(v) > 0$. With only

two redundancy links at our disposal we must leave at least two destinations with $\mathcal{W}^\phi(v) > 0$, resulting in $\mathcal{W}^\phi > 0$

Let us now move to Fig. 5.9b. For this IP topology *MultiGraph* with FEC(4,3) achieves $\mathcal{W}^\phi = 0$ with the presented topology (note two redundant edges in two of the trees). As before, *SingleGraph* is limited to $\mathcal{W}^\phi > 0$. And clearly, under *MultiTree* with FEC(3,2) we always have $\mathcal{W}^\phi(u) > 0$ and thus $\mathcal{W}^\phi > 0$.

Finally, in Fig. 5.9c, *SingleGraph* achieves $\mathcal{W}^\phi = 2$ with the presented \mathbf{G}^λ . In contrast, for every destination under *MultiTree* with FEC(3,2) we have $\mathcal{W}^\phi(v) = 2$ and thus $\mathcal{W}^\phi = 8$. *MultiGraph* with FEC(4,3) every destination v must contribute at least $\mathcal{W}^\phi(v) \geq 1$, unless reinforced with 2 or more redundancy edges. As we have only 2 redundancy edges at our disposal, *MultiGraph* results in $\mathcal{W}^\phi \geq 3$. ■

5.B Lower bounds

In this section we derive a number of lower-bounds on Vulnerability \mathcal{W}^ϕ .

5.B.1 Simple lower-bound on \mathcal{W}^ϕ

Denote by $E^{in}(v) \subseteq E^\lambda$ (resp., and $E^{out}(v) \subseteq E^\lambda$) the sets of all edges incoming to (resp., outgoing from) node $v \in V^\lambda \cup \{s\}$ in the complete graph G^λ . Our basic component in the construction of a lower-bound is the set $E_{acc}^\phi(v) \subseteq E^\phi$ of *effective access links* of node v , defined as

$$E_{acc}^\phi(v) = \begin{cases} \bigcap_{e^\lambda \in E^{out}(v)} M(e^\lambda) & \text{for } v = s \\ \bigcap_{e^\lambda \in E^{in}(v)} M(e^\lambda) & \text{for } v \in V^\lambda \end{cases}$$

In other words, $E_{acc}^\phi(v)$ is the set of IP links that are traversed by *every* IP path incoming to v (or outgoing from s if $v = s$). The sets $E_{acc}^\phi(v)$ depend on the choice of $V^\lambda \cup \{s\}$.

As any path from s to $v \in V^\lambda$ must always traverse all the effective access links of s and v , the minimal contribution $\mathcal{W}^\phi(v)$ to vulnerability \mathcal{W}^ϕ of each destination node v is

$$\mathcal{W}^\phi(v) \geq |E_{acc}^\phi(s)| + |E_{acc}^\phi(v)|, \quad (5.8)$$

which gives us the following simple lower-bound on Vulnerability \mathcal{W}^ϕ :

$$\mathcal{W}^\phi \geq |V^\lambda| \cdot |E_{acc}^\phi(s)| + \sum_{v \in V^\lambda} |E_{acc}^\phi(v)|. \quad (5.9)$$

5.B.2 CompleteGraph lower-bound on \mathcal{W}^ϕ

Another approach is to relax all capacity constraints, set $t_{max} = \infty$, and compute Vulnerability \mathcal{W}^ϕ under *SingleGraph* with $G_1^\lambda = G^\lambda$, i.e., the distribution (complete) graph that contains all the possible edges. Clearly, this globally lower-bounds \mathcal{W}^ϕ ; we refer to this as *CompleteGraph* lower-bound.

5.B.3 SingleGraph (and SingleTree) lower-bound on \mathcal{W}^ϕ

The bounds given above are general, but usually poor. In contrast, it is possible to derive bounds that are better, but restricted to the chosen protection technique. For *SingleGraph* we can prove the following:

Theorem 7 (*SingleGraph lower-bound for \mathcal{W}^ϕ*) *For SingleGraph protection technique we have*

$$\begin{aligned} \mathcal{W}^\phi \geq & |V^\lambda| \cdot |E_{acc}^\phi(s)| + \sum_{v \in V_a^\lambda} |E_{acc}^\phi(v)| + \\ & + \sum_{v \in V^\lambda \setminus V_a^\lambda} \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)|, \end{aligned} \quad (5.10)$$

where $V_a^\lambda \subseteq V^\lambda$ is the set of $a = \lfloor \gamma \cdot |V^\lambda| \rfloor$ destinations with the smallest gains $g(v) = \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)| - |E_{acc}^\phi(v)|$.

Proof of Theorem 7 [*SingleGraph* lower-bound for \mathcal{W}^ϕ]:

Let $\mathbf{G}^\lambda = \{G_1^\lambda\}$, $G_1^\lambda = (V^\lambda \cup \{s\}, E_1^\lambda \subseteq E^\lambda)$, $|E_1^\lambda| > |V^\lambda|$ be a system that uses the *SingleGraph* technique. Consider the minimal contribution $\mathcal{W}^\phi(v)$ of each destination node $v \in V^\lambda$ when the protection is provided by allowing for additional links. It depends on the in-degree $indeg_1(v)$ of v in G_1^λ . If $indeg_1(v) \geq 2$ then we reuse the bound (5.8). In contrast, if $indeg_1(v) = 1$ then we know that any path from s to v must always traverse all the effective access links of s and all IP links in the mapping of its incoming overlay edge e^λ . We can thus lower-bound $\mathcal{W}^\phi(v)$ by taking the edge e^λ with the shortest mapping $M(e^\lambda)$, i.e., if $indeg_1(v) = 1$ then

$$\mathcal{W}^\phi(v) \geq |E_{acc}^\phi(s)| + \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)|. \quad (5.11)$$

(Note that we exclude $E_{acc}^\phi(s)$ from $M(e^\lambda)$ to avoid a potential double counting of the effective access links of s .) The bound (5.11) is equal or higher than that in (5.8). Therefore, in order to find the minimal \mathcal{W}^ϕ we must consider the maximal possible number a of nodes with $indeg_1(v) > 1$. By the global constraint, there are at most $\lfloor \gamma \cdot |V^\lambda| \rfloor$ redundancy edges in the system. As each of them increases the in-degree of one destination by 1, the maximal number of destinations with $indeg_1(v) > 1$ is $a = \lfloor \gamma \cdot |V^\lambda| \rfloor$.

Which a destination nodes should we assign with $indeg_1(v) > 1$? Again, as we minimize \mathcal{W}^ϕ , we must match $indeg_1(v) > 1$ with a set $V_a^\lambda \subseteq V^\lambda$ of a destinations with the smallest gains

$$g(v) = \min_{e^\lambda \in E^{in}(v)} |M(e^\lambda) \setminus E_{acc}^\phi(s)| - |E_{acc}^\phi(v)|$$

due to the existence of more than one incoming edge.

To conclude, for the nodes in V_a^λ we use the bound (5.8) and for all other destinations we use (5.11); this results in the final formula (5.10) for the lower-bound on \mathcal{W}^ϕ given in Theorem 7 that we were to prove. ■

5.B.4 *MultiTree* lower-bound on \mathcal{W}^ϕ

Consider now *MultiTree*. This scenario has also some specific features that can be exploited when lower-bounding \mathcal{W}^ϕ . For example, in Fig. 5.4a (bottom), the set of effective access links of destination u is empty $E_{acc}^\phi(u) = \emptyset$. However, under *MultiTree* with FEC(3,2), for any topology \mathbf{G}^λ either e^ϕ or f^ϕ must be a critical IP link of u . Therefore in this case $\mathcal{W}^\phi(u) \geq |E_{acc}^\phi(s)| + 1$, which improves by 1 the bound in (5.8).

More generally, for each destination node v consider all the IP paths $\{M(e^\lambda) : e^\lambda \in E^{in}(v)\}$, i.e., mappings of all incoming overlay edges of v . Denote by $q(v, i)$ the total number of different IP links on these paths exactly i hops away from v . By convention, if i exceeds the length $\min_{e^\lambda \in E^{in}(v)} |M(e^\lambda)|$ of the shortest of the paths then $q(v, i) = \infty$. In Fig. 5.4 we have $q(u, 1) = |\{e^\phi, f^\phi\}| = 2$ and $q(u, 2) = 3$. For source s the term $q(s, i)$ is defined analogously, except that now we consider $E^{out}(s)$, i.e., all overlay edges in E^λ outgoing from s . Now we can state the following.

Theorem 8 (*MultiTree* lower-bound for \mathcal{W}^ϕ) *For MultiTree protection technique with FEC(n,k) we have*

$$\mathcal{W}^\phi \geq \sum_{v \in V^\lambda} \sum_{i=1}^{\lfloor \frac{1}{2} |M(s,v)| \rfloor} \mathbf{1}_{\{\lfloor n/q(s,i) \rfloor > n-k\}} + \mathbf{1}_{\{\lfloor n/q(v,i) \rfloor > n-k\}}. \quad (5.12)$$

Proof of Theorem 8 [*MultiTree* lower-bound for \mathcal{W}^ϕ]:

Let $\mathbf{G}^\lambda = \{G_1^\lambda, \dots, G_n^\lambda\}$, $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda)$, $|E_i^\lambda| = |V^\lambda|$ be a system that uses the *MultiTree* redundancy technique. There are $q(v, i)$ different IP links on all the $|V^\lambda|$ IP paths incoming to v and i hops away from v . At the same time, there are exactly n overlay edges in the system \mathbf{G}^λ that lead to v . Therefore, at least one IP link e^ϕ of the $q(v, i)$ links must be traversed by at least $\lceil \frac{n}{q(v, i)} \rceil$ of these overlay edges. Thus a single failure of e^ϕ results in a loss of at least $\lceil \frac{n}{q(v, i)} \rceil$ FEC packets. So if $\lceil \frac{n}{q(v, i)} \rceil > n - k$ then the original source packet cannot be recovered and this i th hop becomes critical for v and $\mathcal{W}^\phi(v)$ is incremented by 1. Therefore, $\mathcal{W}^\phi(v)$ is not smaller than $\sum_{i=1}^{\infty} 1_{\{\lceil n/q(v, i) \rceil > n-k\}}$. At the same time all critical elements of the source s are also critical for v , so $\mathcal{W}^\phi(v) \geq \sum_{i=1}^{\infty} 1_{\{\lceil n/q(s, i) \rceil > n-k\}}$. These two sums can be combined, but we need to guarantee that no critical element is counted twice - on the sides of both v and s . We achieve this by running these sums up to at most half of the length of the direct path $M(s, v)$, i.e., to $\lfloor \frac{1}{2} |M(s, v)| \rfloor$. This, together with the convention that $q(v, i) = \infty$ for all i larger than the shortest route incoming to v , allows us to write

$$\mathcal{W}^\phi(v) \geq \sum_{i=1}^{\lfloor \frac{1}{2} |M(s, v)| \rfloor} 1_{\{\lceil n/q(s, i) \rceil > n-k\}} + 1_{\{\lceil n/q(v, i) \rceil > n-k\}} \quad (5.13)$$

We obtain (5.12) by summing (5.13) for all destination nodes $v \in V^\lambda$, which finishes the prove of Theorem 8. ■

5.B.5 *SingleGraph* (and *SingleTree*) lower-bound on \mathcal{W}^λ

Let us now consider the lower-bound on logical Vulnerability \mathcal{W}^λ . Interestingly, it is relatively easy to achieve $\mathcal{W}^\lambda = 0$; this is straightforward for *MultiTree* with $t_{max} = \infty$. This implies that there exist no simple general lower-bound on \mathcal{W}^λ . However, the *SingleGraph* (and *SingleTree*) technique is less effective and usually results in $\mathcal{W}^\lambda > 0$ that can be lower-bounded as follows:

Theorem 9 (*SingleGraph* lower-bound for \mathcal{W}^λ) *For SingleGraph we have*

$$\mathcal{W}^\lambda \geq \frac{1}{|V|} (|V| - \lfloor \gamma_{max} |V| \rfloor - \lfloor b_{max}(s) \rfloor). \quad (5.14)$$

Proof

There are at least $|V| - \lfloor \gamma_{max} |V| \rfloor$ destinations with only one incoming overlay link in G_1^λ . For each of these destinations its direct parent is a critical node,

unless this parent is the source s itself. As s can have at most $\lfloor b_{max}(s) \rfloor$ children, there are at least $(|V| - \lfloor \gamma_{max} |V| \rfloor - \lfloor b_{max}(s) \rfloor)$ destinations with $|C^\lambda(v)| \geq 1$. This implies (5.14). ■

Obviously, Theorem 9 holds also for $\gamma_{max} = 0$, i.e., for the *SingleTree* technique.

5.C Speeding-up the heuristic

In this section we describe two techniques we used to significantly speed-up the heuristic. First, we propose a Bellman-Ford-based algorithm that efficiently finds the critical components and calculates Vulnerability \mathcal{W}^ϕ . Second, we give a useful theorem that allows us to evaluate the effect of a topology change without actually recalculating \mathcal{W}^ϕ .

5.C.1 Computation of critical elements for $G_i^\lambda \in \mathbf{G}^\lambda$

A naive approach to find the critical IP links in $G_i^\lambda \subseteq \mathbf{G}^\lambda$ is to consider each IP link $e^\phi \in E^\phi$ (separately), hide all the overlay edges using e^ϕ and run Dijkstra on the resulting graph. This amounts to running Dijkstra $|E^\phi|$ times for every $G_i^\lambda \subseteq \mathbf{G}^\lambda$. Typically $|E^\phi|$ is large and this approach is quite costly, because the heuristic involves numerous evaluations of the objective function (i.e., vulnerability).

Fortunately, there exist better solutions. We propose an algorithm that computes all critical elements in a given distribution graph $G_i^\lambda = (V^\lambda \cup \{s\}, E_i^\lambda) \in \mathbf{G}^\lambda$. It is inspired by the Bellman-Ford distance vector routing protocol, but there are three main differences. First, we consider one traffic source s only and many destinations V^λ . Second, every node $v \in V^\lambda$ learns the minimal distance *from* s *to* v , not from v to s . Third, and most important, our algorithm investigates some additional properties of the discovered paths, i.e., the network elements that these paths avoid. Clearly, if no path from s to v shorter than t_{max} avoids an IP link e^ϕ then e^ϕ is a critical element of v .

Basic version

Every logical node $v \in V^\lambda \cup \{s\}$ stores a function $t_v : E^\phi \mapsto \mathcal{R}$ defined for all IP links $e^\phi \in E^\phi$. $t_v(e^\phi)$ is the delay on the (so far) shortest path $p_{s,v}^\lambda$ that avoids e^ϕ . If no such path exists (or was found so far) then $t_v(e^\phi) = \infty$. Initially $t_v(e^\phi) = \infty$ for all $e^\phi \in E^\phi$ and $v \in V^\lambda$. In contrast $t_s(e^\phi) = 0$ for every e^ϕ . There are $|V^\lambda|$ iterations. At every iteration, for every overlay edge

$e = (v, w) \in E_i^\lambda$ we update update the function t_w for target node w . This means that at j th iteration we find all paths that use j hops. The very basic algorithm is as follows:

```

1 forall  $e^\phi \in E^\phi$  do
2   forall  $v \in V^\lambda$  do
3      $t_v(e^\phi) \leftarrow \infty$ 
4      $t_s(e^\phi) = 0$ 
5   for  $j \leftarrow 1$  to  $|V^\lambda|$  do
6     forall  $e = (v, w) \in E_i^\lambda$  do
7       forall  $e^\phi \notin M(e)$  do
8         if  $t_w(e^\phi) > t_v(e^\phi) + \text{delay}(e)$  then
9            $t_w(e^\phi) \leftarrow t_v(e^\phi) + \text{delay}(e)$ 

```

More efficient implementation

In practice there are many possible improvements. First, nodes do not have to store full tables. Let initially the domain $DOM(t_v)$ of t_v be empty and change dynamically as follows. Let $t_v(e^\phi) \leftarrow t_{new}$ add e^ϕ to $DOM(t_v)$ (if it was not there yet) and set $t_v(e^\phi)$ to value t_{new} . Let $undefine(t_v(e^\phi))$ delete e^ϕ from $DOM(t_v)$. Finally, let '*' be a default element that covers all elements currently not in $DOM(t_v)$, i.e., $t_v(e^\phi)$ returns $t_v(*)$ for all $e^\phi \notin DOM(t_v)$. Second, in line 6 we may take only those edges whose source nodes were updated at the previous iteration. Moreover, if there are no such edges then we can terminate the algorithm. Third, every value of delay larger than t_{max} can be interpreted as ∞ . With these three improvements the algorithm performs much faster and can be stated as follows:

```

forall  $v \in V^\lambda$  do
   $t_v(*) \leftarrow \infty$ 
   $j \leftarrow 0, t_s(*) \leftarrow 0$  {Iteration  $j = 0$ }
  for  $j \leftarrow 1$  to  $|V^\lambda|$  do
     $E'_i \leftarrow \{(v, w) \in E_i^\lambda : v \text{ was updated at iteration } j-1\}$ 
    if  $E'_i = \emptyset$  then return
    forall  $e = (v, w) \in E'_i$  do
      if  $t_w(e^\phi) \leq t_v(e^\phi) + \text{delay}(e)$  forall  $e^\phi \in \text{DOM}(t_w)$ 
        then continue {Speed-up}
      forall  $e^\phi \in M(e)$  do
         $t_w(e^\phi) \leftarrow t_w(e^\phi)$ 
      forall  $e^\phi \in \text{DOM}(t_v) \cup \text{DOM}(t_w) \setminus M(e) \setminus \{*\}$ 
        do  $t_w(e^\phi) \leftarrow \min(t_w(e^\phi), t_v(e^\phi) + \text{delay}(e))$ 
       $t_w(*) \leftarrow \min(t_w(*), t_v(*) + \text{delay}(e))$ 
      forall  $e^\phi \in \text{DOM}(t_w)$  do
        if  $t_w(e^\phi) = t_w(*)$  then  $\text{undefine}(t_w(e^\phi))$ 

```

5.C.2 Useful theoretical result

Second, based on the knowledge of the critical element in the current distribution graph G_i^λ , it is often possible to evaluate the effect of a topology change without actually recalculating vulnerability, as follows. Denote by $t_v(e^\phi)$ the delay on the shortest path in G_i^λ from s to v that avoids e^ϕ . If no such path exists then $t_v(e^\phi) = \infty$. (Finding $t_v(e^\phi)$ does not introduce any overhead because it must be implicitly computed during the computation of \mathcal{W}^ϕ .) We can prove that:

Theorem 10 (Effects of edge change on critical elements)

Deleting an overlay link $e^\lambda = (v, w) \in E_i^\lambda$ will not change the configuration of critical elements in G_i^λ if for all $e^\phi \notin M(v, w)$ we have $t_v(e^\phi) + t_{e^\lambda} > t_w(e^\phi)$ or $t_v(e^\phi) = \infty$. And conversely, adding a new overlay link $e^\lambda = (v, w) \notin E_i^\lambda$ may decrease the number of critical elements in G_i^λ only if there exists $e^\phi \notin M(v, w)$ such that $t_v(e^\phi) + t_{e^\lambda} < t_w(e^\phi)$.

Proof of Theorem 10 [Effects of edge change on critical elements]:

First we prove the part of this theorem that speaks about deleting the edge $e = (v, w)$. Assume that for all $e^\phi \notin M(v, w)$ we have $t_v(e^\phi) + t_e > t_w(e^\phi)$ or $t_v(e^\phi) = \infty$. Consider the shortest-delay overlay path $SP(s, w, e^\phi)$ from source s to w that avoids e^ϕ in its mapping. For every $e^\phi \in E^\phi$ we will show that $e \notin SP(s, w, e^\phi)$. We can distinguish three cases:

1. $e^\phi \in M(e)$. Then trivially $e \notin SP(s, w, e^\phi)$, because by definition $SP(s, w, e^\phi)$ avoids e^ϕ .
2. $e^\phi \notin M(e)$ and $t_v(e^\phi) = \infty$. The latter says that every path from s to v traverses e^ϕ . So every overlay path that contains $e = (v, w)$ must traverse e^ϕ too, yielding $e \notin SP(s, w, e^\phi)$ again.
3. $e^\phi \notin M(e)$ and $t_v(e^\phi) + t_e > t_w(e^\phi)$. In this case there exists paths from s to w that contain e and avoid e^ϕ . But, because $t_v(e^\phi) + t_e > t_w(e^\phi)$, none of these paths can be shortest, and thus $e \notin SP(s, w, e^\phi)$.

As for every e^ϕ we have shown that $e \notin SP(s, w, e^\phi)$, the deletion of e will not change the shortest paths from s to w avoiding e^ϕ . So the critical elements of w do not change. This can be easily extended to every other node $u \in V^\lambda$, because if e belonged to $SP(s, v, e^\phi)$ then just after e the path $SP(s, v, e^\phi)$ would traverse node w . But we know that $e \notin SP(s, w, e^\phi)$, which leads to contradiction.

The second part of this theorem (adding an edge) we can easily prove by contradiction, reusing the arguments above.

■

Chapter 6

Conclusion

Two-layer networks are a challenging environment for the enforcement of an effective failure protection. This is because a single failure at the physical layer propagates to the logical layer where it may result in multiple logical failures. In this thesis, we have found a confirmation of this problem by studying real-life two-layer systems.

Fortunately, often a careful system design can minimize this problem. We have demonstrated it on the example of IP/WDM networks and interactive ALM systems. Although these two settings have very different objectives, constraints and requirements, in both cases we were able to significantly improve their protection to typical failures. In order to achieve this, we have proposed dedicated algorithms and techniques that lead to a failure-proof design of mapping (under IP/WDM), or of the logical layer (under ALM).

Please, refer to Table 6.1 for a more detailed summary of contributions made in this thesis.

Future Work

The work in this thesis can be a basis for a number of research directions. We classify these directions in two groups, as follows.

Specific problems

First, we have addressed in this thesis many specific problems in concrete settings. The dedicated solutions that we propose can probably be further improved, or extended.

For example, the SMART algorithm as presented in Chapter 4 addresses only the basic version of the routing and wavelength assignment problem in the IP/WDM setting. There are numerous real-life constraints that should be

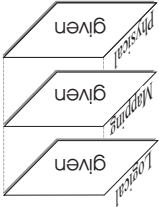
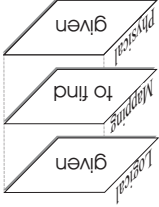
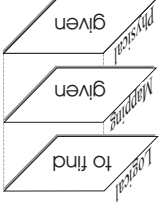
Pictogram	Object and goal of study	Achievements and conclusions
 <p>Large-scale, real-life</p>	<p>In Chapter 3 we study existing large-scale systems, such as:</p> <ul style="list-style-type: none"> • Railway networks, • Gnutella (Internet), • The human brain topology. <p>Our goal is to gain more understanding of the problem of failure propagation and multiplication.</p>	<p>We develop techniques to construct two-layer data sets:</p> <ul style="list-style-type: none"> • An algorithm to infer the physical infrastructure of a public transportation system based only on its timetables (Railway), • A methodology to create a network of long distance connections in the human brain, based on a diffusion MRI data. <p>We draw the following conclusions:</p> <ul style="list-style-type: none"> • Attacks are much more harmful than errors. • The logical graph is affected much faster than the physical graph. • A heterogeneous topology graph decreases system's robustness.
 <p>IP/WDM</p>	<p>In Chapter 4, we consider the IP/WDM networks, where a fixed IP graph is mapped onto a fixed mesh of physical fibers. Our goal is to design a <i>survivable mapping</i>, i.e., such that the logical topology remains connected after any single physical failure.</p>	<ul style="list-style-type: none"> • We introduce a new concept of <i>piecewise survivable mapping</i>, and show a number of properties of a piecewise survivable mapping. • We propose the SMART algorithm, which allows us to formally prove that a survivable mapping does or does not exist. • SMART helps us find and strengthen vulnerable areas in the system. • Moreover, SMART finds a survivable mapping much faster (often by orders of magnitude) than the other approaches. • We cover the failures of links, nodes, spans and double-links.
 <p>ATM/IP</p>	<p>In Chapter 5, we consider delay-constrained Application-Level Multicast (ALM) systems, where the ALM graph is mapped onto the IP topology. Our goal is to proactively protect the system against overlay node failures and against IP packet losses.</p>	<ul style="list-style-type: none"> • We define a topology-based Vulnerability metric W, and prove that W is equivalent to the average packet loss rate. • As W drops with the amount of IP- and application-level path diversity, we propose a framework to increase this diversity in the system. • Within this framework, we optimize the structure of ALM, which reduces the effective loss rate of real Internet topologies by typically 30%-70%. We also show that our approach is nearly optimal. • Finally, we study factors that naturally limit the available IP-path diversity, such as the system size and maximal allowed delay.

Table 6.1: Summary of contributions of this thesis.

additionally taken into account, such as propagation delays or the availability and performance of wavelength converters at the nodes. SMART is an elegant and powerful technique, and thus it can become a basic building block of more sophisticated solutions that address these real-life constraints. Indeed, SMART became a starting point for the works in [116–118].

Similarly, our interactive ALM framework, although quite general, does not cover all possible proactive failure protection techniques. One of the potentially attractive alternatives to adding new links and using FEC is network coding.

Another interesting problem, yet not addressed in this thesis, might be the study of vulnerability of the entire Internet. Indeed, it seems reasonable to view the Internet as a global mesh of application links mapped onto the IP layer, that in turn, is mapped onto the global network of optical fibers. This may lead us to very interesting and possibly scary conclusions. For example, there are only a few tens of submarine fibers spanning Europe and North America, with even fewer landing points (physical locations where cables enter the ocean). Consequently, relatively few fiber cuts (attack) may disconnect the two continents, causing a great damage to the global Internet.

General results

At a more general level, this thesis can be an inspiration for any other field where the two-layer issues arise. For example, our theoretical results, such as the three core theorems of Chapter 4, are relatively general and can be freely applied elsewhere.

This is a good motivation for additional theoretical studies to further improve our understanding of two-layer systems. Indeed, in Chapter 3 we studied experimentally the robustness of a system where the topologies at the two layers are generated with some well-known generators (see ‘ER on ER’ and ‘BA on ER’ data sets). It would be very interesting to rigorously bind the parameters of these topologies with the robustness of the resulting system.

Appendices

Appendix A

Exploiting Path Propagation Time Differences in Multipath Transmission with FEC

“And Now for Something Completely Different”

The best of Monty Python’s Flying Circus, 1971

In this chapter we present one of (numerous) side products of this thesis. It is a set of results related to ‘Multipath Transmission with FEC’. Although our work on this topic was originally inspired by the Overlay/IP design problem, it does not fall in the ‘two-layer’ core of this thesis, and is therefore presented in the Appendix.

A.1 Introduction

We consider a transmission of a delay-sensitive data stream from a single source to a single destination. How to improve the reliability of this transmission? Traditional ARQ (Automatic Repeat-reQuest) mechanisms often cannot be used, as they impose additional and usually unacceptable delays of at least one RTT (Round Trip Time). A more applicable technique is to introduce some type of redundancy, e.g., Forward Error Correction (FEC). Clearly, due to the delay constraints, a FEC block must be of limited length [119]. This, in turn, makes it inefficient against *bursty packet losses* [119] - the predominant type of losses in today’s Internet [108]. A good solution to this problem is to assign the FEC packets to *multiple paths* spanning the source and the destination [99,100,120–125]. An illustration of a multipath FEC system is presented in Fig. A.1. Theoretically, the multiple

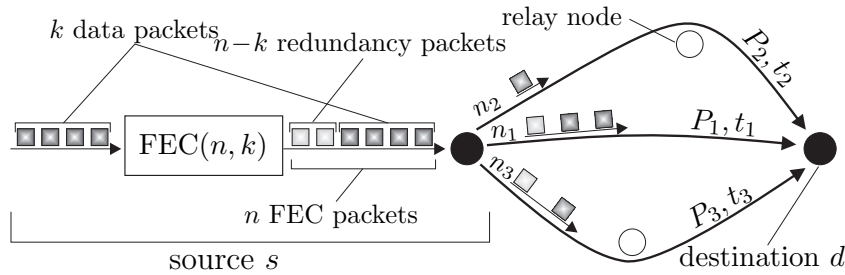


Figure A.1: Illustration of a multipath system with $R = 3$ paths P_1, P_2, P_3 between source s and destination d . t_1, t_2, t_3 are the corresponding path propagation times. k data packets are complemented with $n - k$ redundancy packets, and the resulting n FEC packets are split onto the three paths using the rates n_1, n_2 and n_3 , respectively.

paths could be constructed with the help of source routing, but this technique is not yet fully available in the Internet. A more practical alternative is the usage of overlay relay nodes that forward the traffic (as in Fig. A.1). If the resulting paths are statistically independent, which is especially likely for multi-homed hosts, then the loss bursts get averaged out and FEC regains effectiveness. Similar performance benefits due to multipath were also observed in the context of Multiple Description Coding [98].

When designing a system that splits a FEC block across multiple paths, we have to (1) select some paths out of all candidates, (2) assign the transmission rates to these paths, and (3) schedule the packets. The previous studies proposed techniques to solve (1-3) as a function of the statistical loss properties of the paths [99,100,121].

However, there are other important parameters affecting the performance of the multipath FEC system. In particular, in this chapter we show that the propagation times on the available multiple paths often significantly differ. These differences, in turn, can be exploited to improve the system reliability. Below, we explain and motivate our approach on concrete examples and measurements.

A.1.1 Propagation times on direct and indirect paths may differ significantly

In Fig. A.2 we study the path propagation time differences in the real-life Internet. The measurements were collected by running all-to-all traceroutes between 326 nodes in DIMES [32]. These nodes are usually private hosts located at different sites around the world. (We obtained similar results for

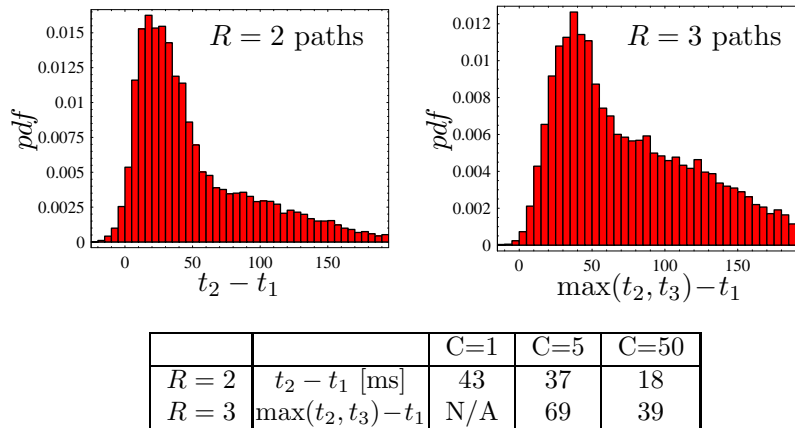


Figure A.2: The difference between propagation times on the direct path P_1 and the best indirect paths P_2 and P_3 . We present the results for $R = 2$ (two paths: one direct and one indirect) and $R = 3$ (the direct path and two indirect paths). The histograms (top) show the distribution of propagation time differences for $C = 5$ available candidate indirect paths. The table (bottom) shows the medians of these distributions for $C = 1, 5$ and 50 . The averages (not shown) are systematically higher than the medians.

measurements on PlanetLab [126].)

For each source-destination pair we construct a set of R paths. We always include the *direct* path P_1 with propagation time t_1 . Each of the remaining $R - 1$ paths is *indirect*, i.e., it uses some overlay relay node to forward the traffic. We choose uniformly at random a number C of candidate relay nodes among the remaining 324 DIMES nodes. This results in C candidate indirect paths. From them we select the $R - 1$ indirect paths following the intuitive selection procedure given in [99]. For $R = 2$ paths we choose the indirect candidate path that is the most disjoint (in terms of IP links) with the direct path P_1 . Clearly, this minimizes the loss correlation between P_1 and P_2 . If there are more paths that achieve the minimal IP overlap, then the one with the smallest propagation time is kept. For $R > 2$ we proceed similarly, except that now we consider the aggregated values of IP overlap and propagation time, i.e., summed over all $R(R - 1)/2$ possible path pairs.

According to Fig. A.2, for $R = 2$, the best indirect path P_2 has propagation times larger by typically $0 \dots 75ms$ than the direct path P_1 (see $t_2 - t_1$ in top-left histogram). This difference gets larger for a smaller number of candidates C (table at the bottom).

Moreover, the path propagation time differences grow significantly with the number of paths R used in the system. As shown in Fig. A.2, already for $R = 3$ the medians of the distributions are roughly doubled compared to

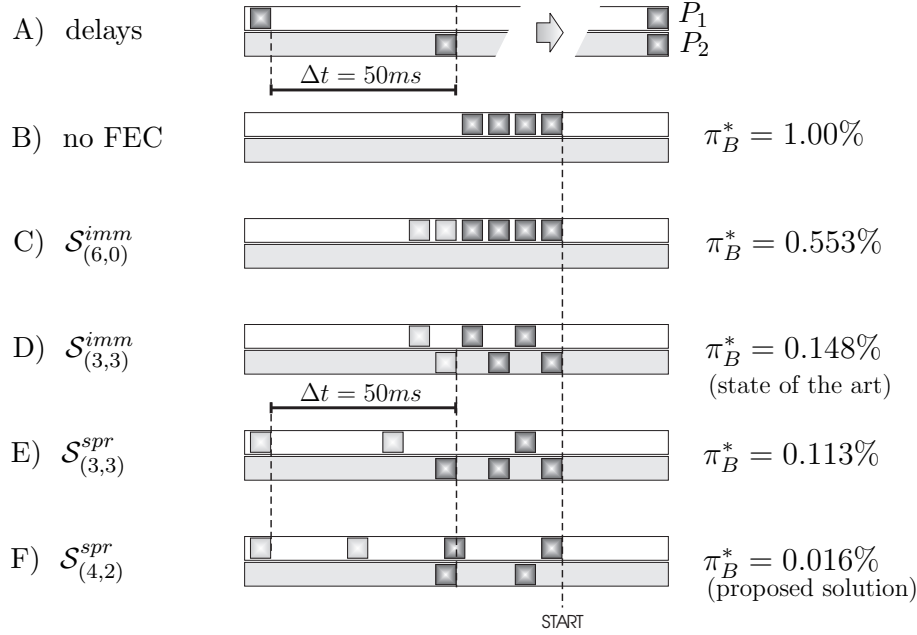


Figure A.3: Illustration of various packet schedules and their performance measured in the effective loss rate π_B^* . We use two independent paths P_1 and P_2 with identical failure distributions. The data packets are generated at the source every $T = 5ms$ and coded with FEC(6,4). (A) The path propagation time t_2 on path P_2 is $\Delta t = t_2 - t_1 = 50ms$ larger than the path propagation time t_1 on path P_1 . (B) No FEC, single path, the packets are sent at times 0, 5, 10, 15ms. (C) FEC on P_1 only, packets are sent as soon as they are generated, i.e., we use the ‘Immediate’ schedule \mathcal{S}^{imm} . (D) Packets alternate between P_1 and P_2 with equal rates $n_1 = n_2 = 3$, as in [99,100]. The total FEC block delay resulting from this scheme serves as a maximal FEC block delay in the following scenarios. (E) Packets alternate between P_1 and P_2 with equal rates, but the three packets sent on P_1 are maximally spread. (F) Packets are split between P_1 and P_2 with optimal rates $n_1 = 4, n_2 = 2$, maximally spread.

$R=2$, and typically P_1 is faster than the slower of the two indirect paths by $\max(t_2, t_3) - t_1 \simeq 0 \dots 150ms$.

We conclude that in the real-life Internet the propagation time differences on multiple paths between a source-destination pair are significant, typically reaching several tens of milliseconds.

A.1.2 The differences in propagation times can be exploited by a multipath FEC system

We propose to exploit these path propagation time differences when designing a multipath FEC system. Our solution is easy to implement and can bring significant performance gains. Let us take a concrete example described in Fig. A.3. Assume that there exist two paths between the source and the destination, the direct path P_1 , and an indirect path P_2 created by employing another peer that works as a relay. Let $t_1 = 100ms$ and $t_2 = 150ms$ be the propagation delays on P_1 and P_2 , respectively. So the path propagation time difference is $\Delta t = 50ms$ (Fig. A.3A). Let the two paths be lossy with the same loss rate 1% and the same average loss burst length of $10ms$, but independent. The data packets are generated at the source every $T = 5ms$. If no form of packet protection is used, then the data packet loss rate observed at the destination, or the *effective loss rate*, is $\pi_B^* = 1\%$ (B). Assume now that we use FEC(6,4) to protect the packets. If we send all packets on P_1 with inter-packet times T , then the effective loss rate after FEC decoding is $\pi_B^* = 0.553\%$ (C). Following [99,100], we now split the packets equally between P_1 and P_2 , which decreases π_B^* to 0.148% (D). This solution represents the state of the art in minimizing π_B^* . Note that now the last FEC packet on path P_2 reaches the destination $t_{\text{FEC}} = t_2 + 4 \cdot T = 170$ milliseconds after the generation of the first FEC packet at source. In other words, in this case the application using multipath FEC must accept the (maximal) delay equal to t_{FEC} . However, we can achieve far better results still respecting this delay constraint. For instance, we can appropriately increase the packet-spacing on P_1 and achieve $\pi_B^* = 0.113\%$ (E). Finally, we get even more significant improvement by sending 4 packets on P_1 and 2 packets P_2 , i.e., by applying *unequal* rates on the paths (F). This results in $\pi_B^* = 0.016\%$, which is almost one order of magnitude smaller than (D).

In other words, we exploit the differences in path propagation times by spreading the packets in time, such that the maximal allowed delay is respected. The gain over the state of the art measured in the effective loss rate π_B^* may be very significant (here 0.016% vs 0.148%, i.e., almost ten-fold). Moreover, some results may seem counterintuitive. For instance, it may be better to use only one path than to use two (un-spaced) paths. It also turns out that even if the loss distributions on the paths are the same, the optimal rates assigned to these paths are not necessarily equal.

A.1.3 Organization of this chapter

The remainder of this chapter is organized as follows. In Section A.2 we fully specify our model, which allows us to precisely state the problem we are solving. Next, in Section A.3 we derive exact analytical expressions for the effective loss rate π_B^* under multipath FEC and an arbitrary schedule. In Section A.4 we describe the ‘Immediate’ schedule representing the state of the art, and propose a ‘Spread’ schedule that exploits the differences in path propagation times. In Section A.5 we evaluate our solution analytically, by simulations and by trace-driven simulations fed with real-life Internet traces. In Section A.6 we discuss the related work. Finally we conclude the chapter.

A.2 Model and problem statement

The packets, called *data* packets, are generated at source s , with constant inter-arrival time T . There exist R paths between sender s and destination d , with the propagation delays t_1, \dots, t_R , respectively.

A.2.1 Path losses

The paths are assumed to be independent. We model bursty losses on each path by the popular two-state Gilbert model. Its basic version is a Discrete Time Markov Chain (DTMC), and captures the loss correlations due to queuing on bottleneck links, when the path is sampled at some constant rate (e.g., $1/T$). However, as we vary the sampling rates, DTMC is not sufficient. Indeed, on the same path we experience much higher loss burstiness under the packet interval of $T = 5ms$ than of $T = 100ms$ [119]. For this reason we use the continuous-time version of the Gilbert model [96,121] that naturally accommodates different sampling rates. It is a two-state stationary Continuous Time Markov Chain (CTMC) $\{X_r(t)\}$. The state $X_r(t)$ at time t assumes one of the two values: G (‘good’) or B (‘bad’). If a packet is sent at time t and $X_r(t) = G$ then the packet is transmitted; if $X_r(t) = B$ then the packet is lost.

We denote by $\pi_G^{(r)}$ and $\pi_B^{(r)}$ the stationary probabilities that the r th path is good or bad, respectively. Similarly, let $\mu_G^{(r)}$ and $\mu_B^{(r)}$ be the transition rates from G to B and from B to G , respectively. In this chapter we use two meaningful, system-dependent parameters to specify the CTMC packet loss model:

- the average loss rate $\pi_B^{(r)}$, and
- the average loss burst length (in seconds) $1/\mu_B^{(r)}$.

\mathbb{P}, \mathbb{E}	probability, expected value
s	source node
d	destination node
T	(constant) interval between two consecutive data packets at source s
R	number of independent paths between source s and destination d
P_r	r th path
t_r	propagation delay on P_r
$\pi_B^{(r)}, 1/\mu_B^{(r)}$	the average loss rate and loss burst length on path P_r
$n, k, (n-k)$	the number of FEC, data, and redundancy packets in a FEC block, respectively
n_r	number of FEC packets assigned to P_r (rate of path P_r)
k_r	number of data packets assigned to path P_r
T_r	(constant) spacing of the n_r packets on path P_r
F, D	number of lost FEC and data packets before FEC recovery
π_B^*	effective loss rate, i.e., the expected fraction of lost data packets at the destination after the FEC recovery
t_{FEC}	FEC block transmission time, i.e., the time between the generation of the first FEC packet at source s and the scheduled delivery of the latest FEC packet at destination d
$\mathcal{S}=(\mathcal{T}, \mathcal{R})$	packet scheduling: The i th packet in a FEC block is sent at time $\mathcal{T}(i)$ over path $\mathcal{R}(i)$

Table A.1: Basic notation used in this chapter.

All other parameters can be easily derived from these two, because

$$\pi_G^{(r)} = \frac{\mu_B^{(r)}}{\mu_G^{(r)} + \mu_B^{(r)}} \quad \text{and} \quad \pi_B^{(r)} = \frac{\mu_G^{(r)}}{\mu_G^{(r)} + \mu_B^{(r)}}. \quad (\text{A.1})$$

A.2.2 Multipath FEC

We use a $\text{FEC}(n, k)$ scheme to protect the data packets against losses (see Fig. A.1). This means that k data packets (not necessarily consecutive) are encoded as one FEC block of n packets, called *FEC* packets. In particular, as in [100,121,122,127,128], we consider a *systematic*¹ FEC, i.e., a scheme where the first k packets are the k data packets (unchanged). The remaining $n - k$ packets, called *redundancy* packets, carry the redundancy information. The destination uses the redundancy packets to recover some of the lost data packets as follows. Let F be the number of lost FEC packets and let D be the number of lost data packets of a FEC block, both before the FEC recovery

¹The non-systematic FEC is easier to handle, but also less efficient. We show its analysis in [11].

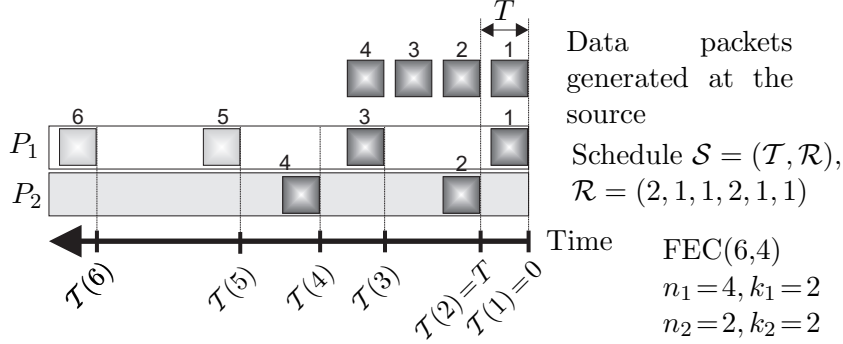


Figure A.4: An illustration of a schedule $\mathcal{S} = (\mathcal{T}, \mathcal{R})$ on $R=2$ paths with FEC(6,4). Four data packets numbered 1-4 are generated at the source at equal intervals T ; the first one specifies time $t = 0$. The $n - k = 2$ redundancy packets are numbered 5 and 6. According to the schedule $\mathcal{S} = (\mathcal{T}, \mathcal{R})$, the i th FEC packet is sent at time $\mathcal{T}(i) \geq 0$ over path $\mathcal{R}(i)$.

(note that D contributes to F). If $F \leq n - k$ then all the n FEC packets and hence all the k data packets are recovered. In contrast, if $F > n - k$, then no FEC recovery is possible and D data packets are lost.

A.2.3 Packet scheduling

Finally, the packets are sent according to some *schedule* that defines *when* and *on which path* each FEC packet is sent. More precisely, we denote by $\mathcal{S} = (\mathcal{T}, \mathcal{R})$ the schedule of packets in a FEC block, where \mathcal{T} and \mathcal{R} are vectors of length n . The i th FEC packet is sent at time $\mathcal{T}(i)$ over path $\mathcal{R}(i)$, as shown in Fig. A.4. The time is counted from the generation (at the source) of the first data packet of the FEC block. Denote by t_{FEC} the *FEC block transmission time*, i.e., the time between the generation of the first FEC packet at source s and the scheduled delivery of the latest FEC packet at destination d . Given a schedule \mathcal{S} , t_{FEC} can be easily computed as

$$t_{\text{FEC}} = \max_{1 \leq i \leq n} \left(\mathcal{T}(i) + t_{\mathcal{R}(i)} \right). \quad (\text{A.2})$$

For a given schedule, t_{FEC} can be interpreted as the total delay imposed by the multipath FEC system on the delay-sensitive application using it. Indeed, if the first packet of a FEC block is lost and needs to be reconstructed by FEC, then we have to wait up to t_{FEC} until the destination is reached by the other FEC packets necessary for the reconstruction of the lost packet. In practice, however, a constraint is likely to come from the delay-constrained application itself, as the maximal acceptable delay t_{FEC} . In this case our goal

is to design a good schedule respecting this constraint, which is the approach used in this chapter.

The schedule also implicitly defines the *rate* n_r of path P_r , i.e., the number of FEC packets sent on P_r . Similarly, let k_r be the number of data packets among the n_r packets sent on P_r . Clearly, $\sum_r n_r = n$ and $\sum_r k_r = k$.

A.2.4 Effective loss rate π_B^* and problem statement

Our ultimate goal is to send a stream of data packets over (possibly multiple) lossy channels in a way that minimizes the losses observed at the destination, given a maximal value for t_{FEC} . Therefore, we adopt a natural performance metric called *effective loss rate* π_B^* . It is defined as the expected fraction of lost data packets observed at the destination d after an attempt of FEC decoding. Now the problem can be stated as follows:

Given the path loss properties ($\pi_B^{(r)}$, $1/\mu_B^{(r)}$ and t_r for every path P_r), the FEC parameters (n and k) and maximal FEC block transmission time t_{FEC} , find the schedule \mathcal{S} that minimizes the effective loss rate π_B^ .*

We approach this problem in two steps. First, in Section A.3 we derive an exact analytical formula for the effective loss rate π_B^* for a given schedule \mathcal{S} . Second, in Section A.4 we introduce a schedule that exploits the differences in path propagation times and outperforms the schedules proposed to date.

A.3 Exact analytical derivation of the effective loss rate π_B^*

In order to design a good schedule we must be able to evaluate it. In this section we derive the exact analytical expression for the effective loss rate π_B^* for a given schedule \mathcal{S} . We consider two cases. First, we derive π_B^* for an arbitrary schedule \mathcal{S} . The resulting formula is simple but computationally expensive and untractable for larger sizes n of the FEC block. Next, we derive π_B^* assuming that on each path separately the packets are evenly spaced. This constraint is compatible with the schedule we propose later and results in a computationally lighter formula for π_B^* .

A.3.1 The effective loss rate π_B^* for an arbitrary schedule

Let c be a n -tuple representing a particular failure configuration; c_i , $1 \leq i \leq n$, takes the value G (resp., B) if i th FEC packet is transmitted (resp., lost). By

considering all possible failure configurations c we can compute the effective loss rate π_B^* for a given schedule \mathcal{S} as follows:

$$\pi_B^* = \frac{1}{k} \sum_{\text{all } c} D(c) \cdot \mathbb{P}(c), \quad (\text{A.3})$$

where $D(c)$ is the number of lost data packets (after the FEC recovery) for a given failure configuration c . For a systematic FEC(n, k) we have

$$D(c) = \begin{cases} 0 & \text{if } \sum_{i=1}^n 1_{\{c_i=B\}} \leq n - k \\ \sum_{i=1}^k 1_{\{c_i=B\}} & \text{otherwise.} \end{cases}$$

In order to compute the probability $\mathbb{P}(c)$ of a failure configuration c , we consider the R paths separately, as follows. Denote by $\mathcal{T}^{(r)}$ the vector of length n_r with departure times of packets scheduled by \mathcal{S} on path P_r . Similarly, let $c^{(r)}$ be an n_r -element vector with the failure configuration on path P_r defined by c . As the R paths are independent, the probability $\mathbb{P}(c)$ is

$$\mathbb{P}(c) = \prod_{r=1}^R \mathbb{P}(c^{(r)}), \quad (\text{A.4})$$

where $\mathbb{P}(c^{(r)})$ is the probability of a failure configuration $c^{(r)}$ on path P_r . The derivation of $\mathbb{P}(c^{(r)})$ for the Continuous Time Markov Chain loss model is straightforward. Indeed, denote by $p_{i,j}^{(r)}(\tau)$ the probability of transition from state i to state j on path P_r in time τ , i.e.,

$$p_{i,j}^{(r)}(\tau) = \mathbb{P}[X_r(\tau) = j | X_r(0) = i].$$

From classical Markov Chain analysis we have:

$$\begin{aligned} p_{G,G}^{(r)}(\tau) &= \pi_G^{(r)} + \pi_B^{(r)} \cdot \alpha \\ p_{G,B}^{(r)}(\tau) &= \pi_B^{(r)} - \pi_B^{(r)} \cdot \alpha \\ p_{B,G}^{(r)}(\tau) &= \pi_G^{(r)} - \pi_G^{(r)} \cdot \alpha \\ p_{B,B}^{(r)}(\tau) &= \pi_B^{(r)} + \pi_G^{(r)} \cdot \alpha \end{aligned} \quad (\text{A.5})$$

where $\alpha = \exp(-(\mu_G^{(r)} + \mu_B^{(r)})\tau)$. Now $\mathbb{P}(c^{(r)})$ can be easily computed. For example, for $c^{(r)} = GBB$ we have

$$\mathbb{P}(c^{(r)} = GBB) = \pi_G^{(r)} \cdot p_{G,B}^{(r)}(\tau_1) \cdot p_{B,B}^{(r)}(\tau_2),$$

where τ_i is the time interval between the i th and $(i+1)$ th FEC packet scheduled by \mathcal{S} on path P_r , i.e., $\tau_i = \mathcal{T}_{i+1}^{(r)} - \mathcal{T}_i^{(r)}$. More generally,

$$\mathbb{P}(c^{(r)}) = \pi_{c_1}^{(r)} \prod_{i=1}^{n_r-1} p_{c_i^{(r)}, c_{i+1}^{(r)}}^{(r)}(\mathcal{T}_{i+1}^{(r)} - \mathcal{T}_i^{(r)}). \quad (\text{A.6})$$

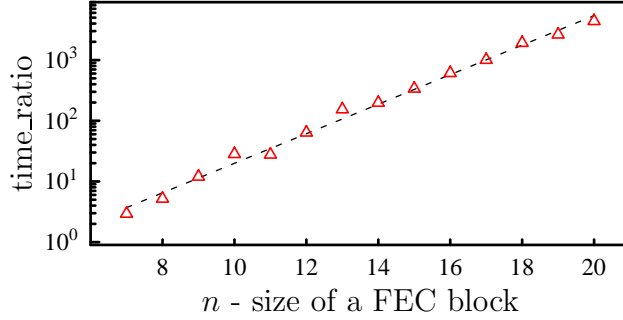


Figure A.5: The time complexity of the effective loss rate π_B^* under an arbitrary schedule (A.3.1) vs. the even-spaced schedule (A.3.2): time_ratio is the runtime of Eq. (A.7) divided by the runtime of Eq. (A.11). Here we use FEC($n, 0.7n$) on two identical paths.

Finally, we plug (A.6) and (A.4) to (A.3), to obtain

$$\pi_B^* = \frac{1}{k} \sum_{\text{all } c} D(c) \prod_{r=1}^R \pi_{c_1}^{(r)} \prod_{i=1}^{n_r-1} p_{c_i^{(r)}, c_{i+1}^{(r)}}^{(r)} (\mathcal{T}_{i+1}^{(r)} - \mathcal{T}_i^{(r)}). \quad (\text{A.7})$$

A.3.2 The effective loss rate π_B^* for even spacing on paths

Equation (A.7) allows us to compute the effective loss rate π_B^* for any schedule \mathcal{S} . However, evaluating (A.7) is computationally expensive because the main sum is over all 2^n failure configurations. Thus it can be applied to relatively small n only. Fortunately, we can significantly reduce the computation complexity by assuming that on each path P_r separately the packets are *evenly spaced*, i.e., for all $1 \leq i \leq n_r-1$ the intervals $\mathcal{T}_{i+1}^{(r)} - \mathcal{T}_i^{(r)}$ are the same and equal to a constant that we denote by T_r . Indeed, this constraint leads us to a formulation of π_B^* (below) that may take orders of magnitude less time to solve than (A.7), as shown in Fig. A.5.

In order to compute π_B^* under the even-spacing case, we look closer at the packets lost on every path separately. Denote by F_r and D_r the number of FEC and data packets lost on path P_r , respectively (both before FEC recovery). Now we can rewrite the total number of lost FEC packets as $F = \sum_r F_r$ and the total number of lost data packets as $D = \sum_r D_r$. This decomposition leads us to the following derivation of π_B^* :

$$\begin{aligned}
\pi_B^* &= \frac{1}{k} \sum_{j=n-k+1}^n \mathbb{P}(F = j) \cdot \mathbb{E}[D|F = j] = \\
&= \frac{1}{k} \sum_{j=n-k+1}^n \sum_{\substack{0 \leq j_1, \dots, j_R \leq j \\ j_1 + \dots + j_R = j}} \mathbb{P}(F_1=j_1, \dots, F_R=j_R) \cdot \mathbb{E}[D|F_1=j_1, \dots, F_R=j_R] = \\
&= \frac{1}{k} \sum_{j=n-k+1}^n \sum_{\substack{0 \leq j_1, \dots, j_R \leq j \\ j_1 + \dots + j_R = j}} \left(\prod_{r=1}^R \mathbb{P}(F_r = j_r) \cdot \sum_{r=1}^R \mathbb{E}[D_r|F_r = j_r] \right) \quad (\text{A.8})
\end{aligned}$$

According to Equation (A.8), in order to evaluate π_B^* , for every path P_r separately we need to calculate two components: (i) the probability $\mathbb{P}(F_r = j_r)$ that j_r FEC packets are lost, and (ii) the expected number $\mathbb{E}[D_r|F_r = j_r]$ of lost data packets given that j_r FEC packets were lost. We achieve this by applying an approach similar to the one used in [127] in the context of a single path FEC, as follows.

Let us first derive $\mathbb{P}(F_r = j_r)$. We consider a path P_r and a set of all n_r FEC packets sent on P_r with equal packet interval T_r . Denote by $[\binom{a}{b}]$ the event that any b out of a consecutive packets are lost.² We allow for concatenation of events, e.g., $G[\binom{a}{b}]$ (resp., $[\binom{a}{b}]B$) means that any b out of a consecutive packets are lost and that this block is preceded by a good packet (resp., followed by a bad packet). We can now compute $\mathbb{P}(F_r = j_r)$ by conditioning on the state of the first packet that conforms the packet loss stationary distribution:

$$\begin{aligned}
\mathbb{P}(F_r = j_r) &= \mathbb{P}(G[\binom{n_r-1}{j_r}]) + \mathbb{P}(B[\binom{n_r-1}{j_r-1}]) = \\
&= \pi_G^{(r)} \cdot \mathbb{P}([\binom{n_r-1}{j_r}] | G) + \pi_B^{(r)} \cdot \mathbb{P}([\binom{n_r-1}{j_r-1}] | B), \quad (\text{A.9})
\end{aligned}$$

where $\mathbb{P}([\binom{a}{b}] | q)$, $q \in \{G, B\}$, is the probability that any b out of a consecutive packets are lost given that this block is preceded by a packet in state q . Although no general closed form of $\mathbb{P}([\binom{a}{b}] | q)$ is known, it can be calculated by the recursive approach first proposed in [129] and extended e.g. in [121,127]. Indeed,

$$\begin{aligned}
\mathbb{P}([\binom{a}{b}] | B) &= R(b+1, a+1) \\
\mathbb{P}([\binom{a}{b}] | G) &= S(b+1, b-a+1),
\end{aligned}$$

²The form of $[\binom{a}{b}]$ is inspired by the similarity with the binomial coefficient.

where functions $R(m, n)$ and $S(m, n)$ can be calculated as follows [127]:

$$R(m, n) = \begin{cases} P(n) & \text{for } m=1 \text{ and } n \geq 1 \\ \sum_{i=1}^{n-m+1} p(i)R(m-1, n-i) & \text{for } 2 \leq m \leq n \end{cases}$$

$$S(m, n) = \begin{cases} Q(n) & \text{for } m=1 \text{ and } n \geq 1 \\ \sum_{i=1}^{n-m+1} q(i)S(m-1, n-i) & \text{for } 2 \leq m \leq n \end{cases}$$

where

$$p(i) = \begin{cases} 1 - q & \text{if } i = 1 \\ q(1 - p)^{i-2}p & \text{otherwise} \end{cases}$$

$$P(i) = \begin{cases} 1 & \text{if } i = 1 \\ q(1 - p)^{i-2} & \text{otherwise} \end{cases}$$

$$q(i) = \begin{cases} 1 - p & \text{if } i = 1 \\ p(1 - q)^{i-2}q & \text{otherwise} \end{cases}$$

$$Q(i) = \begin{cases} 1 & \text{if } i = 1 \\ p(1 - q)^{i-2} & \text{otherwise} \end{cases}$$

$$p = p_{G,B}^{(r)}(T_r) \quad - \text{ given by (A.5)}$$

$$q = p_{B,G}^{(r)}(T_r) \quad - \text{ given by (A.5)}$$

Let us now derive the second building block of Equation (A.8), i.e., $\mathbb{E}[D_r | F_r = j_r]$. To achieve this, we first calculate $\mathbb{P}(D_r = i, F_r = j_r)$. Let us consider the k_r data packets and the $n_r - k_r$ redundancy packets separately, and additionally condition on the state of the last data packet as follows.

$$\begin{aligned} \mathbb{P}(D_r = i, F_r = j_r) &= \\ &= \mathbb{P}(\binom{k_r-1}{i} G) \cdot \mathbb{P}(\binom{n_r-k_r}{j_r-i} | G) + \mathbb{P}(\binom{k_r-1}{i-1} B) \cdot \mathbb{P}(\binom{n_r-k_r}{j_r-i} | B) = \\ &= \mathbb{P}(G \binom{k_r-1}{i}) \cdot \mathbb{P}(\binom{n_r-k_r}{j_r-i} | G) + \mathbb{P}(B \binom{k_r-1}{i-1}) \cdot \mathbb{P}(\binom{n_r-k_r}{j_r-i} | B) = \\ &= \pi_G^{(r)} \mathbb{P}(\binom{k_r-1}{i} | G) \mathbb{P}(\binom{n_r-k_r}{j_r-i} | G) + \pi_B^{(r)} \mathbb{P}(\binom{k_r-1}{i-1} | B) \mathbb{P}(\binom{n_r-k_r}{j_r-i} | B). \end{aligned}$$

The first equality uses the Markov property of the loss model:

$$\begin{aligned} \mathbb{P}(D_r = i, F_r = j_r \mid \text{last data packet is } q) &= \\ &= \mathbb{P}(D_r = i \mid \text{last data packet is } q) \cdot \mathbb{P}(F_r = j_r \mid \text{last data packet is } q), \end{aligned}$$

where $q \in \{G, B\}$. Now it is easy to calculate $\mathbb{E}[D_r | F_r = j_r]$, because

$$\mathbb{E}[D_r | F_r = j_r] = \sum_{i=0}^{k_r} i \cdot \frac{\mathbb{P}(D_r = i, F_r = j_r)}{\mathbb{P}(F_r = j_r)}. \quad (\text{A.10})$$

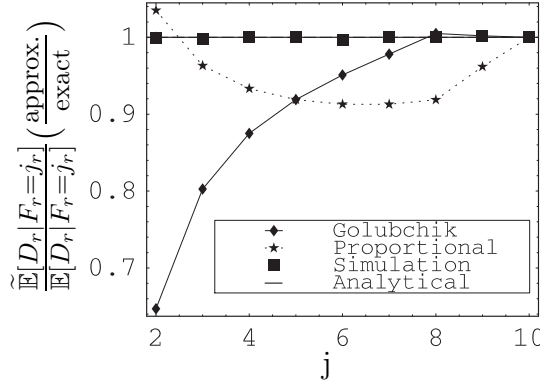


Figure A.6: Approximations of $\mathbb{E}[D_r|F_r = j_r]$ normalized by the correct value given by (A.10). Here $n_r = 10$, $k_r = 8$, $\pi_B^{(r)} = 0.01$ and $1/\mu_B^{(r)} = 2$.

We plug (A.9) and (A.10) into (A.8) and obtain a complete formula for the effective loss rate π_B^* :

$$\pi_B^* = \frac{1}{k} \sum_{j=n-k+1}^n \sum_{\substack{0 \leq j_1, \dots, j_R \leq j \\ j_1 + \dots + j_R = j}} \left(\prod_{r=1}^R \left(\pi_G^{(r)} \cdot \mathbb{P}([n_r^{r-1}]|G) + \pi_B^{(r)} \cdot \mathbb{P}([n_r^{r-1}]|B) \right) \right) \cdot \left(\sum_{r=1}^R \sum_{i=0}^{k_r} i \cdot \frac{\pi_G^{(r)} \cdot \mathbb{P}([k_r^{r-1}]_i|G) \cdot \mathbb{P}([n_r-k_r]_{j_r-i}|G) + \pi_B^{(r)} \cdot \mathbb{P}([k_r^{r-1}]_i|B) \cdot \mathbb{P}([n_r-k_r]_{j_r-i}|B)}{\pi_G^{(r)} \cdot \mathbb{P}([n_r^{r-1}]|G) + \pi_B^{(r)} \cdot \mathbb{P}([n_r^{r-1}]|B)} \right), \quad (\text{A.11})$$

To the best of our knowledge, Equation (A.11) is the first exact solution of this model. Indeed, all previous works used some approximations of $\mathbb{E}[D_r|F_r = j_r]$. In [121] the authors approximate $\mathbb{E}[D_r|F_r = j_r]$ by assuming that any configuration of j losses among the n FEC packets is equally likely; we call this approach ‘Golubchik’. In [122,128] the authors use an intuitive linear formula, i.e., $\mathbb{E}(D_r|F_r = j_r) = \frac{k_r}{n_r} j_r$. Although not mentioned in the papers this is only an approximation that is exact only when $k_r, n_r \rightarrow \infty$; we refer to it as ‘Proportional’. We illustrate the differences between these approximations and the real values in Fig. A.6.

A.4 The design of the schedule \mathcal{S}

In the previous section we derived an exact analytical formula for the effective loss rate π_B^* under a given schedule \mathcal{S} . Here we focus on the design of a good schedule that results in small π_B^* .

Not all schedules are applicable in practice. Indeed, both (i) the maximal allowed FEC block transmission time t_{FEC} and (ii) the packet interval T at the source impose important scheduling constraints. We say that a schedule is *feasible* if all the three following conditions are satisfied:

C1 $\mathcal{T}(i) \geq (i-1) \cdot T$ for $1 \leq i \leq k$, i.e., no data packet is sent before it is generated at the source.

C2 $\mathcal{T}(i) \geq (k-1) \cdot T$ for $k < i \leq n$, i.e., no redundancy packet is sent before all data packets have been generated (we need to collect all data packets in order to create the redundancy packets).

C3 $\mathcal{T}(i) + t_{\mathcal{R}(i)} \leq t_{\text{FEC}}$ for $1 \leq i \leq n$, i.e., all FEC packets should arrive at the destination before the deadline.

We assume that the path rates n_1, \dots, n_R are fixed. There are usually a variety of feasible schedules. Below we discuss two classes of schedules we use in this chapter. The first one, called *Immediate*, reflects the state of the art, whereas the second one, *Spread*, is our proposal.

A.4.1 ‘Immediate’ packet scheduling \mathcal{S}^{imm} - state of the art

We denote by *Immediate* the schedule $\mathcal{S}^{\text{imm}} = (\mathcal{T}^{\text{imm}}, \mathcal{R}^{\text{imm}})$ that represents the approach used in [99,100,121–123,125]. As the name suggests, Immediate sends the data packets as soon as they are generated, i.e., every time interval T . The redundancy packets use the same spacing T . So in general

$$\mathcal{T}^{\text{imm}}(i) = (i-1) \cdot T \quad \text{for } 1 \leq i \leq n. \quad (\text{A.12})$$

This specifies *when* the FEC packets are sent, but not on which path. A good and commonly used guideline for \mathcal{R}^{imm} is to spread the packets on each path separately with (roughly) even spacing [100]. When the rates are equal, i.e., $n_1 = n_2 = \dots = n_R$, then this boils down to a simple round-robin schedule applied in [121–123,125]. In contrast, when the rates differ, a more elaborate approach should be used. For this purpose we adopt the credit-based technique proposed in [100], as follows. Each path is associated with a credit initially equal to 0. Before each FEC packet transmission the credit of every path P_r is increased by n_r/n . Next, the path with the largest credit is selected to transmit this packet; the credit of this path is decreased by 1. This scheme is iterated until all n FEC packets are sent.

The Immediate schedule can be interpreted as the following function:

$$\mathcal{S}^{\text{imm}} = \text{Immediate}(n_1 \dots n_R, T)$$

Two examples of Immediate schedules \mathcal{S}^{imm} are given in Fig. A.3: (C) is a single-path schedule, i.e., with $n_1 = 6$ and $n_2 = 0$, whereas in (D) we use two paths and $n_1 = n_2 = 3$.

A.4.2 ‘Spread’ packet scheduling \mathcal{S}^{spr} - our proposal

Under Immediate, all packets are sent as soon as they are generated, i.e., according to (A.12). Instead, we propose to *spread the packets evenly in all the available time on each path*. We call this schedule *Spread* $\mathcal{S}^{spr} = (\mathcal{T}^{spr}, \mathcal{R}^{spr})$. Compared with Immediate, Spread additionally takes the path propagation times $t_1 \dots t_R$ and the maximal FEC block delay t_{FEC}^{spr} as parameters, i.e.,

$$\mathcal{S}^{spr} = \text{Spread}(n_1 \dots n_R, T, t_1 \dots t_R, t_{FEC}^{spr}).$$

The design of Spread is not straightforward. Indeed, as the k data packets are generated at the source with spacing T , the paths are inter-dependent, which may easily lead to the violation of the constraint C1. For example, if we schedule packet 1 on P_1 at time $\mathcal{T}(1) = 0$ (and $k > 1$), then no other packet on any path can be scheduled before time $t = T$.

We guarantee the feasibility of Spread as follows. First, we order the paths according to their rates, starting from the path with the highest rate. (When two paths have the same rate, we take the one with a higher path propagation time first.) We consider the paths one by one following this order. For each path P_r we spread the packets evenly on time interval $[t^{(r)}, t_{FEC}^{spr} - t_r]$, where $t^{(r)}$ takes the smallest possible value that satisfies the feasibility condition. (The value of $t^{(r)}$ usually grows with the number of paths processed.) We iterate this algorithm until all paths have been scheduled.

We present two examples of Spread schedules \mathcal{S}^{spr} in Fig. A.3. We use $t_{FEC}^{spr} = 170ms$ and two different sets of rates: $n_1 = n_2 = 3$ in (E) and $n_1 = 4$, $n_2 = 2$ in (F).

Spread is very effective. Indeed, we can prove that

Theorem 11 *The Spread schedule is optimal for the repetition code $FEC(n, 1)$.*

Proof Under $FEC(n, 1)$ every data packet is replicated and sent in n copies; the reception of at least one such copy leads to a success. As there is only one data packet, all the redundancy packets (i.e., the duplicates of the data packet) can be generated already at time $t=0$. This eliminates all the time dependencies between the paths. Therefore, every path P_r separately must maximize the probability of at least one successful transmission. It is achieved by *even* spreading on the maximal allowed time interval $[0, t_{FEC}^{spr} - t_r]$. (The proof for the under repetition code on a single path can be found in [96].) This, in turn, is exactly what Spread returns under $FEC(n, 1)$. ■

Spread builds on even packet spreading - a simple and widely accepted guideline that is often thought of as leading to the optimal solution. Indeed, its optimality was proven for some particular cases [96]. But, surprisingly, this is not a general result. Consider for example FEC(4,3) on a single path (i.e., $\mathcal{R} = (1, 1, 1, 1)$) with loss rate $\pi_B^{(1)} = 1\%$ and average loss burst length $1/\mu_B^{(1)} = 5ms$, and available time interval equal to 15ms. The even spreading schedule $\mathcal{S}_1 = ((0, 5, 10, 15), \mathcal{R})$ yields $\pi_B^* = 0.53\%$. But the optimal schedule (found with optimization tools of Mathematica [130]) is $\mathcal{S}_1 = ((0, 7.16, 12.51, 15), \mathcal{R})$ and yields $\pi_B^* = 0.50\%$.

This means that Spread does *not* guarantee optimality in the general FEC(n, k) case. However, we show later in simulations that it usually leads to almost-optimal solutions and is thus an effective and practical rule of thumb.

A.4.3 Comparison of \mathcal{S}^{imm} and \mathcal{S}^{spr} : Optimal schedules \mathcal{S}_{opt}^{imm} and \mathcal{S}_{opt}^{spr} , and loss rate improvement γ .

It was shown in previous studies that Immediate multipath is better than a single path communication. The main point we make here is that once we allow for multipath, the Spread schedule \mathcal{S}^{spr} that we propose in this chapter is significantly better than the Immediate schedule \mathcal{S}^{imm} representing the state of the art.

In order to demonstrate this, we compare the performance of \mathcal{S}^{imm} and \mathcal{S}^{spr} in terms of their effective loss rates. What rates $n_1 \dots n_R$ and what FEC block transmission time t_{FEC} should we use to make this comparison meaningful and fair? We should allow Immediate and Spread to optimize their rates $n_1 \dots n_R$ independently, given that they impose identical FEC block transmission times $t_{FEC}^{imm} = t_{FEC}^{spr}$. More precisely, we assume that the FEC parameters n and k are fixed, and we proceed in two steps. First, we optimize the rates $n_1 \dots n_R$ of Immediate, such that the effective loss rate π_B^* is minimized. It results in the optimal Immediate schedule \mathcal{S}_{opt}^{imm} . This, in turn, specifies t_{FEC}^{imm} as shown in (A.2). In the second step, we set $t_{FEC}^{spr} = t_{FEC}^{imm}$ and optimize the rates $n_1 \dots n_R$ of Spread, resulting in the optimal Spread schedule \mathcal{S}_{opt}^{spr} .³

Finally, we define the *relative effective loss rate improvement* γ as the relative gain in π_B^* due to the usage of optimal Spread instead of optimal

³Note that \mathcal{S}_{opt}^{imm} and \mathcal{S}_{opt}^{spr} are optimal subject to their construction constraints presented in A.4.1 and A.4.2, respectively.

Immediate, i.e.,

$$\gamma = \frac{\pi_B^*(\mathcal{S}_{opt}^{imm})}{\pi_B^*(\mathcal{S}_{opt}^{spr})}. \quad (\text{A.13})$$

The metric γ can be precisely evaluated by formulas (A.7) and (A.11). The values of γ can be easily interpreted; for example, $\gamma > 1$ means that Spread performs better than Immediate.

A.4.4 Capacity constraints

So far we have considered the case where every path P_r can be assigned with any rate $0 \leq n_r \leq n$. In practice, however, P_r may have a relatively limited capacity, which would impose a direct constraint on n_r . Fortunately, integrating these constraints in our model is straightforward. Indeed, it is enough to respect them when computing the rates $n_1 \dots n_R$ in \mathcal{S}_{opt}^{spr} and \mathcal{S}_{imm}^{spr} in A.4.3.

A.5 Performance evaluation

In this section we evaluate our approach first in simulations and next on real-life traces.

A.5.1 Simulation results

The goal of simulations is twofold. First, we verify the correctness of our analytical results. Second, we can test our idea in a fully controlled environment and study the effect of various parameters on the results.

Default values of parameters

If not stated otherwise, in our simulations we use the following default values. The data packets are generated at the source with interval $T = 5ms$. Next they are encoded by systematic FEC(10,8) and sent over R independent paths. For the sake of simplicity we speak mainly of systems with $R = 2$ paths: P_1 and P_2 . It allows us to describe the path propagation time differences by a single parameter $\Delta t = t_2 - t_1$ that takes the default value $\Delta t = 100ms$. Finally, the paths P_1 and P_2 have the same average failure rate $\pi_1 = 0.01$ and the average loss burst length equal to $1/\mu_1 = 10ms$.

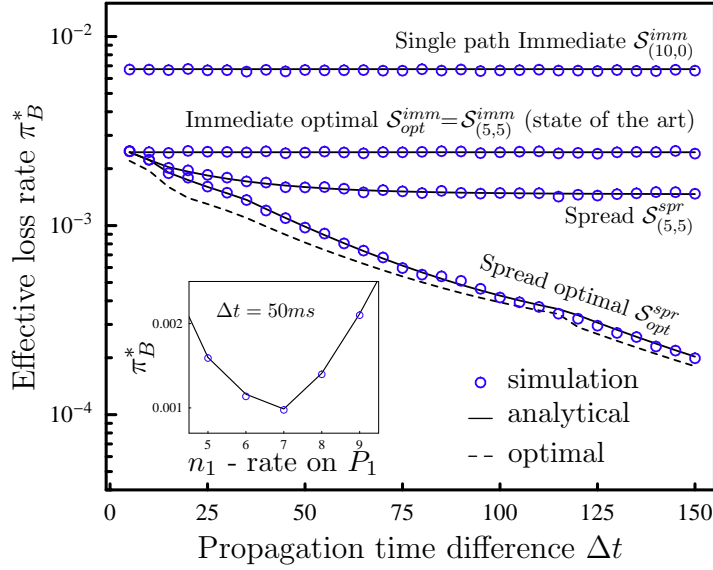


Figure A.7: The effective loss rate π_B^* as a function of path propagation time difference Δt . We use $FEC(10, 8)$ on two independent paths, P_1 and P_2 , with data packet spacing $T = 5$ at the source. The losses on P_1 and P_2 are modeled by continuous time Gilbert model with the same average failure rate $\pi_B = 0.01$ and the average burst length equal $1/\mu_B = 10ms$. Four schedules are used: $\bullet \mathcal{S}_{(10,0)}^{imm}$ - all packets are sent on a single path P_1 with interval T , $\bullet \mathcal{S}_{(5,5)}^{imm}$ - Immediate with optimal rates $n_1 = n_2 = 5$, $\bullet \mathcal{S}_{(5,5)}^{spr}$ - Spread with $n_1 = n_2 = 5$, $\bullet \mathcal{S}_{opt}^{spr}$ - Spread with the rates n_1, n_2 chosen optimally based on the value of Δt . Additionally, the dashed curve shows the effective loss rate of the optimal schedule, where packets are not restricted to even spacing on each path, as described in Section A.4. The optimal schedule was found with numerical optimization tools of Mathematica [130]. **Inset:** π_B^* as a function of rate n_1 on path P_1 for $\Delta t = 50ms$ under Spread. In both figures the plain lines are the theoretical values according to formula (A.11), whereas the circles are the results obtained in a simulation of the model. The size of confidence intervals (not shown) is comparable with the size of the circles.

The effective loss rate π_B^* as a function of Δt

In Fig. A.7 we plot the effective loss rate π_B^* as a function of Δt for four different schedules. Our first observation is that the results obtained in a simulation of the model (circles) fit precisely the analytical curves (plain lines).

Next, we compare the performance of various schedules. As the loss properties of the two paths are identical, the previous techniques described

in [99,100,121] split the FEC packets equally between P_1 and P_2 . This results in the optimal Immediate schedule $\mathcal{S}_{opt}^{imm} = \mathcal{S}_{(5,5)}^{imm}$, i.e., with $n_1 = n_2 = 5$. As this schedule uses multipath transmission, it is not surprising that \mathcal{S}_{opt}^{imm} significantly outperforms the single path Immediate schedule $\mathcal{S}_{(10,0)}^{imm}$. Note also that, by construction, Δt does not affect the performance of any of them.

In contrast, in Spread $\mathcal{S}_{(5,5)}^{spr}$ we use the same rates as in \mathcal{S}_{opt}^{imm} , but we spread the packets uniformly within the time budget t_{FEC}^{imm} set by $\mathcal{S}_{(5,5)}^{imm}$ (similar schedule is shown in Fig. A.3E). It results in a further decrease of the effective loss rate π_B^* . This difference moderately grows with Δt . However, for larger Δt the rates (5, 5) become suboptimal under Spread. For instance, in the inset in Fig. A.7 we show the performance of Spread under various rate configurations $(n_1, n - n_1)$; the minimum is reached for (7, 3). As described in A.4.3, allowing for this rate optimization leads to the optimal Spread schedule \mathcal{S}_{opt}^{spr} . Its advantage over $\mathcal{S}_{(5,5)}^{imm}$ grows roughly exponentially with Δt .

Finally, we observe that the performance of the optimal Spread schedule \mathcal{S}_{opt}^{spr} is very close to the global optimum (dashed curve) where packets are not necessarily evenly-spaced, as described in Section A.4. This confirms the usefulness of the even-spread guideline that we follow in Spread.

Loss rate improvement γ as a function of various parameters

Clearly, there are many parameters that affect the performance of the schedules. We study the effect of some of them on the relative loss rate improvement γ in Fig. A.8.

First, plot (A) confirms that the advantage of Spread over Immediate grows with the path propagation time difference Δt .

Second, with growing packet interval T at the source, the fixed Δt becomes a smaller fraction of the entire FEC block transmission time t_{FEC} . As a consequence, there is relatively less to exploit and γ drops with T , see plot (B). A similar phenomenon can be observed in plot (C), where t_{FEC} grows due to an increase of the number n of FEC packets.

Finally, in Fig. A.8D we vary the loss rate $\pi_B^{(2)}$ of path P_2 . The difference between path loss rates is a crucial parameter affecting the performance gain of Immediate multipath over the single path transmission. Indeed, if out of two paths one is very lossy and the other one is very good, then the optimal Immediate multipath schedule \mathcal{S}_{opt}^{imm} uses mainly (or only) the better path, which substantially limits the gain of multipath [99,123]. This is illustrated in plot (D) by the dashed curve; the ratio $\pi_B^*(\mathcal{S}_{(10,0)}^{imm})/\pi_B^*(\mathcal{S}_{opt}^{imm})$ is the largest

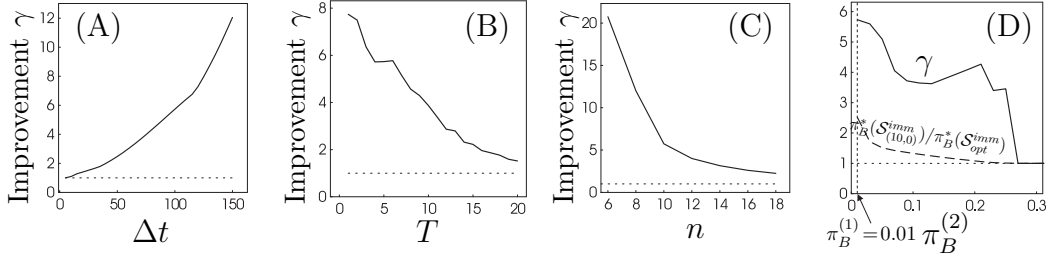


Figure A.8: Relative loss rate improvement γ due to usage of Spread instead of Immediate as a function of four parameters: (A) path propagation time difference Δt , (B) packet generation interval T at the source, (C) the size n of a FEC block, (D) loss rate $\pi_B^{(2)}$ of path P_2 . We consider a system with $R = 2$ paths and the following default parameters: FEC(10,8), $\Delta t = 100ms$, $T = 5ms$, $\pi_B^{(r)} = 1\%$, $1/\mu_B^{(r)} = 10ms$, $k = n - 2$. All results shown here are analytical. The irregular shapes of the curves in this and other figures are expected, because the computation of γ involves the rates optimization (see A.4.3). For instance, in figure (D), going from left to right, the optimal Immediate and Spread rates (n_1, n_2) change gradually (and separately) from (5, 5) to (10, 0); every such rate transition may introduce irregularities in the shape of the curves.

when the paths have identical loss properties, and quickly diminishes with growing difference between $\pi_B^{(1)}$ and $\pi_B^{(2)}$.

We could expect a similar diminishing effect for the advantage $\gamma = \pi_B^*(\mathcal{S}_{opt}^{imm})/\pi_B^*(\mathcal{S}_{opt}^{spr})$ of Spread over Immediate. Surprisingly, this is not the case; γ remains relatively stable ($3 < \gamma < 6$) for a wide range of values of $\pi_B^{(2)}$. For $\pi_B^{(2)} \approx 0.25$ the path P_2 becomes too lossy, and both Immediate and Spread send all packets on P_1 only and thus become equivalent.

Minimizing t_{FEC} - decreasing delays and fighting jitter

So far we used Spread to minimize the effective loss rate π_B^* while keeping the FEC block transmission time t_{FEC}^{spr} not larger than that of Immediate schedule t_{FEC}^{imm} . Let us now reverse the problem: Let us minimize the FEC block transmission time t_{FEC}^{spr} of Spread, while keeping its effective loss rate not larger than that of Immediate, i.e., subject to $\pi_B^*(\mathcal{S}_{opt}^{spr}) \leq \pi_B^*(\mathcal{S}_{opt}^{imm})$.

We plot the results in Fig. A.9. The gain $t_{FEC}^{imm} - t_{FEC}^{spr}$ in FEC block transmission time is significant and grows roughly linearly with Δt , as $t_{FEC}^{imm} - t_{FEC}^{spr} \simeq \Delta t/2$. The reduction of t_{FEC} brings obvious advantages to delay-constrained applications using the multipath FEC system. First, the effective end-to-end delays get smaller which allows us to reduce the playout time at

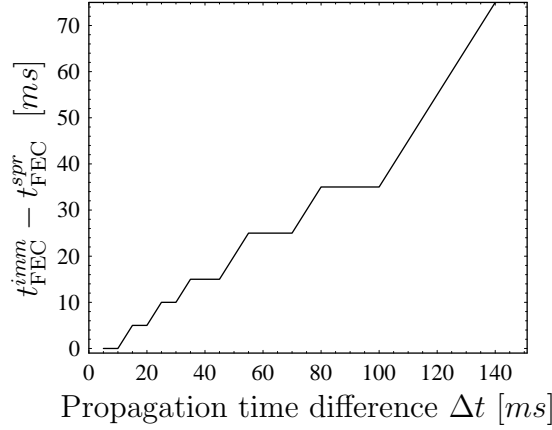


Figure A.9: The gain in FEC block transmission time t_{FEC} by the usage of Spread instead of Immediate. Parameters: FEC(10,8), $\pi_B^{(1)} = \pi_B^{(2)} = 0.01$, $1/\mu_B^{(1)} = 1/\mu_B^{(2)} = 10\text{ms}$, $T = 5\text{ms}$. For these parameters, the optimal Immediate rates are $n_1 = n_2 = 5$, which results in the effective loss rate $\pi_B^*(\mathcal{S}_{(5,5)}^{\text{imm}}) = 0.24\%$. For Spread we choose the minimal FEC block transmission time $t_{\text{FEC}}^{\text{spr}}$ such that $\pi_B^*(\mathcal{S}_{\text{opt}}^{\text{spr}}) \leq \pi_B^*(\mathcal{S}_{(5,5)}^{\text{imm}})$.

the destination, keeping the same level of the effective loss rate.

Another important interpretation is related to the delay *jitter*, i.e., variations of path propagation times. Indeed, in this work, as in most previous works on multipath FEC, we consider the path propagation times constant and focus on (correlated) packet losses only. However, as Spread results in a smaller delay t_{FEC} , it also leaves more space to accommodate potential jitter, naturally making Spread more robust to jitter than Immediate.

Other FEC parameters n, k

So far we assumed that Immediate and Spread use the same general FEC parameters n and k ; only the rates on particular paths could be optimized. However, in some cases the optimal choice of n and k under Spread may differ from that of Immediate, given the same redundancy k/n .

For example, according to our additional simulations (not shown here), for the setting in Fig. A.7 and $\Delta t > 220\text{ms}$, the Spread schedule using FEC(15,12) would outperform Spread with FEC(10,8). Similarly, FEC(12,6) would be better for Spread than FEC(10,5) for $\Delta t > 140\text{ms}$. Note, however, that this phenomenon can be observed only for relatively large values of Δt that rarely occur in reality.

To conclude, the loss rate improvement γ can be manifold, but its exact

value strongly depends on various parameters. First, the advantage of Spread over Immediate grows with path propagation time differences Δt , but drops with the data packet generation interval T and FEC block size n . Second, the optimal Immediate rates are not always optimal under Spread; usually optimal Spread sends more packets on faster paths. Third, although the advantage of Immediate over a single path transmission quickly diminishes with growing differences between the loss rates of the paths, the advantage of Spread over Immediate is relatively stable. Finally, Spread can also achieve FEC block transmission times much smaller than Immediate, still guaranteeing the same or better performance in terms of the effective loss rate. This results not only in smaller effective delays, but also in a higher robustness to delay jitter.

A.5.2 Trace-driven PlanetLab evaluation

In the previous section we presented analytical and simulation results where the packet losses were modeled by the Continuous Time Gilbert Model. As any model, it is only an approximation of reality. In this section we feed our simulations with real-life packet loss traces collected in Internet experiments.

Data sets

The traces come from two different PlanetLab (PL) [126] experiments. On every path the packets are sent with time-interval T , i.e., with the generation rate at the source. Every trace is a sequence composed of symbols G (packet not lost) and B (packet lost).

Every time-constrained experiment on PlanetLab should be designed and interpreted carefully. This is because at any point in time most of PlanetLab nodes are overloaded. Not only their CPU utilization is at 100%, but more importantly the queueing delays experienced by the running processes can be very significant - even up to several seconds between two consecutive accesses to CPU. This results in incorrect propagation time measurements and packet dropping due to incoming buffer overflow at the destination [109, 131]. Moreover, the situation changes dynamically. We minimize the effects of these problems by introducing periodic pauses in packet generation and avoiding the highly loaded PlanetLab nodes.

We use the following two data sets.

‘Relays’ - PlanetLab with relays In this experiment every trace is collected on a two-hop overlay path between three PlanetLab nodes: source, relay and destination. The UDP packets at the source are generated every

$T = 5ms$ and sent immediately to the relay that forwards them to the destination. After every one-second-long packet generation period we introduce one second of idle time in order to avoid dropping packets at PlanetLab hosts when the probing traffic is too bursty. We collected more than 5'000 traces, each covering 100 seconds of packet generation time.

In order to further reduce the effect of overloaded PL nodes on the results, for every experiment separately we select the source, relay and destination randomly from 50 currently least loaded PL nodes. As the load estimate we use the number of processes queueing for the CPU and I/O devices; it can be obtained by parsing the file `/proc/stat` that stores the information about kernel activity.

‘Web sites’ - PlanetLab to popular web sites This data set consists of 2'839 traces used in [100]. They were collected by sending 16-byte ICMP echo packets from 57 PlanetLab hosts to 55 popular web sites selected from [132]. Next, the ICMP echo-reply packets were captured by `Tcpdump`, resulting in traces where packets travel from a PlanetLab node to a web site and back to the original PlanetLab node. The packets were sent every $T = 2ms$. As above, every one-second packet generation time was followed by one-second idle time. Each measurement lasted at least 800 seconds. As in [100], we split it into 40-second long intervals that we call chunks.

Despite the measures we took, in both data sets we find traces with numerous long (100ms and more) blocks of consecutive losses. As this is not caused by network losses, but rather by buffer overflow at the nodes due to CPU queueing, we exclude these traces from our simulations.

Simulation setting

In a simulation of a R -path scenario we use R traces (one per path) randomly chosen from the pool of all available traces. Thus, by construction, the R traces are independent (typically generated at different times and places in the Internet). For the sake of simplicity, we restrict the presentation to the case $R = 2$.

Our basic metric is loss rate improvement γ . As described in Section A.4.3, it optimizes the rates of Immediate and Spread. This optimization is based on the observed traces. One approach to do this is to infer for every path its loss rate $\pi_B^{(r)}$ and the average loss burst length $1/\mu_B^{(r)}$, feed them into the model and optimize the rates as in section A.5.1. However, this technique has two drawbacks: it introduces errors when measuring the path properties, and assumes a particular packet loss model. We avoid these problems by

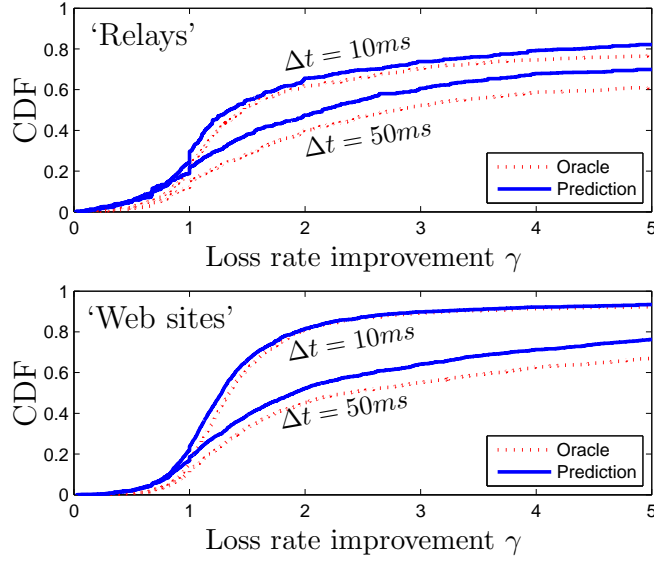


Figure A.10: The effective loss rate improvement γ (by using Spread instead of Immediate) in trace-driven simulations under FEC(10, 8). We use $R = 2$ independent paths with real-life loss traces; their propagation times differ by Δt . We consider two data sets: ‘Relays’ (top) and ‘Web sites’ (bottom).

working directly on the traces - the optimal rates in \mathcal{S}_{opt}^{imm} and \mathcal{S}_{opt}^{spr} are those that perform best on a given chunk.

We present two types of results. In *Oracle* we choose the optimal rates for the currently evaluated chunk. In contrast, in *Prediction* we use the optimal rates of the preceding chunk to evaluate the current chunk. Thus Oracle shows the best achievable results for Immediate and Spread with no prediction errors, whereas Prediction is a practical implementation.

Results

In Fig. A.10 we present the results for FEC(10, 8). The figure presents the cumulative distribution of the relative loss rate improvement γ for $\Delta t = 10ms$ and $\Delta t = 50ms$. We consider the cases where optimal Immediate uses both paths (i.e., $n_1 \neq 0$ and $n_1 \neq 10$) and there is a space for improvement (i.e., $\pi_B^*(\mathcal{S}_{opt}^{imm}) > 0$). In about 90% of cases we observe an advantage of Spread over Immediate. For instance, for both data sets under Oracle with $\Delta t = 50ms$, in 50% of cases the loss rate drops by a factor of 3 or more when we use Spread instead of Immediate. For smaller Δt the advantage is less pronounced, which is in agreement with the results presented in the previous section.

Surprisingly, in roughly 10% of cases Spread performs slightly worse than Immediate. A possible explanation is that in some traces we can find loss patterns that are periodic, presumably due to other applications running on PlanetLab nodes. If such an unnatural loss pattern gets aligned with the packets scheduled by Spread on one or more paths, then the performance of Spread may drop below Immediate.

Finally, we find our simple prediction method satisfactory, as the Prediction curve is always close to Oracle.

A.6 Related work

The performance of FEC on a *single* path with correlated loss failures was studied e.g., in [96,119,127]. One common conclusion is that the FEC efficiency drops with growing burstiness of packet losses.

Multipath transmission as a way of de-correlating the packet losses and increasing the performance of FEC was first proposed in [120]. It got more attention recently, e.g., in [99,100,121–123,125]. Multipath was also studied in the context of Multiple Description Coding [98].

In [99] the authors study a multipath FEC system by simulations only, on artificially generated graphs. They also give a heuristic to select from a number of candidate paths a set of highly disjoint paths with relatively small propagation delays.

There are a number of approaches to evaluate *analytically* the performance of multipath FEC with independent paths and bursty path losses. For instance, [121–123] and [100] give four different derivations of the effective loss rate π_B^* (or related metrics) in such a setting. However, in all four cases the resulting formula is only an *approximation* of the complete solution due to (sometimes very significant) model simplifications. First, [122] [123] use a discrete Gilbert model. Thus two consecutive packets on one path are equally correlated irrespectively of the time intervals between them, which makes the models inherently unable to capture any aspects of varying packet spacing. [100] also uses the a discrete Gilbert model, but adapts the transition matrix appropriately. Second approximation comes when computing the number of lost data packets given that a FEC block cannot be entirely recovered: [121] and [122] use approximations described at the end of section A.3.2, [123] simplifies the model by assuming that in such a case all data packets are lost, and [100] assumes that the numbers of lost data packets and redundancy packets are not correlated. Third, [122] considers only a scenario with identical loss statistics on every path. Finally, [100] assumes a large number of active paths $R \gg 1$ and small individual path rates $n_r \ll n$.

This allows the authors to apply the central limit theorem and approximate the joint distribution of the number of lost data and redundancy packets by a bivariate normal distribution.

To the best of our knowledge, we are the first ones to give an exact analytical formula for the effective loss rate π_B^* of FEC protection scheme on multiple independent paths with path losses modeled by the Gilbert model.

As in most other approaches, we assume that the background cross-traffic is much larger than our own, and thus the load we impose on a path does not affect its loss statistics. Scenarios where this assumption does not hold were studied in [128] in the context of a single path FEC, and in [133] for multipath FEC.

As in [100,121,125,133] we assume the paths to be independent. This can be achieved by detecting correlated paths in end-to-end measurements [134] and treating them as one. Another approach is to find paths that are IP link disjoint, which should be possible if the site is multi-homed. Finally, even if all the available paths are to some extent correlated we can still get some performance benefits [99,122,124,135], though limited [136,137].

Finally and *most importantly*, to the best of our knowledge no attempt has been made to exploit the path propagation time differences in multipath FEC. Indeed, all the works listed above use some variant of the Immediate schedule, where packets are sent as soon as they arrive at the source. In contrast, we have proposed the Spread schedule that exploits these propagation time differences and significantly improves the performance.

A.7 Conclusion

In this chapter we presented some interesting side results of the thesis. In particular, we considered the multipath transmission with FEC. We started from the observation that the propagation times on multiple paths between a pair of nodes may significantly differ. We proposed to exploit these differences in the context of delay-constrained multipath systems using FEC, by applying the Spread schedule. We have evaluated our solution by a precise analytical approach, and simulations based on both the model and real-life Internet traces. Our studies show that Spread substantially outperforms previous solutions. It achieves a several-fold improvement (reduction) of the effective loss rate. Or conversely, keeping the same level of effective loss rate Spread significantly decreases the FEC block transmission time, which limits the observed delays and helps fighting the delay jitter.

Bibliography

- [1] M. Kurant and P. Thiran, “Trainspotting: Extraction and analysis of traffic and topologies of transportation networks,” *Phys. Rev. E*, vol. 74, p. 036114, 2006.
- [2] —, “Layered complex networks,” *Phys. Rev. Lett.*, vol. 96, no. 13, p. 138701, April 2006.
- [3] A. Liebers, “Analyzing train time table graphs,” Ph.D. dissertation, University of Konstanz, Department of Computer and Information Science, 2001.
- [4] P. Hagmann, M. Kurant, X. Gigandet, P. Thiran, V. Wedeen, R. Meuli, and J. Thiran, “Mapping brain networks of structural connectivity with MRI tractography,” *Manuscript in preparation*, 2006.
- [5] M. Kurant, P. Hagmann, and P. Thiran, “Error and attack tolerance of layered complex networks,” *Phys. Rev. E*, vol. 76, p. 026103, 2007.
- [6] M. Kurant and P. Thiran, “On Survivable Routing of Mesh Topologies in IP-over-WDM Networks,” *Proc. of Infocom*, 2005.
- [7] —, “Survivable routing of mesh topologies in ip-over-wdm networks by recursive graph contraction,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 5, pp. 922–933, June 2007.
- [8] —, “Survivable Mapping Algorithm by Ring Trimming (SMART) for large IP-over-WDM networks,” *Proc. of BroadNets*, 2004.
- [9] —, “Survivable Routing in IP-over-WDM Networks in the Presence of Multiple Failures,” <http://arxiv.org/abs/cs.NI/0604053>, 2006.
- [10] —, “Achieving maximal path diversity in low-delay application-level multicast,” *Submitted to Infocom 2009*, 2008.

- [11] M. Kurant, “Exploiting the path propagation time differences in multipath transmission with fec,” *Infocom*, 2009.
- [12] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of the internet topology,” *Proc. of ACM SIGCOMM*, 1999.
- [13] R. Albert, H. Jeong, and A.-L. Barabási, “Diameter of the world wide web,” *Nature*, vol. 401, pp. 130–131, 1999.
- [14] D. J. Watts and S. H. Strogatz, “Collective dynamics of “small-world” networks,” *Nature*, vol. 393, pp. 440–442, 1998.
- [15] A. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, pp. 509–512, 1999.
- [16] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, “The architecture of complex weighted networks,” *Proc. Natl. Acad. Sci. USA*, vol. 101, no. 11, p. 3747, 2004.
- [17] M. Newman, A.-L. Barabasi, and D. J. Watts, *The Structure and Dynamics of Networks*. Princeton University Press, 2006.
- [18] *Recent attacks on TVG:* <http://tinyurl.com/5mnt5v>, <http://tinyurl.com/5mnt5v>.
- [19] R. Albert, H. Jeong, and A.-L. Barabási, “Error and attack tolerance of complex networks,” *Nature*, vol. 406, p. 378, 2000.
- [20] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin, “Resilience of the internet to random breakdowns,” *Phys. Rev. Lett.*, vol. 85, p. 4626, 2000.
- [21] ———, “Breakdown of the internet under intentional attack,” *Phys. Rev. Lett.*, vol. 86, p. 3682, 2001.
- [22] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts, “Network robustness and fragility: Percolation on random graphs,” *Phys. Rev. Lett.*, vol. 85, p. 5468, 2000.
- [23] P. Holme and B. J. Kim, “Vertex overload breakdown in evolving networks,” *Phys. Rev. E*, vol. 65, p. 066109, 2002.
- [24] L. K. Gallos, R. Cohen, P. Argyrakis, A. Bunde, and S. Havlin, “Stability and topology of scale-free networks under attack and defense strategies,” *Phys. Rev. Lett.*, vol. 94, p. 188701, 2005.

- [25] A. E. Motter, “Cascade control and defence in complex networks,” *Phys. Rev. Lett.*, vol. 93, no. 9, p. 098701, 2004.
- [26] L. Zhao, K. Park, and Y.-C. Lai, “Attack vulnerability of scale-free networks due to cascading breakdown,” *Phys. Rev. E*, vol. 70, p. 035101(R), 2004.
- [27] L. da Fontoura Costa, “Reinforcing the resilience of complex networks,” *Phys. Rev. E*, vol. 69, p. 066127, 2004.
- [28] T. Tanizawa, G. Paul, R. Cohen, S. Havlin, and H. E. Stanley, “Optimization of network robustness to waves of targeted and random attacks,” *Phys. Rev. E*, vol. 71, p. 047101, 2005.
- [29] V. Latora and M. Marchiori, “Vulnerability and protection of infrastructure networks,” *Phys. Rev. E*, vol. 71, p. 015103(R), 2005.
- [30] M. Schäfer, J. Scholz, and M. Greiner, “Proactive robustness control of heterogeneously loaded networks,” *Phys. Rev. Lett.*, vol. 96, p. 108701, 2006.
- [31] L. Dall’Asta, A. Barrat, M. Barthélemy, and A. Vespignani, “Vulnerability of weighted networks,” *physics/0603163*, 2006.
- [32] “Dimes,” <http://www.netdimes.org>.
- [33] <http://www.caida.org/>.
- [34] D. Stutzbach, R. Rejaie, and S. Sen, “Characterizing unstructured overlay topologies in modern p2p file-sharing systems,” *Proc. of IMC’05*, 2005.
- [35] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Sociometry*, vol. 40, p. 35, 1977.
- [36] K.-I. Goh, B. Kahng, and D. Kim, “Universal behavior of load distribution in scale-free networks,” *Phys. Rev. Lett.*, vol. 87, no. 27, p. 278701, Dec. 2001.
- [37] B. Bollobás and O. Riordan, “Shortest paths and load scaling in scale-free trees,” *Phys. Rev. E*, vol. 69, no. 3, p. 036114, Mar. 2004.
- [38] L. Gao, “On inferring autonomous system relationships in the internet.” *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

- [39] P. Sen, S. Dasgupta, A. Chatterjee, P. A. Sreeram, G. Mukherjee, and S. S. Manna, “Small-world properties of the Indian railway network,” *Phys. Rev. E*, vol. 67, p. 036106, 2003.
- [40] K. A. Seaton and L. M. Hackett, “Stations, trains and small-world networks,” *Physica A*, vol. 339, p. 635, 2004.
- [41] J. Sienkiewicz and J. A. Hołyst, “Statistical analysis of 22 public transport networks in Poland,” *Phys. Rev. E*, vol. 72, p. 046127, 2005.
- [42] V. Latora and M. Marchiori, “Efficient behavior of small-world networks,” *Phys. Rev. Lett.*, vol. 87, p. 198701, 2001.
- [43] —, “Is the boston subway a small-world network?” *Physica A*, vol. 314, p. 109, 2002.
- [44] M. T. Gastner and M. E. J. Newman, “Shape and efficiency in spatial distribution networks,” *J. Stat. Mech.*, no. P01015, January 2006.
- [45] I. Vragović, E. Louis, and A. Diaz-Guilera, “Efficiency of informational transfer in regular and complex networks,” *Phys. Rev. E*, vol. 71, p. 036122, 2005.
- [46] J. A., M. B., S. R., and X. X., “Guest editorial: Traffic engineering for multi-layer networks,” *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 5, pp. 865–867, June 2007.
- [47] H. Zang, J. P. Jue, and B. Mukherjee, “A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks,” *SPIE Optical Networks Magazine*, vol. 1, no. 1, pp. 47–60, 2000.
- [48] G. Li, B. Doverspike, and C. Kalmanek, “Fiber Span Failure Protection in Mesh Optical Networks,” *Optical Networks Magazine*, vol. 3, no. 3, pp. 21–31, May/June 2002.
- [49] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, “Characterization of Failures in an IP Backbone,” *Proc. of IEEE INFOCOM’04*, 2004.
- [50] L. Sahasrabudde, S. Ramamurthy, and B. Mukherjee, “Fault management in IP-Over-WDM Networks: WDM Protection vs. IP Restoration,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 1, January 2002.

- [51] M. Kurant, H. X. Nguyen, and P. Thiran, *Dependable Systems: Software, Computing, Networks*. Springer Lecture Notes in Computer Science, 2006, ch. Survey on Dependable Networks.
- [52] A. Fumagalli and L. Valcarengi, “IP Restoration vs. WDM Protection: Is There an Optimal Choice?” *IEEE Network*, Nov/Dec 2000.
- [53] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, “Feasibility of IP restoration in a tier-1 backbone,” *Sprint ATL Research Report Nr. RR03-ATL-030666*, 2003.
- [54] J. Armitage, O. Crochat, and J. Y. L. Boudec, “Design of a Survivable WDM Photonic Network,” *Proceedings of IEEE INFOCOM 97*, April 1997.
- [55] E. Modiano and A. Narula-Tam, “Survivable lightpath routing: a new approach to the design of WDM-based networks,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 800–809, May 2002.
- [56] F. Giroire, A. Nucci, N. Taft, and C. Diot, “Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection,” *Proc. of IEEE INFOCOM 2003*, 2003.
- [57] E. Leonardi, M. Mellia, and M. A. Marsan, “Algorithms for the Logical Topology Design in WDM All-Optical Networks,” *Optical Networks Magazine*, January 2000.
- [58] F. Glover, E. Taillard, and D. Werra, “A user’s guide for tabu search,” *Annals of Operations Research*, no. 41, pp. 3–28, 1993.
- [59] O. Crochat and J. Y. L. Boudec, “Design Protection for WDM Optical Networks,” *IEEE Journal of Selected Areas in Communication*, vol. 16, no. 7, pp. 1158–1165, September 1998.
- [60] A. Nucci, B. Sansò, T. Crainic, E. Leonardi, and M. A. Marsan, “Design of Fault-Tolerant Logical Topologies in Wavelength-Routed Optical IP Networks,” *Proc. of IEEE Globecom 2001*, 2001.
- [61] F. Ducatelle and L. Gambardella, “Survivable routing in IP-over-WDM networks: An efficient and scalable local search algorithm,” *Optical Switching and Networking*, vol. 2, no. 2, pp. 86–99, September 2005.

- [62] L.-W. Chen and E. Modiano, "Efficient Routing and Wavelength Assignment for Reconfigurable WDM Networks with Wavelength Converters," *Proc. of IEEE INFOCOM 2003*, 2003.
- [63] H. Lee, H. Choi, S. Subramaniam, and H.-A. Choi, "Survival Embedding of Logical Topology in WDM Ring Networks," *Information Sciences : An International Journal, Special Issue on Photonics, Networking and Computing*, 2002.
- [64] A. Sen, B. Hao, B. Shen, and G. Lin, "Survivable routing in WDM networks logical ring in arbitrary physical topology," *Proceedings of the IEEE International Communication Conference ICC02*, 2002.
- [65] J. Gross and J. Yellen, *Graph Theory and its Applications*. CRC Press, 1999.
- [66] A. Frank, *Packing paths, circuits and cuts - a survey (in Paths, Flows and VLSI-Layout)*. Springer, Berlin, 1990.
- [67] G. H. Sasaki, C.-F. Su, and D. Blight, "Simple layout algorithms to maintain network connectivity under faults," *In Proceedings of the 2000 Annual Allerton Conference*, 2000.
- [68] H. Zang, C. Ou, and B. Mukherjee, "Path-protection routing and wavelength-assignment (rwa) in wdm mesh networks under duct-layer constraints," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 248–258, 2003.
- [69] S. Kim and S. Lumetta, "Addressing node failures in all-optical networks," *Journal of Optical Networking*, vol. 1, no. 4, pp. 154–163, April 2002.
- [70] H. Choi, S. Subramaniam, and H.-A. Choi, "On Double-Link Failure Recovery in WDM Optical Networks," *Proc. of IEEE INFOCOM'02*, 2002.
- [71] W. He, M. Sridharan, and A. K. Somani, "Capacity Optimization for Surviving Double-Link Failures in Mesh-Restorable Optical Networks," *Proc. of OptiComm'02*, 2002.
- [72] M. Clouqueur and W. D. Grover, "Mesh-restorable Networks with Complete Dual-failure Restorability and with Selectively Enhanced Dual-failure Restorability Properties,," *Proc. of OptiComm'02*, 2002.

- [73] I. Stoica, "Lecture notes on overlay networks."
- [74] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *Sigmetrics*, 2000.
- [75] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," *ICMCS '97: Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97)*, 1997.
- [76] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *Sigcomm*, 2002.
- [77] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: A framework for delivering multicast to end users," *Infocom*, 2002.
- [78] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," *SOSP'03*, 2003.
- [79] D. Tran, K. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," *Proc. of Infocom*, 2003.
- [80] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," *IEEE Infocom*, 2003.
- [81] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *ACM NOSSDAV*, 2002.
- [82] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, 2003.
- [83] M. Hefeeda, A. Habib, B. Boyan, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast," *ACM Multimedia Conference MM'03*, 2003.
- [84] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," *In Proceedings of IEEE INFOCOM*, 2005.

- [85] R. Zimmermann and L.S.Liu, "Active: Adaptive low-latency peer-to-peer streaming," *SPIE/ACM Multimedia Computing and Networking*, 2005.
- [86] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," *Proc. of Infocom*, 2006.
- [87] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven mesh-based streaming," *Proc. of Infocom*, 2007.
- [88] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, 2007.
- [89] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," *Sigcomm*, 2001.
- [90] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," *Proc. of Infocom*, 2007.
- [91] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," *IEEE Infocom*, 2004.
- [92] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in peer-to-peer gaming," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 79–82, 2007.
- [93] S.-W. Tan, A.G.Waters, and J. Crawford, "Meshtree: A Delay optimised Overlay Multicast Tree Building Protocol," *ICPADS*, 2005.
- [94] S.-W. Tan, G. Waters, and J. Crawford, "A performance comparison of self-organising application layer multicast overlay construction techniques," *Computer Communications*, vol. 29, no. 12, 2006.
- [95] R. Zimmermann, B. Seo, L. S. Liu, R. S. Hampole, and B. Nash, "Audiopeer: A collaborative distributed audio chat system," *Distributed Multimedia Systems (DMS 2004)*, 2004.
- [96] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive fec-based error control for internet telephony," *IEEE Infocom*, 1999.
- [97] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *In SIGMETRICS*, 2003.

- [98] J. Apostolopoulos, “Reliable video communication over lossy packet networks using multiple state encoding and path diversity,” *Proc. Visual Communication and Image Processing, VCIP*, 2001.
- [99] T. Nguyen and A. Zakhor, “Path diversity with forward error correction (pdf) system for packet switched networks,” *Proc. of Infocom*, 2003.
- [100] Y. Li, Y. Zhang, L. Qiu, and S. Lam, “Smarttunnel: Achieving reliability in the internet,” *Proc. of Infocom’07*, 2007.
- [101] J. Han, D. Watson, and F. Jahanian, “Topology aware overlay networks,” *Proc. of Infocom’05*, 2005.
- [102] W. Cui, I. Stoica, and R. Katz, “Backup path allocation based on a correlated link failure probability model in overlay networks,” *ICNP*, 2002.
- [103] C. Tang and P. McKinley, “Improving multipath reliability in topology-aware overlay networks,” *Fourth International Workshop on Assurance in Distributed Systems and Networks (ADSN) (ICDCSW’05)*, 2005.
- [104] T. Fei, S. Tao, L. Gao, and R. Guérin, “How to select a good alternate path in large peer-to-peer systems?” *IEEE Infocom*, 2006.
- [105] Z. Li and P. Mohapatra, “The impact of topology on overlay routing,” *Proc. of Infocom’04*, 2004.
- [106] H. Zhang, J. Kurose, and D. Towsley, “Can an overlay compensate for a careless underlay?” *In Proceedings of IEEE Infocom*, 2006.
- [107] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, “The feasibility of supporting large-scale live streaming applications with dynamic application end-points,” *In Proc. of ACM SIGCOMM*, 2004.
- [108] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, “On the constancy of internet path properties,” *ACM SIGCOMM Internet Measurement Workshop*, 2001.
- [109] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, “An overlay architecture for high quality voip streams,” *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1250 – 1262, 2006.
- [110] W. Wang, D. A. Helder, S. Jamin, and L. Zhang, “Overlay optimizations for end-host multicast,” *Workshop on Networked Group Communications (NGC)*, 2002.

- [111] A. El-Sayed and V. Roca, "On robustness in application-level multicast: the case of hbm," *IEEE Symposium on Computers and Communications*, 2004.
- [112] J. Silber, S. Sahu, J. P. Singh, and Z. Liu, "Augmenting overlay trees for failure resiliency," *Proc. of Globecom*, 2004.
- [113] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu, "Robust and efficient path diversity in application-layer multicast for video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, 2005.
- [114] X. Hoang and Y. Lee, "Dual parent multicast graph for failure resilient peer-to-peer multimedia streaming," *CCNC*, 2006.
- [115] S. Y. Shi, J. S. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," *NOSSDAV '01*, 2001.
- [116] M. Javed, K. Thulasiraman, and G. Xue, "Survivability aware routing of logical topologies: On thiran-kurant approach, evaluation and enhancements," *IEEE Globecom*, 2006.
- [117] —, "Lightpaths routing for single link failure survivability in ip-over-wdm networks," *Journal Of Communication and Networks*.
- [118] —, "Logical topology design for ip-over-wdm networks: A hybrid approach for minimum protection capacity," *ICCCN*, 2008.
- [119] W. Jiang and H. Schulzrinne, "Perceived quality of packet audio under bursty losses," *Proc. of Infocom*, 2002.
- [120] N. F. Maxemchuk, "Dispersity routing in store and forward networks," *Ph.D Dissertation, University of Pennsylvania*, 1975.
- [121] L. Golubchik, J. Lui, T. Tung, A. Chow, W. Lee, G. Franceschinis, and C. Anglano, "Multi-path continuous media streaming. what are the benefits?" *Performance Evaluation Journal*, vol. 39, 2002.
- [122] X. Yu, J. Modestino, and I. V. Bajic, "Modeling and analysis of multi-path video transport over lossy networks using packet-level fec," *Proc. of Distributed Multimedia Systems (DMS)*, 2005.

- [123] E. Vergetis, R. Guérin, and S. Sarkar, “Realizing the benefits of user-level channel diversity,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 15–28, 2005.
- [124] B. Ribeiro, E. de Souza e Silva, and D. Towsley, “On the efficiency of path diversity for continuous media applications,” *Technical Report: UM-CS-2005-019*, 2005.
- [125] H. Levy and H. Zlatokrilov, “The effect of packet dispersion on voice applications in ip networks,” *IEEE/ACM Trans. on Netw.*, vol. 14, no. 2, pp. 277–288, 2006.
- [126] “Planetlab,” <http://www.planet-lab.org/>.
- [127] P. Frossard, “Fec performances in multimedia streaming,” *IEEE Communications Letters*, vol. 5, no. 3, p. 122, 2001.
- [128] X. Yu, J. Modestino, and X. Tian, “The accuracy of gilbert models in predicting packet-loss statistics for a single-multiplexer network model,” *Infocom*, 2005.
- [129] E. O. Elliott, “A model of the switched telephone network for data communications,” *Bell System Technical Journal*, vol. 44, no. 1, p. 89, 1965.
- [130] “Mathematica,” <http://www.wolfram.com/>.
- [131] J. Sommers and P. Barford, “An active measurement system for shared environments,” *Internet Measurement Conference*, 2007.
- [132] “100hotsites,” <http://www.100hotsites.com>.
- [133] A. L. H. Chow, L. Golubchik, J. C. S. Lui, and W.-J. Lee, “Multi-path streaming: optimization of load distribution,” *Perform. Eval.*, vol. 62, no. 1-4, pp. 417–438, 2005.
- [134] D. Rubenstein, J. Kurose, and D. Towsley, “Detecting shared congestion of flows via end-to-end measurement,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 3, pp. 381–395, 2002.
- [135] D. Jurca and P. Frossard, “Media-specific rate allocation in multipath networks,” *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1227–1240, October 2007.

- [136] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, “Best-path vs. multi-path overlay routing,” *Proc. of IMC’03*, 2003.
- [137] S. Tao, K. Xu, Y. Xu, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang, “Exploring the performance benefits of end-to-end path switching,” *in IEEE ICNP*, 2004.

Publications

The study presented in this Ph.D. dissertation led us to the following publications:

Journal papers

- X. Gigandet, P. Hagmann, M. Kurant et. al. “Estimating the confidence level of white matter connections obtained with MRI tractography.” *PLoS ONE*, 3(12): e4006, December 2008.
- M. Kurant, P. Hagmann and P. Thiran. “Error and Attack Tolerance of Layered Complex Networks.” *Phys. Rev. E* **76**, 026103, 2007.
- P. Hagmann, M. Kurant et. al. “Mapping human whole-brain structural networks with diffusion MRI.” *PLoS ONE*, 4(2):e597, July 2007.
- M. Kurant and P. Thiran. “Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction.” *IEEE Journal on Selected Areas in Communications*, Volume 25, Issue 5, June 2007.
- M. Kurant and P. Thiran. “Trainspotting: Extraction and Analysis of Traffic and Topologies of Transportation Networks.” *Phys. Rev. E* **74**, 036114, September 2006.
- M. Kurant and P. Thiran. “Layered Complex Networks.” *Phys. Rev. Lett.* **96**, 138701, April 2006.
- M. Kurant, H. X. Nguyen and P. Thiran. “Survey on Dependable IP-over-Fiber Networks.” *In Springer Lecture Notes in Computer Science (LNCS) 4028 : ‘Dependable Systems: Software, Computing, Networks’* p. 55-81, 2006.
- F. Ducatelle, L. M. Gambardella, M. Kurant, H. X. Nguyen and Patrick Thiran. “Algorithms for Failure Protection in Large IP-over-Fiber and

Wireless Ad Hoc Networks.” *In Springer Lecture Notes in Computer Science (LNCS) 4028 : ‘Dependable Systems: Software, Computing, Networks’*, p.231-259, 2006.

Conference papers

- M. Kurant and P. Thiran. “Achieving Maximal IP Path Diversity in Low-Delay Application-Level Multicast.” *In preparation*.
- M. Kurant. “Exploiting the Path Propagation Time in Multipath Transmission with FEC.” *Infocom 2009*.
- M. Kurant and P. Thiran. “Survivable Routing in IP-over-WDM Networks in the Presence of Multiple Failures.” *EuroNGI Workshop on Traffic Engineering, Protection and Restoration for NGI*, May 4-5, 2006, AGH, Kraków, Poland.
- P. Hagmann, M. Kurant et. al. “Imaging the brain neuronal network with diffusion MRI: a way to understand its global architecture.” *Proc. Intl. Soc. Magn. Res. Med.*, 2006.
- M. Kurant and P. Thiran. “On Survivable Routing of Mesh Topologies in IP-over-WDM Networks.” *Infocom 2005*.
- M. Kurant and P. Thiran. “Survivable Mapping Algorithm by Ring Trimming (SMART) for large IP-over-WDM networks.” *Proceedings of BroadNets 2004*, 25-29 October, San Jose, California, USA.

Maciej KURANT

I am a communicative, energetic and flexible researcher, always full of fresh ideas and ready to turn them into reality. I enjoy taking a risk and discovering new fields.

**Contact:**

Av. du 1er Mai 6B, 1020, Renens, Switzerland
maciej.kurant@epfl.ch
Tel. +41 764023377

Education:

2003 – 2009: Ph.D. studies at EPFL, Lausanne, Switzerland.

- 15 publications, including prestigious conferences and journals.
- Network Science Workshop and Conference (NetSci'06) stipend, May 2006.

2002 – 2003: Graduate School in Communication Sciences, EPFL, Switzerland.

- GPA: 5.75 out of 6 (top 5 out of 30)

1997 – 2002: M.Sc. in Telecommunication, Gdańsk University of Technology, Poland.

- With Honors.
- GPA: 4.89 out of 5.0 - the best student on the year (out of ~100)
- Scholarship of The Minister of National Education of Poland.
- Scholarship of The President of Gdansk for results in studies (twice).
- Award of the Dean for exceptional achievements.

before 2002: Four awards in Olympiads in Physics and Mathematics.

Main areas of interest:

- Various aspects of multilayer systems (survivability in optical networks, IP path diversity in P2P networks),
- Multipath routing with FEC,
- Analysis of large-scale real-life complex networks,
- Human Brain Topology Mapping (based on fMRI data),
- Graph Theory.

Work experience:

Oct 2003–now: Ph.D. studies on “Survivability in Multilayer Networks”, EPFL.

In the theoretical part of my PhD I conducted rigorous and insightful mathematical analysis of various communication systems such as optical networks, multipath transmission and P2P live streaming.

In the practical part of this work I implemented a measurement system working concurrently on about 500 computers distributed around the world (in PlanetLab) and communicating over the Internet.

I was also the leading teaching assistant at the “Introduction to Communication Systems” course for 4 semesters. I was the main coordinator between 3 professors, 9 assistants and 150 students. Additionally, I supervised 5 individual student semester projects.

2005-2007: Side project on Human Brain Topology Mapping.

I co-developed a method to infer the network of long range structural connections in the human brain, based on the large amount of Magnetic Resonance Imaging (MRI) data. It resulted in [this](#) article. The follow-up work (without me on board) got a press coverage e.g., in [The New York Times](#), [Telegraph](#), and [The Scientist](#).

2006-2007: Starting the **ACE** Acronym **CrE**ator at www.acronymcreator.net.

Acronym CrEator is a unique tool to create catchy names that form meaningful acronyms. I invented and developed it independently from my Ph.D., just because I wanted to try out my idea. Roughly 3'000 distinct visitors generate about 20'000 queries to ACE each month.

Professional skills:

- Good programming skills (C++, Python, Mathematica, Matlab, TCP/IP)
- Data mining, processing and analyzing large-scale complex graphs.
- Creativity, diversity, flexibility, efficiency. Good communication skills.

Languages:

English (fluent), French (fair), German (basic), Russian (basic), Polish (mother tongue).

Selected publications (out of the total of 15 publications):

M. Kurant

Exploiting the Path Propagation Time in Multipath Transmission with FEC
INFOCOM 2009

M. Kurant, P. Hagmann and P. Thiran

Error and Attack Tolerance of Layered Complex Networks
Phys. Rev. E 76, 026103, 2007

P. Hagmann, M. Kurant, et. al.

Mapping human whole-brain structural networks with diffusion MRI
PLoS ONE, 4;2:e597, July 2007.

M. Kurant and P. Thiran

On Survivable Routing of Mesh Topologies in IP-over-WDM Network
IEEE JSAC, Volume 25, Issue 5, June 2007 (earlier version presented at INFOCOM 2005)

M. Kurant and P. Thiran

Layered Complex Networks
Phys. Rev. Lett. 96, 138701, April 2006

Organization skills.

I like organizing various events and being responsible for them. I hold (or held) the following functions:

- Co-founder of the Graduate Student Organization at EPFL-IC (2005 – now). I organize numerous cyclic and ad-hoc events such as scientific seminars, mass brainstorming and student integration.
- Chairman of Academic Sport Association in Gdansk (2001-2002). I founded the official Academic Bridge Club at the university. I popularized this intellectual card game by organizing regular trainings and more than 20 tournaments with the average turnout of 60 students.
- Organizer of a student expedition to Mongolia (2001). I found the sponsors, organized the press and radio coverage and led a group of 15 students for 2 months on a route of about 25'000 kilometers. Details at <http://mongolia2001.webpark.pl>

Extra-professional activities.

I am an active person and I feel well in a competitive and challenging environment. Some examples include travelling (South America, Africa, Asia, Europe), skiing (11th in Academic Championship of Poland, 2001), sailing (5th place in International Micro Cup, Poland, 2000), bridge (2nd in Academic Championship of Poland, 2001), mountaineering (Huayna Potosi 6088m, Mount Blanc 4810m), photography (1st place in an academic contest in Gdansk), hitchhiking (1st in an international competition).